Iterative User Interface Design

Jakob Nielsen by Jakob Nielsen on 1993-11-01 November 1, 1993

Summary: Redesigning user interfaces on the basis of user testing can substantially improve usability. In four case studies, the median improvement in overall usability was 165% from the first to the last iteration, and the median improvement per iteration was 38%. Iterating through at least three versions of the interface is recommended, since some usability metrics may decrease in some versions if a redesign has focused on improving other parameters.

*Originally published in IEEE Computer **Vol. 26**, No. 11 (November 1993), pp. 32–41.*

Introduction

It has long been recognized [Refs. 1,2,3,4] that user interfaces should be designed iteratively in almost all cases because it is virtually impossible to design a user interface that has no usability problems from the start. Even the best usability experts cannot design perfect user interfaces in a single attempt, so a usability engineering lifecycle should be built around the concept of iteration. [Ref. 5]

Iterative development of user interfaces involves steady refinement of the design based on user testing and other evaluation methods. Typically, one would complete a design and note the problems several test users have using it. These problems would then be fixed in a new iteration which should again be tested to ensure that the "fixes" did indeed solve the problems and to find any new usability problems introduced by the changed design. The design changes from one iteration to the next are normally local to those specific interface elements that caused user difficulties. An iterative design methodology does not involve blindly replacing interface elements with alternative new design ideas. If one has to choose between two or more interface alternatives, it is possible to perform comparative testing to measure which alternative is the most usable, but such tests are usually viewed as constituting a different methodology than iterative design as such, and they may be performed with a focus on measurement instead of the finding of usability problems. Iterative design is specifically aimed at refinement based on lessons learned from previous iterations.

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

After a general discussion of the nature of iteration and usability metrics, this article presents four examples of iterative user interface design and measurement. The first example is discussed in greater depth than the others to provide some insight into how the various usability parameters are measured and further refined.

The Benefits of Iteration

The value of iteration in a usability engineering process is illustrated by a commercial development project analyzed by Karat. [Ref. 7] The user interface to a computer security application was tested and improved through three versions as discussed further below. A lower bound estimate of the value of

the user interface improvements comes from calculating the time saved by users because of the shorter task completion times, thus leaving out any value of the other improvements. The number of users of the security application was 22,876 and each of them could be expected to save 4.67 minutes from using version three rather than version one for their first twelve tasks (corresponding to one day's use of the system), for a total time savings of 1,781 work hours. Karat estimates that this corresponds to saved personnel costs of $41,700 which compares very favorably with the increased development costs of only $20,700 due to the iterative design process. The true savings are likely to be considerably larger since the improved interface was also faster after the first day.

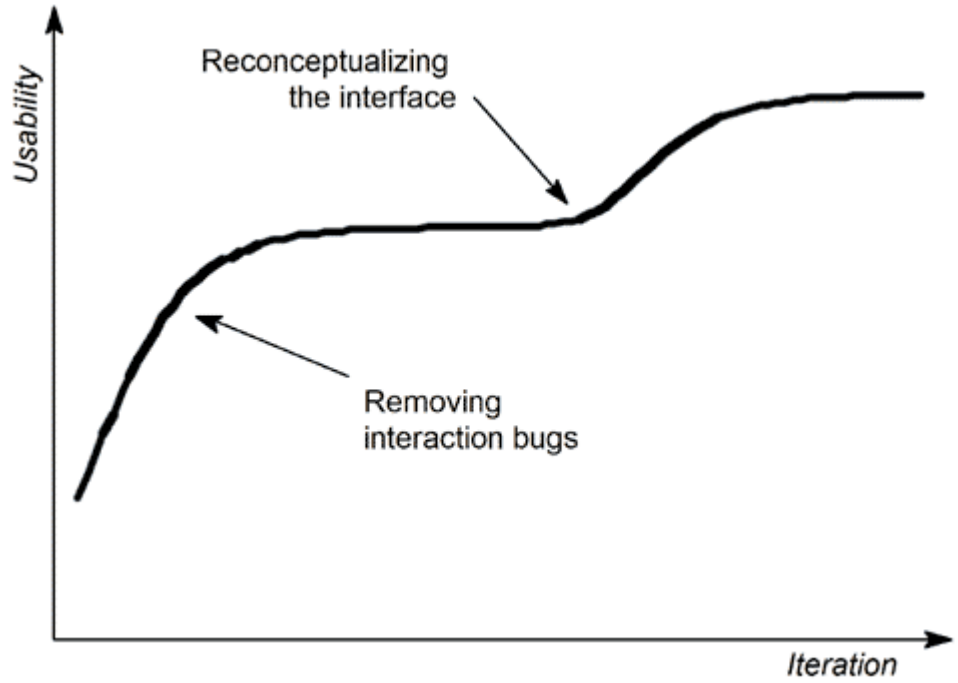

Figure 1. *Interface quality as a function of the number of design iterations: Measured usability will normally go up for each additional iteration, until the design potentially reaches a point where it plateaus.*

