Iterative User-Interface Design

Jakob Nielsen, Bellcore

Four case studies show that redesigning user interfaces on the basis of user testing — iterating through at least three versions — can substantially improve usability.

Because even the best usability experts cannot design perfect user interfaces in a single attempt, interface designers should build a usability-engineering life cycle around the concept of iteration.[1-5] Iterative development of user interfaces involves steady design refinement based on user testing and other evaluation methods. Interface designers complete a design and note the problems several test users have using it. They then fix these problems in a new iteration, which they test again to ensure that the "fixes" did indeed solve the problems and to find any new usability problems introduced by the changed design.

Normally, the design changes from one iteration to the next are local to the specific interface elements that caused user difficulties. An iterative design methodology does not involve blindly replacing interface elements with new, alternative design ideas. If designers must choose between two or more interface alternatives, they can perform comparative testing to measure which alternative is the most usable. However, such tests are usually viewed as constituting a methodology different from iterative design as such, and they are most often devised to measure rather than find usability problems. Iterative design aims specifically at refinement based on lessons learned from previous iterations.

The user tests I present in this article were rigorous, with a fairly large number of test subjects measured carefully in several different ways while performing a fixed set of tasks for each system. In many practical usability-engineering situations,[6] however, we can gain sufficient insight into the usability problems in a design iteration with only a few test subjects, and it may not be necessary to collect quantitative measurement data. The quantitative measures emphasized in this article may be useful for management in larger projects, but they are not the main goal of usability evaluations. Instead, the principal outcome of a usability evaluation in a practical development project is a list of usability problems and suggestions for interface improvements.

After a general discussion of iteration and usability metrics, I present four examples of iterative user-interface design and measurement.

Benefits of iteration

To show the value of iteration in usability engineering, Karat[7] analyzed a commercial development project in which the user interface to a computer security application was tested and improved through three versions. For a lower-bound

0018-9162/93/1100-0032$03.00 © 1993 IEEE

estimate of the value of the user-interface improvements, she calculated the time saved by users due to quicker task completion, thus leaving out any value of the other improvements. The security application had 22,876 users. Each could be expected to save 4.67 minutes by using version 3 rather than version 1 to perform a set of 12 initial tasks (corresponding to one day's system use), for a total savings of 1,781 work hours. From this, Karat estimated saved personnel costs of $41,700, which compared very favorably with the increased development costs of $20,700 for iterative design. The true savings were likely to be considerably larger, since the improved interface was also faster after the first day.

Figure 1 shows a conceptual graph of the relation between design iterations and interface usability. Ideally, each iteration would result in an interface better than the previous version, but as I show later, this is not always true in practice. Some changes in an interface may turn out not to be improvements. Therefore, the true usability curve for a particular product would not be as smooth as the curve in Figure 1. Even so, Figure 1 reflects the general nature of iterative design, though we do not yet have enough documented case studies to estimate the curve precisely.

The first few iterations will in general probably result in major usability gains as interface designers find and fix the true "usability catastrophes." Later iterations have progressively smaller potential for improvements because the major usability problems are eliminated, and the design may eventually become so polished that very little potential for further improvement remains. Because the number of documented cases is small, we do not yet know the point of diminishing returns in terms of number of iterations. We do not even know for sure whether there is an upper limit on the usability of an interface or whether it would be possible to improve some interfaces indefinitely with continued substantial gains for each iteration.

Assuming that designers stay with the same basic interface and keep refining it, I feel that there is a limit to the level of usability they can achieve. At the same time, I believe that designers can often break such limits by rethinking and completely redesigning the interface as they gain more experience with

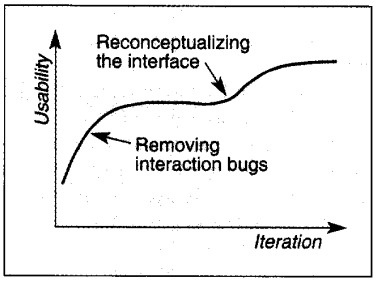

Figure 1. The conceptual relation between interface usability and number of design iterations.

it over time. For example, the task of programming a computer has been reconceptualized several times, with languages changing from octal machine code to mnemonic assembler to higher level programming languages.

Figure 1 shows a reconceptualization after a long period of stable usability for a system, but we do not really know what leads to the creative insights necessary for a fundamentally novel and better interface design. Interface reconceptualizations may not always be immediately followed by the increased usability indicated in Figure 1. A fundamental redesign may introduce unexpected usability problems that designers would have to iron out with a few additional iterations.

We know of interface reconceptualizations in individual development projects mainly through anecdotal evidence. They have not been documented or had their usability impact measured. One exception is the development of an electronic white pages system by a large telephone company. The system was intended to let customers with home computers search for telephone numbers. Search terms could be matched either exactly or phonetically, and users could expand searches to a larger geographical area.

