

See Sally Work: Techniques for Designing and Evaluating User Interfaces

Judy Loren

UNUM Life Insurance Company of America

Introduction

This paper came about because I discovered a person at UNUM Life Insurance Company who is devoted to improving the quality of computer systems interfaces by applying the principles of usability testing and engineering. I had never heard of usability, but it impressed me as having a lot of native validity. I researched the topic both by reading some literature and by observing the technique in practice. This paper presents an overview of what I learned. I hope it will inspire others to research and apply usability principles in developing computer systems.

The first part of the paper outlines the theory. Following that I summarize some real-world results. I include a description of one particular technique for eliciting user needs, called paper prototyping. Finally I give some suggestions for people who want to try to use this practice.

What is Usability

Usability: a new word for an age-old concept. A usable tool helps the user get the job done better, faster, easier, safer. If it's still not clear, consider the difference between usability and functionality. Functionality is what a tool can do. Usability is how easily a person can make the tool do what it's supposed to do. For instance, does your VCR show the correct time? Or does it blink constantly at 12:00 because it's too much trouble to set it? That's the difference between functionality and usability.

Once you understand the difference between functionality and usability, the tradeoffs between the two become apparent. The more functionality an item has, the more challenging it becomes to make it usable. Take for instance a watch. The old analog watches that told only the time were fairly easy to operate. Wind the watch or replace the battery. Use the stem to set the time. When digital watches with AM and PM settings, dates, and stopwatch features came out, it was no longer obvious even how to set the time correctly, much less adjust for leap years.

So digital watches with all their buttons became the butt of Saturday Night Live jokes.

Applying the concept of usability to computer systems, you might want to ask:

- Is the system easy to learn?
- Is the system internally consistent so that the user can predict the results of his/her actions?
- Does the system make efficient use of the user's time?
- Can casual users remember how to use the system from one effort to the next?
- Does the system minimize user errors?
- Does the user feel a sense of satisfaction from using the system?

To develop a system that meets the above criteria, you can add usability to the list of user requirements and invest early and often in engineering usability into the ultimate product. And to counter the arguments that it is an unnecessary expense, note that you will pay for usability either before you release the system, in engineering efforts, or after, in wasted productivity, revisions, or lack of use.

A Rose By Any Other Name

You might have heard the same ideas discussed under one of the following headings:

HCI	Human Computer Interaction
UCD	User Centered Design
MMI	Man Machine Interface
HMI	Human Machine Interface
OMI	Operator Machine Interface
UID	User Interface Design
HF	Human Factors

Change in landscape

Usability has grown in importance as the cost of computer resources has plummeted compared to the cost of human resources. Where once developers focused on saving disk space and

CPU time, they now must devote themselves to using those resources to conserve operator labor.

What to Test

What kinds of tools can be assessed for usability? Anything people have to deal with, from the handles on refrigerator doors to a nuclear missile launching system. As a SAS user, you want to improve the usability of computer information system interfaces.

How to Assess Usability: The Short Answer

What is the best way to determine how usable an interface is? The answer is both obvious and complex: Ask the user! To the developer, the user represents a source of information s/he can get no other way: how the system will be used.

But if it were that simple to do, just ask the user and fix the system, we wouldn't still be making jokes about Windows 95. The complexities of *how* to ask the user, and *when* to ask the user, and *what* to ask the user, and *what user* to ask, and *what to do* with the answer make up the bulk of this paper.

Cornerstone

If there is one linchpin, one essential ingredient, one key element that determines the ultimate usability of a system it is this advice to the developer:

Know thy user, for (s)he is not you.

This means, for instance, that one developer asking another in the next cube to test an interface will not suffice to maximize usability. Four developers sitting in a room discussing potential alternative workflows, trying to guess what the user "would" do, does not constitute usability engineering. Even managers and supervisors of the eventual users cannot substitute for the users themselves.

The foremost obstacle to improving the usability of a product or system is the tendency of developers to blame usability problems on the user. The user needs more training. The user has to understand the developers' problems. The user doesn't "get it". The user should read the manual.

These kinds of comments reflect an attitude not uncommon in any human encounters: that the speaker's style is "right" and the listener's style is "wrong". Attempts to counter this assumption form the basis for ancient myths as well as modern stories, and provide the foundation for current corporate efforts at diversity training.

Only from a foundation of respect for the user as the ultimate judge can you build a process for improving the usability of a system.

Do not construe this to mean that users are always right. Believe what they say when they describe their actual experience, but do not take as fact what they say about what they would like or dislike. In particular, do not be misled by what users say in an abstract discussion. Make sure you measure and validate by observing what users actually do. Prototype and test.