Figure 1 shows a conceptual graph of the relation between design iterations and interface usability. Ideally, each iteration would be better than the previous version, but as shown later in this article, this is not always true in practice. Some changes in an interface may turn out not to be improvements after all. Therefore, the true usability curve for a particular product would not be as smooth as the curve in Figure 1. Even so, the general nature of iterative design can probably be expected to be somewhat like Figure 1, even though it must be admitted that we do not yet have sufficiently many documented case studies to estimate the curve precisely. The first few iterations can probably be expected to result in major gains in usability as the true "usability catastrophes" are found and fixed. Later iterations have progressively smaller potential for improvements as the major usability problems are eliminated, and the design may eventually become so polished that very little potential for further improvement would remain. Because of the small number of documented cases, we do not yet know where this point of diminishing returns is in terms of number of iterations. It is not even known for sure whether there is in fact an upper limit on the usability of a given interface or whether it would be possible to improve some interfaces indefinitely with continued substantial gains for each iteration.

Assuming that one stays with the same basic interface and keeps refining it, my personal intuition is that there would be a limit to the level of usability one could achieve. I do believe, however, that one can often break such limits by rethinking the interface and base it on a completely new design as more experience is gained 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 stabile 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 increased usability as indicated by Figure 1 since a fundamental redesign may introduce unexpected usability problems that would have to be ironed out by a few additional iterations.

Unfortunately, interface reconceptualizations in individual development projects are mainly known through anecdotal evidence and 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 allow customers with home computers to search for telephone numbers where search terms could be matched either exactly or phonetically and with the possibility of expanding the search to a larger geographical area. Unfortunately, even after the iterative design had been through 14 versions, test users still indicated that they were intimidated and frustrated by the system. At this stage of the iterative design it was decided to abandon the basic interface design which had been based on a command-line interface. The interface was reconceptualized based on the insight that the command line made the structure of the system invisible to the user. The interface was redesigned around a menu of services displaying all options at all times, thus making them visible and presumably easier to understand. Indeed, user satisfaction on a 1-5 rating scale (where 5="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

The overall quality of a system is actually a sum of many quality attributes, only one of which is usability. Additionally, a system should of course be socially acceptable and be practically feasible with respect to cost, maintainability, etc. Furthermore, the system should fulfill additional requirements such as fitting the user's job needs and allowing users to produce high-quality results since that is the reason for having the system at all. I will not consider these issues here since they are related to *utility* : whether the functionality of the system in principle can do what is needed. This article focuses on usability as the question of *how well* users can use that functionality. Note that the concept of "utility" does not necessarily have to be restricted to work-oriented software. Educational software has high utility if students learn from using it, and an entertainment product has high utility if it is fun to use.

In spite of simplified conceptual illustrations like Figure 1, usability is not really a one-dimensional property of a user interface. Usability has many aspects, and is often associated with the following 5 attributes: [Ref. 6]

- **Easy to learn:** The user can quickly go from not knowing the system to getting some work done with it.
- **Efficient to use:** Once the user has learned the system, a high level of productivity is possible.
- **Easy to remember:** The infrequent user is able to return to using the system after some period of not having used it, without having to learn everything all over.
- **Few errors:** Users do not make many errors during the use of the system, or if they do make errors they can easily recover from them. Also, no catastrophic errors should occur.
- **Pleasant to use:** Users are subjectively satisfied by using the system; they like it.

Focusing in on usability as a quality goal, we thus define it in terms of these five quality attributes. For each of these quality attribute, one then needs to define a metric to numerically characterize the attribute. In the usability field, these metrics are commonly defined in terms of the performance of a

randomly chosen set of representative users while using the system to do a set of representative tasks, and the metrics are thus dependent on these benchmark tasks. Finally, usability measures are empirically derived numbers resulting from the application of one or more metrics to a concrete user interface. This progression from quality goals over quality attributes and their metrics to the actual measures thus makes usability steadily more concrete and operationalized.