Unfortunately, even after the iterative design had been through 14 versions, test users still said they were intimidated and frustrated by the system. At this stage, the designers decided to abandon the basic interface design, which was based on a command-line interface. From the insight that the command line made the system's structure invisible to the user, they reconceptualized the interface. The redesigned interface had a menu of services displaying all options at all times, thus making them visible and presumably easier to understand. User satisfaction on a 1-to-5 rating scale (where 5 corresponded to "like very much") increased from 3.4 for the command-line-based version to 4.2 for the menu-based version, indicating a drastic improvement in usability for the reconceptualized interface.

Usability metrics and overall usability

A system's overall quality is actually a sum of many quality attributes, only one of which is usability. Additionally, a system should be socially acceptable and practically feasible with respect to cost, maintainability, and so on. The system should fit users' job needs and let them produce high-quality results, since that is the reason for having the system at all. I do not consider these issues here because they are related to *utility*: whether the system's functionality can do what is needed. This article focuses on usability, that is, on *how well* users can use that functionality. (The concept of utility is not necessarily restricted to work-oriented software. Educational software has high utility if students learn from it, and an entertainment product has high utility if it is fun.)

Despite simplified conceptual illustrations like Figure 1, usability is not really a one-dimensional property of a user interface. A system is usable if it is[6]

- easy to learn, so users can go quickly from not knowing the system to doing some work;
- efficient, letting the expert user attain a high level of productivity;
- easy to remember, so infrequent users can return after a period of inactivity without having to learn everything all over;
- relatively error-free or error-forgiving, so users do not make many errors, and so those errors are not catastrophic (and are easily recovered from); and
- pleasant to use, satisfying users subjectively, so they like to use the system.

Focusing on usability as a quality goal, we can define it in terms of these five attributes. We numerically characterize each attribute with a metric. In the usability field, such metrics are com-

monly defined in terms of the performance of randomly chosen representative users who do representative, benchmark tasks with the system. Usability measures are empirically derived numbers resulting from one or more such metrics applied to a concrete user interface. This progression from quality goals to quality attributes and their metrics, then to the actual measures, makes usability steadily more concrete and operationalized.

We should not necessarily pay equal attention to all usability attributes. Some are more important than others and should be the focus of development efforts throughout the iterative design. In some cases, we might even accept lower scores on certain usability metrics as long as their values remain above some minimally acceptable value.[8] Each case study I discuss here had its own prioritized set of usability attributes, and other projects might have yet other goals. (It is important to prioritize usability attributes early in the project's usability life cycle.)

User efficiency is a measure of how fast users can use the computer system to perform their tasks. User efficiency is different from computer response time, even though faster response times often let users complete their tasks more quickly. Because of its direct bottom-line impact, user efficiency is often seen as the most important attribute of usability. For example, Nynex (the regional telephone company serving New York and New England) estimates that each one-second reduction in work time per call for its toll and assistance operators will save three million dollars per year.[9] However, other usability attributes may be critical for applications that monitor or control dangerous industrial plants or processes, where an especially low frequency of user errors is desired.

Normally, the values of the chosen usability metrics are measured empirically through user testing. Some research projects aim at analytical methods for estimating certain usability metrics, and they have had some success in quantifying expert-user performance[9] and the transfer of user learning from one system to another. These analytical methods are far less successful at estimating error rates, however, and they are unable to address the subjective "pleasant-to-use" dimension of usability. In any case, practical development projects should probably stay with user testing for the time being, since the analytical methods can be rather difficult to apply.[10]

Often, usability engineers engage about 10 test users for usability measurements, though sometimes they conduct tests with 20 or more users for tight confidence intervals on the measurement results. It is important to pick test users who represent as closely as possible the people who will actually use the system after its release. For repeated user tests, designers should use different users for each test to eliminate transfer of learning from previous iterations.

It is normally quite easy to conduct user testing with novice users to measure learnability and initial performance, error rates, and subjective satisfaction. Usability measures for expert users are unfortunately harder to come by, as are measures of experienced users' ability to return to the system after a period of absence. Such tests require users with actual experience in using the system. For some user interfaces, full expertise may take several years to acquire, and even for simpler interfaces, users normally train for several weeks or months before their performance plateaus at the expert level.

Such extended test periods are infeasible during new-product design, so expert performance is much less studied than novice performance. Two ways of obtaining expert users without having to train them are to involve expert users of any prior release of the product and to use the developers themselves as test users. Of course, a developer has a much more extensive understanding of the system than even an expert user, so tests should include some real users, too.

Quantifying relative changes in usability. In this article, I use a uniform scoring of usability improvements to compare the different iterative design projects. The approach normalizes all usability measures with respect to the values measured for the initial design. Thus, the initial design has a normalized usability value of 100, and I normalize the subsequent iterations by dividing their measured values by those measured for the initial design. For example, if users made four errors while completing a task with version 1 and three errors with version 2 of an interface, the normalized usability of interface 2 with respect to user errors is 133 percent of the usability of interface 1, indicating a 33 percent improvement in usability. In the same way, the normalized usability of interface 2 would still be 133 percent if users made eight errors with interface 1 and six errors with interface 2.