Assessment vs Engineering

Techniques for assessing usability do not in themselves improve the usability. They only provide feedback to the developers about what works and what doesn't. Taking those results and actually improving the product constitute usability engineering. And as anyone knows who has developed even one system, the earlier you get information on user requirements, the easier and cheaper it is to adjust the system.

So you can put off the usability test until the product is almost ready to be released, but the chances that any significant problems encountered will actually be addressed before release are minuscule.

So how do you engineer usability into a product? Involve users from the beginning and allow their needs to drive decisions as product development proceeds through several iterations. If the concept of usability is new to your organization, it might help to get a usability specialist on the project team to help conduct the assessment process and to make sure usability gets the attention it deserves.

Allowing user needs to drive decisions can be more difficult than it seems. For one thing, those needs must take precedence even over such accepted goals as storing data efficiently, maintaining consistent architecture, using existing technical skills, or especially avoiding change from what currently exists. Choices that

seem expensive up front can often be cost-justified by knowing the alternative cost of unusable systems or inefficient workflow.

Cost-Benefit Justification

If anyone still doubts the benefits of usability engineering, consider these costs of an unusable system:

- Fixing problems later
- Support calls
- Training
- Lost productivity
- Higher cost of "expert" operators
- Improper use through confusion, resulting in wrong data

IBM conducted a cost benefit analysis and found that it cost \$20,000 to engineer usability into a small system, but the benefit was \$41,700. For a large system, an investment of \$68K resulted in annual savings of \$6.8MM.

The four biggest reasons for cost overruns on software projects are:

- frequent requests for changes by users
- overlooked tasks
- users' lack of understanding of their own requirements
- insufficient user-analyst communication and understanding

Usability engineering, with its focus on communication between user and developer, addresses all of these.

One caveat: A small system designed for a few skilled users may not justify a lot of effort in improving usability, unless the system will be used in life and death situations, like a hospital monitor.

Usability Testing: Overview

The basic usability test consists of developers watching users attempt certain tasks using the new system (or a prototype). Valuable information comes from both the user's verbalized thought processes as s/he works, and from quantitative measures such as time to complete the task and number of errors the user makes. Strict usability engineering involves having pre-set goals for time and error counts,

but those goals can be problematic to set. For example, how long should it take to save a record? What is acceptable? 10 seconds? 5 seconds? If you set a quantitative goal and achieve it, but the user is still dissatisfied, is the system usable?

This paper will focus on the essential elements of usability testing, or usability on a shoestring.

Usability Testing: Design

Goals: The primary goal of usability testing is *to improve the usability of the product*. A secondary goal can be to improve the process of assessing usability for the next round. These goals may sound obvious, but they are not the same as testing the functionality of a product (although improper functionality can be revealed in a usability test, and it can certainly confound the results). Nor should the goal of testing be stated as "to make sure the product is usable."

You can also resolve differences in opinion among development group members about how a system ought to work by testing with users. Put some users to work on a prototype with targeted tasks to decide questions of design philosophy. Sometimes questions raised by others outside the project development staff, such as planners or technical communicators, drive usability tests.

Usability testing can also help you choose among packaged software to meet an internal need. Functionality testing alone, although much more common, will not address all the necessary issues.

Participants: Real users, at least 3 and as many as 12, one or two at a time. Depending on your ultimate user community, you may want a simple random sample of users, or you may want to stratify your sample to include different kinds of users.

Some of the user characteristics you may want to consider differentiating are:

- amount of computer background
- amount of business background
- age
- amount of industry experience
- kind of computer background

These will probably affect the way the user approaches the system, his/her assumptions

about how it ought to work and what kind of terminology s/he is expecting for the business functions.

You want several users in your sample, to avoid having one user's idiosyncrasies affect system development. Encountering several users with the same issues also prevent developers from discounting usability problems as user idiosyncrasies!

Facilitator: Someone interested and trained, or preferably experienced, in usability assessment. The facilitator handles most of the interaction with the participants, before, during and after the test. S/He should be aware of the legal issues and international guidelines for treatment of human test participants.

Human test participants have the right to:

- Know about any risks
- Ask questions
- Withdraw at any time
- Ask for a break
- Maintain privacy
- Understand purpose and procedures

The privacy guideline, for example, dictates that the supervisor of the participant should not observe his/her test. You should not present findings of any individual test, or at least not identified with a particular participant. You should tell the participant that you will use the materials and test results only for improving the product, not for assessing him/her in any way.

The care you take of your participants shows that you take the test very seriously.