From a usability engineering perspective, one should not necessarily pay equal attention to all usability attributes. Some are more important than others and should be the focus of the development efforts throughout the iterative design. In some cases one might even accept lower scores on certain usability metrics as long as their values remain above some minimally acceptable value. [Ref. 8] Each of the case studies discussed below had their own priorities with regard to the different attributes of usability and other projects might have yet other goals. It is an important early part of the usability lifecycle to set priorities for the usability attributes and decide on which ones will deserve the main consideration in a given project.

User efficiency is a measure of how fast users can use the computer system to perform their tasks. Note that user efficiency is a different concept than traditional computer response time, even though faster response times will often allow users to complete their tasks faster. Often, user efficiency is seen as the most important attribute of usability because of its direct bottom-line impact. For example, NYNEX estimates that each one second reduction in work time per call for its toll and assistance operators will save three million U.S. dollars per year. [Ref. 9] Other usability attributes may be critical for certain applications such as control rooms for dangerous 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. There are research projects aiming at analytical methods for estimating certain usability metrics, and they have had some success at dealing with user performance for expert users [Ref. 9] and the transfer of user learning from one system to another. These analytical methods are substantially weaker at estimating error rates and are unable to address the subjective "pleasant to use" dimension of usability. In any case, practical development projects are probably best advised to stay with user testing for the time being since the analytical methods can be rather difficult to apply. [Ref. 10] Often, about ten test users are used for usability measurements, though tests are sometimes conducted with twenty or more users if tight confidence intervals on the measurement results are desired. It is important to pick test users who are as representative as possible of the actual people who will be using the system after its release. When repeated user tests are performed as part of iterative design, one 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 in order 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 the ability of experienced users to return to the system after a period of absence. The basic problem is the need for 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, one would normally need to train users for several weeks or months before their performance plateaued out at the expert level. Such extended test periods are of course infeasible during the design of new products, so expert performance tends to be 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, one should keep in mind that a developer has a much more extensive understanding of the system than a user would have, so one should always test with some real users also.

Quantifying Relative Changes in Usability

In order to compare the different iterative design projects later in this article, a uniform way of scoring usability improvements is desirable. The approach used here is to normalize all usability measures with respect to the values measured for the initial design. Thus, the initial design is by definition considered to have a normalized usability value of 100, and the subsequent iterations are normalized by dividing their measured values by those measured for the initial design. For example, if users made four errors while completing a set task with version one and three errors with version two of an interface, the normalized usability of interface two with respect to user errors was 133% of the usability of interface one, indicating that a 33% improvement in usability had been achieved. In just the same way, the normalized usability of interface two would still be 133% if users made eight errors with interface one and six errors with interface two.

A particular problem is how to convert measures of subjective satisfaction into improvement scores, since subjective satisfaction is often measured with questionnaires and rating scales. It would not be reasonable just to take the ratio of the raw rating scale scores. Doing so would mean, say, that a system rated 2 on a 1-5 scale would be twice as good as a system rated 1 on a 1-5 scale, but also that a system rated 2 on a 1-7 scale would be twice as good as a system rated 1 on a 1-7 scale, even though it is clearly easier to achieve a rating of 2 on a 1-7 scale than on a 1-5 scale.

The underlying problem is that the rating scales are not ratio scales, so it is theoretically impossible to arrive at a perfect formula to calculate how much better one rating is relative to another. Assuming that the goal of usability engineering is purely economical, it might be possible to gather empirical data to estimate how various subjective satisfaction ratings translate to increased sales of a product or increased employee happiness (and one could then further measure the contribution to a company's profits from having happier employees). I am not aware of any such data, so I proceeded with calculating relative subjective satisfaction scores as well as possible, using a method that had proven successful in a larger study of subjective preference measures, [Ref. 11] even though the resulting formula is admittedly not perfect. This transformation method is discussed further in the box.

The case studies in this article all used a given set of test tasks to measure usability throughout the iterative design process. This was possible because they had their basic functionality defined before the start of the projects. In contrast, some development projects follow a more exploratory model where the tasks to be performed with a system may change during the development process as more is learned about the users and their needs. For such projects, it may be more difficult to use a measurement method like the one outlined here, but the general results with respect to the improvements from iterative design should still hold.