Converting measures of subjective satisfaction into improvement scores is more challenging, because subjective satisfaction is often measured with questionnaires and rating scales. The underlying problem is that rating scales are not ratio scales: It is theoretically impossible to arrive at a perfect formula to calculate how much better one rating is relative to another. We cannot simply take the ratio of the raw rating-scale scores. Doing so would mean, say, that a system rated 2 on a 1-to-5 scale would be twice as good as a system rated 1. However, a system rated 2 on a 1-to-7 scale would also be twice as good as a system rated 1, even though it is clearly easier to achieve a rating of 2 on a 1-to-7 scale than on a 1-to-5 scale.

Assuming usability engineering has a purely economic goal, we might be able to gather empirical data to estimate how various subjective-satisfaction ratings translate to increased sales of a product or increased employee happiness. Unfortunately, I am not aware of any such data, so I calculated relative subjective-satisfaction scores using a method proven successful in a larger study of subjective-preference measures (see sidebar).[11]

The case studies in this article all used a fixed set of tasks to measure usability throughout the iterative design process. This was possible because developers defined the systems' basic functionality before the start of the projects. In contrast, some development projects follow a more exploratory model, where the tasks users perform with a system may change during the development process as designers learn more about the users and their needs. For such projects, it may be more difficult to use a measurement method like the one I outline here, but the general results showing improvements from iterative design should still hold.

Calculating scores for overall usability. A single measure of overall user-interface usability is often desirable. For example, interface designers typically can choose between several alternative solutions for various design issues and want to know how each alternative affects the resulting system's overall us-

ability. As another example, company management wanting a single spreadsheet as its corporate standard would need to know which of the many available spreadsheets was the most usable. The interface designers balance usability against other considerations such as implementation time for the design alternatives, and the corporate managers balance usability against purchase price for the spreadsheets.

Ultimately, the measure of usability should be monetary to allow comparisons with other measures such as implementation or purchase expenses. For some usability metrics, economic equivalents are fairly easy to derive, given data about the number of users, their loaded salary costs, and their average workday. (Loaded salaries include not just the money paid to employees but also any additional costs, such as social security taxes and medical and pension benefits.) For example, I recently performed a usability evaluation of a telephone company application. I estimated improvements in the user interface would reduce training time half a day per user and increase expert-user performance 10 percent. Given the number of users and their loaded salaries, these numbers translated to savings of $40,000 from reduced training costs and about $500,000 from increased expert-user performance in the first year alone. In this example, the overall value of the usability improvements is obviously dominated by expert-user performance — especially since the system will probably be used for more than one year.

Other usability metrics are harder to convert into economic measures. In principle, user-error rates can be combined with measures of the seriousness of each error and estimates of the error's impact on corporate profits. The impact of some errors is easy to estimate. For example, an error in the use of a print command causing a file to be printed on the wrong printer has a cost approximately corresponding to the time needed for the user to discover the error and walk to the wrong printer to retrieve the output. An error causing a cement plant to burn down has a cost corresponding to the plant's value plus any business lost while it is rebuilt. In other cases, error costs are harder to estimate. For example, an error causing a customer to be shipped the wrong product has direct costs corresponding to the administrative costs of handling the return plus the wasted freight charges. However, much larger indirect costs could result if the company were to receive a reputation for being an unreliable supplier, causing customers to do business elsewhere.

Subjective satisfaction is perhaps the usability attribute with the least direct economic impact in the case of software for in-house use, even though it may be one of the most important factors influencing individual purchases of shrink-wrap software. Of course, it is conceptually easy to consider the impact of increased user satisfaction on software sales or employee performance, but actual measurements are difficult and were not available in the case studies discussed here.

Because using monetary measures of all usability attributes is difficult, I chose an alternative approach here. I calculated overall usability in relative terms as the geometric mean (the nth root of the product) of the normalized values of the relative improvements in the individual usability metrics. Doing so involves certain weaknesses: First, all the usability metrics measured for a given project are given equal weight, even though some may be more important than others. Also, there are no cutoff values where any further improvement in a given usability metric would be of no practical importance. For example, once the error rate has been reduced to, say, one error per 10,000 user transactions, it may make no practical difference to reduce it further. Even so, reducing the error rate to one error per

Transforming subjective ratings to a ratio scale

To ensure uniform treatment of all subjective rating-scale data, I applied three transformations to the raw rating-scale scores:

(1) Rescale the various rating scales linearly to map onto the interval from −1 to +1, with 0 as the midpoint and +1 as the best rating. This transformation provides a uniform treatment of scales independent of their original scale intervals.

(2) Apply the arcsine function to the rescaled scores. This transformation compensates for the fact that rating scales are terminated at each end, making it harder for an average rating to get close to one end of the scale. For example, an average rating of 4.6 on a 1-to-5 scale might result from four users rating the system 4.O and six users rating it 5.0. Now, even if all 10 users like some other system better and would like to increase their rating by one point, the second system would get an average rating of only 5.0 (an increase of 0.4 instead of 1.0, as deserved), because six users were already at the end of the scale. Because of this phenomenon, changes in ratings toward the end of a rating scale should be given extra weight. The arcsine function achieves this by stretching the ends of the interval.