Observers: Developers, managers, trainers, technical writers. In the ideal situation, the participants sit in a room by themselves with the computer running the system. The developers and other observers sit in another room, watching and listening through the miracle of modern technology. A fully equipped usability lab has video cameras on the user, a communication system between the participant and the observers, and a video monitor in the observers' room that duplicates the participant's monitor so the observers can see what is happening while they listen to the participant talk.

The most important guideline for observers is to avoid any hypotheses or beliefs about the system. It might help to think of yourself as an anthropologist, observing and trying to understand human behavior. Assume there are lots of opportunities to improve system design (as opposed to assuming you are there to prove the system is usable as is). Censor tendencies to defend the design. As the test proceeds, go ahead and hypothesize potential solutions but when the test is over, discuss your interpretations with other observers and clarify differences with participant. Make sure the participant supports your interpretations before pursuing solutions.

Activities: Real tasks. A good usability test will consist of several scenarios of data and instructions to the user to accomplish real-world tasks with those scenarios. For example, if the screens that are being tested are data entry screens, the test should consist of actual documents (perhaps with dummy data) from which the users will be or already are working, and instructions to enter the data into the database. The amount of training you give to the participants should approximate the training they would receive in the real world. You should err on the side of too little training, and find out what users will do when they don't know how to proceed and what assumptions they will make about the screens, rather than compromise your test results by too much guidance.

A key piece of usability testing, perhaps the most beneficial to the attempts to solve the problems encountered, is understanding the participant's thought process. Therefore, participants should be asked directly to "think out loud" and explain everything they do. The developers watching the process do not respond to these verbalizations in any way except to record their observations.

Schedule: It can take a morning to run multiple users through a short usability study, or it can take a month for a complex stratified sample to test a large system. In general, no user should spend more than half a day participating in a test. Longer than that will cause fatigue to become a major factor, which will confound your results (unless that is in fact what you want to test). Similarly, developers, in their role as observers, will lose the ability to focus on what they are hearing and seeing after several concentrated hours.

Results: Improved systems, provided the developers use the information collected in usability testing. The first opportunity for developers to watch actual users attempting to be productive with their system can also serve as a catalyst to accomplish the attitude change described above. Sometimes the usability testing process dramatically changes the developers' priorities and understanding of the goals of the project.

"People who have been developing complex products but have never systematically watched users set up, learn, or use those products find watching a usability test fascinating, shocking, sometimes humiliating and painful, but invariably eye-opening. ... The experience of attending a test ... is often enough to stimulate both a demand for more testing and a discussion about how to improve usability." [Dumas & Redish, p. 89]

Observations made during the test can consist of descriptions of problems encountered and/or measurements of time and errors. For developers wanting to assess users' response to a prototype, the users' comments can suffice for results. If you are trying to compare the usability of several pieces of software, it will help you to have some quantitative measures for direct comparison. Quantitative measures also increase the impact of a presentation to management about the need to improve a system.

Optimal results occur only when the test observers succeed in finding the real problems, not just the symptoms. Sometimes one global problem looks like several local problems. Defining the true problem takes inspiration, or even genius, on the part of the developers. Making good use of the usability test process helps.

Reporting: Oral communication among interested parties should take place throughout the testing process. Used appropriately, the process will energize developers to rework their design and improve the software. A formal write-up may not add value to that piece, but management may require it. It also serves to document the results for future iterations of the test process. Other systems can benefit from what you discover.

When you write, avoid long prose. Think of your audience as users of your write-up rather

than readers. Make sure you mention the positive feedback, things that did not require changes, as well as negative findings. Including verbatim comments from participants will liven up your message and bring it home to people who couldn't attend the test. Finally, allow members of the test team to review the report and make the necessary revisions before publishing it.

Usability Testing: Alternatives

Other ways of getting user feedback about a system include focus groups, surveys, and beta testing. Each of these has its drawbacks. Focus groups typically do not let you observe users behaving with the product. Since users are by definition not developers, they may not be 100% accurate in their ideas about what would really make them more productive. Surveys have that same problem, plus they can be misunderstood. Beta testing doesn't provide feedback in time to make any real difference in the end product.

Complement Usability Testing with Heuristic Evaluation

One technique for improving usability does work effectively as a complement to usability testing, and that is heuristic evaluation. This can be done by a developer or by someone experienced in the field of human-computer interaction. It consists of testing the system against a set of rules for good usability. Conducting this evaluation and using the results to improve the system before requiring user participants saves valuable user time.