(3) Apply the exponential function e^x to the stretched scores to achieve numbers for which ratios are meaningful. Without such a transformation, a score of 0.2 would seem twice as good as a score of 0.1, whereas a score of 0.7 would be only 17 percent better than a score of 0.6, even though the improvement was the same in both cases. After the exponential transformation, a score of 0.2 is 10.5 percent better than a score of 0.1, and a score of 0.7 is also 10.5 percent better than a score of 0.6.

Admittedly, the choice of functions for these transformations is somewhat arbitrary. For example, an exponential function with a basis other than e would achieve the same effect, while resulting in different numbers. However, the purpose of my analysis is to compare relative ratings of the various interface iterations, and I applied the same transformations to each original rating. Nowhere do I use the value of a single transformed rating. I always look at the ratio between two equally transformed ratings, and all the transformations are monotonic. Also, an analysis of a larger body of subjective-satisfaction data indicated that these transformations result in reasonable statistical characteristics.[11]

Table 1. Four case studies of iterative design.

Name of System	Interface Technology	Versions Tested	Subjects per Test	Overall Improvement (percent)
Home banking	Personal-computer graphical user interface	5	8	242
Cash register	Specialized hardware with character-based interface	5	9	87
Security	Mainframe character-based terminal	3	9	882
Hypertext	Workstation graphical user interface	3	10	41

20,000 transactions would double the usability score for the user-error metric as long as no cutoff value has been set for user errors. Using geometric means rather than arithmetic means (the sum of values divided by the number of values) to calculate the overall usability somewhat alleviates this problem. A geometric mean gives comparatively less weight to uncommonly large numbers. The geometric mean increases more when all the usability metrics improve a little than when a single metric improves a lot and the others are stagnant.

1. Basic tasks operating on the customer's own accounts
   - Find out the balance for all your accounts.
   - Transfer an amount from one of your accounts to the other.
   - Investigate whether a debit-card transaction has been deducted from the account yet.
2. Money transfers to accounts owned by others
   - Order an electronic funds transfer to pay your March telephone bill.
   - Set up a series of electronic funds transfers to pay monthly installments for a year on a purchase of a stereo set.
   - Investigate whether the telephone-bill transfer has taken place yet.
3. Foreign exchange and other rare tasks
   - Order Japanese yen corresponding to the value of 2,000 Danish kroner.
   - Order 100 Dutch guilders.
   - Order an additional account statement for your savings account.
4. Special tasks
   - Given the electronic funds transfers you have set up, what will the balance be on August 12? (Do not calculate this manually; find out through the system.)
   - You returned the stereo set to the shop, so cancel the remaining installment payments.

Figure 2. User tasks to test the prototype home-banking system.

Table 2. Absolute values of the constituent measurements of the "task time" usability metric for the home-banking system (all times in seconds).

Version	Basic Tasks (own account)	Transfers to Others' Accounts	Foreign Exchange	Special Tasks	Total Task Time (seconds)
1	199	595	245	182	1,220
2	404	442	296	339	1,480
3	211	329	233	317	1,090
4	206	344	225	313	1,088
5	206	323	231	208	967

Case studies

Table 1 summarizes the four case studies in iterative user-interface design that I review here. I discuss the first example in somewhat greater depth than the others to show how the various usability metrics are measured and further refined into estimates of improvements in overall usability.

Home banking. The home-banking system was a prototype of a system to let Danish bank customers access their accounts and other bank services from a home computer with a modem. The interface was explicitly designed to explore the possibilities for such a system with customers who own a personal computer supporting a graphical user interface and a mouse.

The prototype system was developed on a single personal-computer platform under the assumption that alternative versions for other personal computers with graphical user interfaces would use similar designs and have similar usability characteristics. This assumption was not tested. Prototyping aims at generating a running user interface much faster than standard programming. To do this, developers accept some limitations on code generality. One such prototyping limitation was system implementation on a stand-alone personal computer without actual on-line access to bank computers. Instead, the user interface provided access to a limited number of

accounts and other predefined database information stored on the personal computer.

The relevant usability attributes for this application include user task performance, the users' subjective satisfaction, and the number of errors users made. Task performance is less important for this application than for many others, as the users will not be paid employees. Nevertheless, users will be running up telephone bills and taking up resources on the central computer while accessing the system, so task performance is still of interest. Subjective satisfaction may be the most important usability attribute, as usage of a home-banking system is completely discretionary. Moreover, the reputation for being "user friendly" (or rather, "customer friendly") is important for a bank. Minimizing user errors is also important, but most important of all is the avoidance of usage catastrophes, defined as situations where the user does not complete a task correctly. The worst errors are those users make without realizing what they have done, and thus without correcting them.

Each test user was asked to perform specified tasks with the system, accessing some dummy accounts set up for the prototype. In general, it is important to pick benchmark tasks that span the system's expected use, since it would be easy for a design to score well on a single, limited task while being poor for most other tasks. Figure 2 lists the four sets of tasks.

Table 2 shows the measures of task time for each of the four task types measured in user testing of the home-banking system. The best way to calculate a score for task efficiency would be to weigh the measured times for the individual subtasks relative to the frequency with which users would be expected to perform those subtasks in actual system use. To do this perfectly, we need true, contextual field studies of how consumers actually access their bank accounts and other bank information. Unfortunately, it is rarely easy to use current field data to predict frequencies of use for features in a future system, because the system's introduction changes the way the features are used.

For the home-banking example, it is likely that most users would access their own accounts more frequently than they would perform foreign-currency tasks or the special tasks. On the other hand, some small-business users of this kind of system rely extensively on a foreign-exchange feature, so a proper weighting scheme is by no means obvious. In this article, I simply assume that all four tasks are equally important and thus have the same weight. This means that task efficiency is measured by the total task time, which is simply the sum of the times for the four task groups in Table 2.

To measure regular errors and catastrophes, an experimenter counted the number of each, observed as the users performed the specified tasks. The two-question questionnaire shown in Figure 3 measured users' subjective satisfaction after they performed the tasks.

The overall subjective-satisfaction score was computed as the average of numeric scores for the user's replies to these two questions. Table 3 lists the raw data for the subjective-satisfaction and error metrics.

The usability test showed that the first version of the user interface had serious usability problems. Both error rates and catastrophes were particularly high. Even so, users expressed a fairly high degree of subjective satisfaction, perhaps due to the pleasing appearance of graphical user interfaces compared with the types of banking interfaces they were used to.

A major problem was the lack of explicit error messages (the system just beeped when users made errors), which made it hard for users to learn from their mistakes. Another major problem was the lack of a help system. Also, many dialog boxes did not have "cancel" buttons, so users were trapped once they had chosen a command.* Several menu options had names that were dif-

*An anonymous referee of this article asked whether this example was real or contrived: "Would anyone really design a dialog box without a cancel button?" I can confirm that the example is real, and that the designer indeed had forgotten this crucial dialog element in the first design. Not only did this supposedly trivial problem occur in this case study, it occurred again with another designer in another case I am studying. Also, several commercial applications have been released with similar problems.

1. How did you like using the bank system?
   - Very pleasant (1)
   - Somewhat pleasant (2)
   - Neutral (3)
   - Somewhat unpleasant (4)
   - Very unpleasant (5)

2. If you had to perform a task that could be done with this system, would you prefer using the system or would you contact the bank in person?
   - Definitely use the system (1)
   - Likely to use the system (2)
   - Don't know (3)
   - Likely to contact the bank (4)
   - Definitely contact the bank (5)

Figure 3. Questionnaire to measure test users' satisfaction with the home-banking interface.

Table 3. Absolute values of the usability parameters for the home-banking system. Subjective satisfaction was measured on a 1 to 5 scale, where 1 indicated the highest satisfaction.

Version	Subjective Satisfaction (1-5 scale)	Errors Made per User	Catastrophes per User
1	1.92	9.2	2.56
2	1.83	4.3	1.00
3	1.78	2.5	0.44
4	1.86	1.5	0.43
5	1.67	1.5	0.17

Table 4. Normalized improvements in usability parameters for the home-banking system. (Version 1 is the baseline and has zero improvement by definition. All numbers except version numbers are percentages.)

Version	Efficiency (inverse time on task)	Subjective Satisfaction with Dialog	Correct Use (inverse error frequency)	Catastrophe Avoidance	Overall Usability Improvement
1	0	0	0	0	0
2	−18	6	114	156	48
3	12	9	268	582	126
4	12	4	513	495	155
5	26	17	513	1,406	242

ficult to understand and easy to confuse (for example, the two commands "Transfer money" and "Move an amount"). There were several inconsistencies. Users had to type in account numbers as digits only (without any dashes), even though the computer displayed lists of account numbers with dashes separating groups of digits.

To reduce the number of catastrophes, version 2 introduced confirming dialog boxes to let users check the system's interpretation of major commands before execution. For example, one such dialog read, "Each month we will transfer 2,000.00 kroner from your savings account to the Landlord Company, Inc. The first transfer will be made April 30, 1992. OK/Cancel." Also, written error messages in alert boxes replaced the beep, and a special help menu was introduced.

Version 1 had three main menus: Functions, Accounts, and Information. The first menu included commands for ordering account statements, initiating electronic funds transfers, and viewing a list of funds transfers. The second menu included commands for viewing an account statement, transferring money between the user's own accounts, and viewing a list of the user's accounts and their current balances. The third menu included commands for showing current special offers from the bank and accessing foreign-currency exchange rates.

The test showed that users did not understand the conceptual structure of this menu design but instead had to look through all three menus every time they wanted to activate a command. Therefore, the menu structure was completely redesigned for version 2, with only two main menus: Accounts (for all commands operating on the user's accounts) and Other (for all other commands).