Several authors and organizations have published suggested principles for good interface design. Not surprisingly, the list includes IBM (Systems Application Architecture: Common User Access Advanced Interface Design Guide) and Microsoft (The Windows Interface: An Application Design Guide). One place you might start for a larger bibliography is ACM's SIGCHI (Association for Computing Machinery's Special Interest Group on Computer Human Interaction).

Some examples of these usability principles are:

- Give the operator control
- Minimize the operator's work (allow shortcuts)
- Be consistent

- Give feedback (e.g., "One record saved to database.")
- Minimize dependence on recall
- Help the operator stay oriented
- Offer error handling
- Permit easy reversal of actions

Keeping these principles in mind, a developer can detect and avoid many usability problems. A group of developers working together to evaluate a system can be even more effective.

Pre-Design Work

What can you do before designing the system to engineer in usability?

As a first step, try to adopt the attitude described above of respecting the user as the ultimate arbiter of the value of the system. That naturally leads to some investment in getting to understand the prospective users of a new system. Visit some and observe their workflow.

Frank Gilbreth, a turn of the century efficiency expert whose family and professional lives were immortalized by his children in the biography Cheaper by the Dozen, used to ask to be shown the "laziest man on the site." He claimed that the lazy man would already have figured out how to minimize his own effort (an early form of "Work Smarter, Not Harder"). Similarly, as you visit users, don't necessarily focus on the go-getters. Be sure to observe the quieter or less active users for they may have valuable insights.

Some of the questions you may want to ask yourself as you observe potential users are: Will this system be central to the user's job function? Or is it supposed to make a peripheral task faster and easier? During use, will the typical user be focused primarily on the system? Or on something else such as a customer? Will the user be interrupted frequently while using the system? Where will the PC or terminal be located physically with respect to the user's primary job site? Will each user have his/her own PC or will they have to share?

These kinds of factors may not be addressed during requirements definition or functionality testing, but they have significant effects on the usability of the system. One way to get at these factors early enough in the process to incorporate them easily is to use prototyping. The sooner the users can get something concrete to try out, the

sooner you can learn more specifically what their needs are.

Paper Prototyping

One technique that has worked well to identify user requirements early is paper prototyping. The word paper refers to the materials used in developing the prototype. This method helps determine user requirements not only for functionality but for usability as well.

The goal of paper prototyping is to create a visual model (no code) that system requirements can be derived from.

To develop a paper prototype, you gather some users in a room with a facilitator and some tools. These people work together to develop samples of what the screens of a new system might look like and how the user will navigate among them. Because it is concrete rather than abstract, it can help users define functional requirements more precisely than the usual requirements definition sessions, which are primarily discussions.

What kinds of tools do you use? Mostly Post-It notes. Different sizes and different colors represent different functions or concepts. Have huge paper backdrops on the walls to represent computer screens and stick the Post-It notes on to compose a paper mock-up of each screen in the system. Use separate pieces of paper for each drop-down menu and dialog box.

Why paper? First, it's fast and easy. You can process many iterations in a short period of time. Second, anyone, not just a programmer, can be a developer. Illustrations communicate ideas among participants more clearly than verbal descriptions. Finally, when you're done you have a tool to present the design to others. Not bad for a week's work.

How do you do it? Besides using low fidelity, easy to change materials like paper, the paper prototyping process relies on a time-driven iteration process. The facilitator keeps the participants moving by setting goals and limiting the amount of time the group spends getting there. The process consists of four steps which the participants repeat as necessary to perfect the design.

- Plan: What do you need to find out?

- **Implement:** Build just the piece(s) of the product you need.
- **Measure:** Find out how users work with it, see where problems are.
- **Learn:** What changes can you make to get back on track?

The more frequently you measure and react, the closer to your goal you will get. Consider an airplane flying from Washington, DC to Chicago. It probably doesn't follow a linear path, but it constantly checks the adjustment it has to make to stay on course.

Another analogy for the iteration process is long division. You make a guess (Plan), do the multiplication (Implement), compare the result (Measure), and decide whether that guess needs revision (Learn).

Rapid Iteration the Key

The most surprising, and almost counterintuitive, guideline for developing paper prototypes is that it doesn't matter how good the first iteration is. The more iterations you can get through in a given period of time, the better the final result. This argues not only for putting time limits on the group's activities in a paper prototyping session, but for the whole concept of paper prototyping over desk design by one or several experienced developers. The sooner you can get something in front of the users to evaluate, and the more opportunities they have to provide feedback, the more usable the ultimate product will be.