The user test of version 2 showed that users had significantly fewer problems with finding commands and made fewer errors. At the same time, the help system was somewhat confusingly structured, and users wasted a fair amount of time reading help information they did not need to complete their current tasks. Table 3 shows version 2 did succeed in its major goal of reducing the very high levels of user errors and catastrophes. However, Table 2 shows this improvement was made at the cost of slower task completion for most tasks. Transfers to others' accounts were faster in version 2 due to the elimination of errors when users typed account numbers the way they were normally formatted.

Version 3 had a completely restructured help system. Also, several of the error messages introduced in version 2 were rewritten to better inform users how to correct their errors.

To further reduce error rates, version 4 identified the user's own accounts by account type (for example, checking account or savings account) instead of just account number. Also, the dialog box for transferring money was expanded with an option to show the available balance for the account where the transfer was to originate. Furthermore, a minor change to the confirmation dialog box for electronic funds transfers prevented an error that occurred fairly frequently with users not entering the correct date for transfers scheduled for future execution. As mentioned above, the confirming dialog normally stated the date on which the transfer was to be made. With version 4, when the transfer was specified to happen immediately, the text was changed to read, "The transfer will take place today." Using the word "today" rather than displaying a date made the message easy to understand and different from the message for future transfers. This iterative design further modified an interface change that had been introduced in version 2.

In the final version, version 5, the designer completely restructured the help system for the second time. The help system introduced in version 3 was structured according to the menu commands in the system and let users get information about any command by selecting it from a list. The revised help system presented a single conceptual diagram of system-supported tasks, linking them with a list of available menu commands and providing further levels of information through a simple hypertext access mechanism.

In addition to the changes outlined here, the designer made many smaller changes to the user interface in each iteration. Few interface elements survived unchanged from version 1 to version 5.

Table 4 shows the normalized values for the four usability metrics as well as the overall usability computed as their geometric mean. Most usability metrics improved for each iteration, but there were also cases where an iteration scored worse than its predecessor on some metric. Many of these lower scores led to further redesigns in later versions. Also, many smaller iterative changes had to be further modified after observations of user testing, even if they were not large enough to cause measurable decreases in the usability metrics.

Cash-register system. The cash-register system was a point-of-sales application for a chain of men's clothing stores in Copenhagen. A computerized cash register with a built-in alphanumeric screen let sales clerks sell specific merchandise with payment received as cash, debit card, credit card, check, foreign currencies converted at the current exchange rate, and gift certificates, as well as combinations of these. Clerks could also accept returned goods, issue gift certificates, and discount prices by a specified percentage.

The usability metrics of interest for

this application included time needed for the users to perform 17 typical tasks, subjective satisfaction, and errors made while completing the tasks. Furthermore, an important metric was the number of times the users requested help from a supervisor while performing the task. This metric was especially important because the users would often be temporary staff taken on for sales or holiday shopping seasons, when the shop would be very busy and such help requests would slow the selling process and inconvenience shoppers. Table 5 shows the values measured for these usability metrics for each of the five versions of the system.

Table 5. Absolute values of usability metrics for the cash-register system. A rating of 1 indicates the best possible subjective satisfaction.

Version	Time on Task (seconds)	Subjective Satisfaction (1-5 scale)	Errors per User During 17 Tasks	Help Requests per User During 17 Tasks
1	2,482	2.38	0.0	2.7
2	2,121	1.58	0.1	1.1
3	2,496	1.62	0.2	1.2
4	2,123	1.45	0.2	0.9
5	2,027	1.69	0.0	0.4

Table 6 shows the normalized improvements in usability for each iteration. Since the error rate was zero for the first version, *any* occurrence of errors would, in principle, constitute an infinite degradation in usability. However, as mentioned earlier, very small error rates (such as one error per 170 tasks, which was the measured error rate for version 2) are not really disastrous for an application like the one studied here. Hence, the table represents a 0.1 increase in error rate as a 10 percent degradation in usability. Another number may have been more reasonable, but it turned out that the error rate for the last version was also zero, so the method chosen to normalize the error rates did not affect the conclusion about overall improvement from the total iterative design process.

Table 6. Normalized improvements in usability metrics for the cash-register system. (Version 1 is the baseline and has zero improvement by definition. All numbers except version numbers are percentages.)

Version	Efficiency (inverse time on task)	Subjective Satisfaction with Dialog	Correct Use (inverse error frequency)	Help (inverse help requests)	Overall Usability Improvement
1	0	0	0	0	0
2	17	61	−10	145	43
3	−1	56	−20	125	29
4	17	77	−20	200	49
5	22	49	0	575	87

The main changes across the iterations were in the wording of the field labels, which were changed several times to make them more understandable. Also, some field labels were made context sensitive to better fit the user's task. Fields not needed for the user's current task temporarily disappeared from the screen. Several shortcuts were introduced to speed common tasks, and these were also changed over subsequent iterations.

Security application. The security application[7] was the sign-on sequence for remote access to a large data-entry and inquiry mainframe application for employees at the branch offices of a major computer company. Users already used their alphanumeric terminals to access an older version of the system. A main goal in the new system development was to ensure that the transition from old to new did not disrupt the branch-office users. Therefore, the most important usability attribute was the users' ability to successfully sign on to the remote mainframe without any errors. Other relevant attributes were the time needed for users to sign on and their subjective satisfaction.

Three usability metrics were measured:

- user performance, as the time needed to complete 12 sign-ons;
- success rate, as the proportion of users who could sign on error-free after the third attempt; and
- users' subjective attitudes toward the system, as the proportion of test users who believed the product was good enough to deliver without any further changes.

Table 7 shows the results for the three versions of the interface with test users who had experience with the old system but not with the new interface. The table shows substantial improvements on all three usability metrics. A further test with an expert user showed that the optimum time to complete a sign-on for the given system was 6 seconds. The test

Table 7. Absolute values of usability metrics for three iterations of a computer security application in a major computer company.[7]

Version	Time to Complete Tasks (minutes)	Success Rate (percent)	Subjective Satisfaction (percent)
1	5.32	20	0
2	2.23	90	60
3	0.65	100	100

Table 8. Normalized improvements in usability metrics for the security application. (Version 1 is the baseline and has zero improvement by definition. All numbers except version numbers are percentages.)

Version	Time to Complete Tasks	Success Rate	Subjective Satisfaction	Usability Improvement
1	0	0	0	0
2	139	350	488	298
3	718	400	2214	882

users were able to complete their 12th sign-on attempt in 7 seconds with the third iteration of the interface, indicating little room for further improvement without changing the system's fundamental nature. Table 8 shows the normalized improvements in usability for the security application over its three versions.

Hypertext system. The hypertext system[12] was an interface designed to facilitate access to large amounts of text in electronic form. Typical applications include technical reference manuals, on-line documentation, and scientific journal articles converted to on-line form. The system ran on a standard workstation using multiple windows and such graphical user-interface features as one that let the user click on a word in the text to find more information about the word.

For use with technical reference texts, the most important usability attributes were the search time for users to find the information they wanted and the search accuracy (the proportion of times they found the correct information). The interface developers measured both attributes by having test users answer 32 questions about a statistics software package using a hypertext version of its reference manual. Table 9 shows the results for the three versions of the hypertext system.

A hypertext system for technical reference texts will obviously have to compete with text presented in traditional printed books. The system developers therefore conducted a comparative test: They asked test users to perform the same tasks with the printed version of the manual.[12] The search time for the printed version was 5.9 minutes with a search accuracy of 64 percent, indicating that the manual was better than the initial version of the hypertext system. This example shows that iterative design should not simply try to improve a system with reference to itself, but also aim at better usability characteristics than the available competition.

Table 10 shows the normalized improvements in usability for the hypertext system. Many of the changes from version 1 to version 2 encouraged users to use a proven search strategy — by making it both easier to use and more explicitly available. Version 3 introduced several additional changes, including an important shortcut that automatically performed a second step that users almost always did after a first step in the initial tests. Both improvements show how iterative design can mold the interface according to user strategies that do not become apparent until after testing has begun. Users always find new and interesting ways to use new computer systems, so it is not enough to rely on preconceived notions to make an interface usable.

The median improvement in overall usability from the first to the last version for the case studies discussed here is 165 percent. A study of 111 pairwise comparisons of user interfaces found that the median difference between two interfaces compared in the research literature was 25 percent.[11] Given that the mean number of iterations in the designs discussed here was three, we might expect the improvement from first to last interface to be around 95 percent, based on an average improvement of 25 percent for each iteration. (Three iterations, each improving usability by 25 percent, would give an improvement from version 1 to version 4 of 95 percent rather than 75 percent, due to a compounding effect similar to that in bank-account interest calculations.) However, the measured improvement was larger, corresponding to an average improvement of 38 percent from one version to the next.

There is no fundamental conflict between the estimate of 25 percent average usability difference between interfaces compared in the research literature and 38 percent difference between versions in iterative design. Picking the better of two proposed interfaces to perform the same task is actually a very primitive usability engineering method that does not let designers combine appropriate features from each design. In contrast, iterative design relies much more on the usability specialist's intelli-

Table 9. Absolute values of usability metrics for the hypertext system.[12] (A version numbering scheme starting with version 0 has been used in other discussions of this system.)

Version	Search Time (minutes)	Search Accuracy (percent)
1	7.6	69
2	5.4	75
3	4.3	78

Table 10. Normalized improvements in usability metrics for the hypertext system. (Version 1 is the baseline and has zero improvement by definition. All numbers except version numbers are percentages.)

Version	Search Time	Search Accuracy	Usability Improvement
1	0	0	0
2	41	9	24
3	77	13	41

gence and expertise throughout the design process. It lets designers combine features from previous versions based on accumulated evidence of what works under what circumstances. Thus, it is not surprising that iterative design might yield a larger improvement between versions than that resulting from simply picking the better of two average design alternatives.