Scenarios

How do you measure the prototype (step 3 in the 4 step process)? Construct scenarios of tasks that will be done with the new system and step through them using the prototype. The first scenarios should be common or mainstream activities. As the prototype is refined, and it seems to be handling common activities well, start testing it on unusual or fringe activities.

Since the scenarios you choose constitute the primary means of honing the prototypes, it is critical to have the entire range of tasks the new system will be expected to handle represented in the assortment. Constructing the scenarios in advance, while you still have perspective on the total system, allows you to devote the appropriate amount of attention. When you have

a particular prototype in mind to test, you might give certain aspects of the system more or less focus than they deserve, resulting in a prototype that ultimately doesn't do the job.

User Feedback

One group that used paper prototyping even before formal requirements definition had some useful feedback:

- Small groups (4-5 people) are more effective when doing specific activities.
- Use entire team to set the base level.
- Remember there is no right or wrong way to do it.
- Listen to each other.
- Respect each other.
- Participate!
- Need to make sure the team shares a common understanding of the existing processes that the system will address.
- It helps to have a facilitator with experience or training in the paper prototyping technique.
- Dividing the proposed activities for the system into manageable chunks helped get through it faster.
- Need to solicit input from everyone (go around the room) rather than let a vocal few dominate.
- A chart showing what has been accomplished helps keep the energy level up toward the end of a long session.
- People need to allocate the time for this and give it priority. Missing people wastes everyone's time.
- Keep a bias toward building the prototype first, then discuss, rather than letting discussion delay building.
- Do a high level functional road map to start the process. Then assign chunks to subgroups.

Who Uses Usability?

The American Institutes for Research's Usability Test Laboratories are the foremost independent laboratories for user testing in this country. In 1997, Oracle Corporation retained them to test its data query tool Oracle Discoverer 3.0 against Business Objects 4.0 and Cognos Impromptu 4.0/PowerPlay 5.2. Oracle posted the summary report on its website at <http://www.oracle.com/products/tools/disc/collateral/air/air.html>.

Intuit, maker of Quicken and other software, focuses on usability throughout the software development process. The president makes a habit of answering customer support calls, to stay in touch with user difficulties. The development manager goes to new users' homes to observe them as they try to install and start up Intuit software. The result is a small sales force, as word of mouth by satisfied customers suffices to market the product.

Usability in Your Organization

If you want to find out whether a focus on usability would benefit your organization, try testing an existing system. Either hire a usability specialist (and the number is growing—contact one of the organizations listed at the end of this paper) or send an internal resource out for training.

To gain management support, stress the increased productivity from a usable system. If you market software as a profit center, a usable system will increase your market share.

Appointing a specific usability resource will prevent the effort from being eliminated or overlooked as time goes on. The more you integrate a focus on usability into early system design, the better your results and the smaller the investment.

You will have more success in implementing a usability program if you take care to tailor the principles to fit your organization and its projects. It is not a one-size-fits-all product.

You should make it a policy to test *all* user interfaces for usability. Remember that if you don't test your system for usability, your users will.

Contact Organizations

American Institutes for Research
490 Virginia Road
Concord, MA 01742
www.air-dc.org

Association for Computing Machinery (ACM)
Special Interest Group on Computer Human
Interaction (SIGCHI)
1515 Broadway
New York, NY 10036
www.acm.org/sigchi/

User Interface Engineering
800 Turnpike Street, Suite 101
North Andover, MA 01845
(508) 975-4343
<http://world.std.com/~uieweb>

Usability Professionals' Association
414 Plaza Drive, Suite 209
Westmont, IL 60559
Phone: (630) 655-1647
Fax: (630) 655-0391
office@upassoc.org
www.upassoc.org

The Society for Information Display
31 East Julian Street
San Jose, CA 95112
Phone: (408) 977-1013
Fax: (408) 977-1531
E-mail: office@sid.org
www.sid.org

References

Dumas, Joseph S. and Redish, Janice C., A Practical Guide to Usability Testing, Norwood, NJ: Ablex Publishing Corp., 1993.

Gilbreth, Frank B., Jr. and Carey, Ernestine
Gilbreth, Cheaper by the Dozen, New York, NY:
Thomas Crowell Company, 1948.

Nielsen, Jakob, Usability Engineering, Chestnut Hill, MA: Academic Press, Inc., 1993.

Acknowledgement:

I am indebted to Kevin Frye of UNUM Life Insurance Company of America for turning me on to usability, sharing his private library with me, and encouraging me to watch and listen during paper prototyping efforts and usability labs.

For more information, contact the author:

Judy Loren
P. O. Box 306
Cumberland, ME 04021
JLoren@Worldnet.ATT.net