Of course, the 38 percent improvement between versions in iterative design is only a rough estimate based on four case studies. Furthermore, there is a large variability in the magnitude of usability improvement from case to case, so we should not expect a 38 percent improvement in every case. Also, we should not expect to sustain exponential improvements in usability across iterations indefinitely. With many iterations, usability improvements would probably follow a curve somewhat like the one shown in Figure 1, since it is probably not possible to achieve arbitrary improvements in usability simply by iterating sufficiently many times. An open research question is how much usability can be improved and how good an "ultimate user interface" can get, since practical development projects always stop iterating before achieving perfection.

The median improvement from version 1 to version 2 was 45 percent, whereas the median improvement from version 2 to version 3 was "only" 34 percent. In general, we can probably expect the greatest improvements from the first few iterations, as designers discover and remove usability catastrophes. I recommend continuing beyond the initial iteration: Designers sometimes introduce new usability problems in the attempt to fix the old ones. Also, user testing may turn up new interface strategies that need refinement through further iterations.

The projects reviewed in this article include several in which at least one usability-attribute score went down from one version to the next. Also, it was often necessary to redesign a user-interface element more than once to find a usable version. Three versions (two iterations) should be the minimum in an iterative design process. Of course, as in the cash-register case study, the third version may score worse than the second version, so designers cannot always follow a plan to stop after version 3. The actual results of iterative design and testing must guide decisions on how to proceed. ■

Acknowledgments

I collected data on the home-banking and cash-register projects analyzed in this article while on the faculty of the Technical University of Denmark, and these systems do not represent Bellcore products. I thank my students Jens Rasmussen and Frederik Willerup for helping collect data on these projects. I also thank Susan T. Dumais, Michael C. King, and David S. Miller, as well as several anonymous *Computer* referees, for helpful comments on this article.

References

1. K.F. Bury, "The Iterative Development of Usable Computer Interfaces," *Proc. IFIP Interact '84 Int'l Conf. Human-Computer Interaction*, IFIP, Geneva, 1984, pp. 743-748.
2. W. Buxton and R. Sniderman, "Iteration in the Design of the Human-Computer Interface," *Proc. 13th Ann. Meeting Human Factors Assoc. of Canada*, 1980, pp. 72-81.
3. J.D. Gould and C.H. Lewis, "Designing for Usability: Key Principles and What Designers Think," *Comm. ACM*, Vol. 28, No. 3, Mar. 1985, pp. 300-311.
4. L. Tesler, "Enlisting User Help in Software Design," *ACM SIGCHI Bull.*, Vol. 14, No. 3, Jan. 1983, pp. 5-9.
5. J. Nielsen, "The Usability Engineering Life Cycle," *Computer*, Vol. 25, No. 3, Mar. 1992, pp. 12-22.
6. J. Nielsen, *Usability Engineering*, Academic Press, San Diego, Calif., 1993.
7. C.-M. Karat, "Cost-Benefit Analysis of Iterative Usability Testing," *Proc. IFIP Interact '90 Third Int'l Conf. Human-Computer Interaction*, IFIP, Geneva, 1990, pp. 351-356.
8. M. Good et al., "User-Derived Impact Analysis as a Tool for Usability Engineering," *Proc. ACM CHI '86 Conf. Human Factors in Computing Systems*, ACM Press, New York, 1986, pp. 241-246.
9. W.D. Gray et al., "GOMS Meets the Phone Company: Analytic Modeling Applied to Real-World Problems," *Proc. IFIP Interact '90 Third Int'l Conf. Human-Computer Interaction*, IFIP, Geneva, 1990, pp. 29-34.
10. J.M. Carroll and R.L. Campbell, "Softening up Hard Science: Reply to Newell and Card," *Human-Computer Interaction*, Vol. 2, No. 3, 1986, pp. 227-249.
11. J. Nielsen and J. Levy, "Subjective User Preferences versus Objective Interface Performance Measures," to be published in *Comm. ACM*.
12. D.E. Egan et al., "Formative Design-Evaluation of SuperBook," *ACM Trans. Information Systems*, Vol. 7, No. 1, Jan. 1989, pp. 30-57.

Jakob Nielsen is a member of the Computer Sciences Department in Bellcore's applied research area. His research interests include usability engineering, hypertext, and next-generation interaction paradigms, with a particular emphasis on applied methodology.

Nielsen is the author of the books *Hypertext and Hypermedia* and *Usability Engineering*. His previous affiliations include the IBM User Interface Institute in Yorktown Heights, N.Y., and the Technical University of Denmark in Copenhagen. An electronic business card with further information can be retrieved by sending any email message to the server at nielsen-info@bellcore.com.

Readers can contact Nielsen at Bellcore, MRE 2P-370, 445 South Street, Morristown, NJ 07960; e-mail nielsen@bellcore.com.