Transforming Subjective Ratings to a Ratio Scale

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

1. The various rating scales were linearly rescaled to map onto the interval from -1 to +1 with zero as the midpoint and +1 as the best rating. This transformation provided a uniform treatment of scales independent of their original scale intervals.

2. The *arcsine* function was applied to the rescaled scores. This transformation was intended to compensate for the fact that rating scales are terminated at each end, making it harder for an average rating to get close to one of the ends of the scales. For example, an average rating of 4.6 on a 1-5 scale might result from four users rating the system 4 and six users rating it 5.0. Now, even if all

ten users like some other system better and would like to increase their rating by one point, the second system would only get an average rating of 5.0 (an increase of 0.4 instead of 1.0 as "deserved") because six of the users were already at the end of the scale. Because of this phenomenon, changes in ratings towards the end of a rating scale should be given extra weight, which is achieved by the way the *arcsine* function stretches the ends of the interval.

3. The exponential function e^x was applied to the stretched scores to achieve numbers for which ratios were meaningful. Without such a transformation, it would seem that, say, a score of 0.2 was twice as good as a score of 0.1, whereas a score of 0.7 would only be 17% 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% better than a score of 0.1, and a score of 0.7 is also 10.5% better than a score of 0.6. Admittedly, the choice of functions for these transformations is somewhat arbitrary. For example, an exponential function with another basis than *e* would have achieved the same effect while resulting in different numbers. Remember, however, that the purpose of the present analysis is to compare *relative* ratings of the various interface iterations, and that the same transformations were applied to each of the original ratings. Nowhere do we use the value of a single transformed rating; we always look at the ratio between two equally transformed ratings, and all the transformations are of course monotonic. Also, an analysis of a larger body of subjective satisfaction data indicated that the use of these transformations resulted in reasonable statistical characteristics. [Ref. 11]

Calculating Scores for Overall Usability

It is often desirable to have a single measure of overall usability of a user interface. Typically, interface designers can choose between several alternative solutions for various design issues and would like to know how each alternative impacts the overall usability of the resulting system. As another example, a company wanting to choose a single spreadsheet as its corporate standard would need to know which of the many available spreadsheets was the most usable. In both examples, usability would then need to be traded off against other considerations such as implementation time for the design alternatives and purchase price for the spreadsheets.

Ultimately, the measure of usability should probably be monetary in order 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 work day. Loaded salaries include not just the money paid out to employees but also any additional costs of having them employed, such as social security taxes and medical and pension benefits. For example, I recently performed a usability evaluation of a telephone company application. Improvements in the user interface based on this study were estimated to result in a reduction in training time of half a day per user and a speedup in expert user performance of 10%. Given the number of users and their loaded salaries, these numbers translate 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. User error rates can in principle be combined with measures of the seriousness of each error and estimates of the impact on corporate profits from each error. 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 would have a cost approximately corresponding to the time needed for the user to discover the error and to walk to the wrong printer to retrieve the output. An error causing a cement plant to burn down would have a cost corresponding to the value of the plant plus any business lost while it was being rebuilt. In other cases, error costs are harder to estimate. For example, an error causing a customer to be

shipped a different product from that ordered would have direct costs corresponding to the administrative costs of handling the return plus the wasted freight charges. However, there could also very well be much larger indirect costs if the company got a reputation of being an unreliable supplier causing customers to prefer doing business elsewhere.

Finally, subjective satisfaction is perhaps the usability attribute that has 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 shrinkwrap 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 to make and were not available in the case studies discussed below.

Due to the difficulties of using monetary measures of all usability attributes, an alternative approach has been chosen here. Overall usability is calculated in relative terms as the geometric mean (the *n* 'th 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. Second, there is no use of cut-off values where any further improvement in a given usability metric would be of no practical importance. For example, once error rates have been reduced to, say, one error per ten thousand user transactions, it may make no practical difference to reduce them further. Even so, reducing the error rate to, say, one error per twenty thousand transactions would double the usability score for the user error metric as long as no cut-off 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 latter problem as a geometric mean gives comparatively less weight to uncommonly large numbers. The geometric mean increases more by improving all the usability metrics a little than by improving a single metric a lot and leaving the others stagnant.

Case Studies in Iterative User Interface Design

The following sections present data from the four case studies in iterative user interface design summarized in Table 1. The first example is described in somewhat greater depth than the others in order to provide some insight into how the various usability metrics are measured and further refined into estimates of improvements in overall usability.

Name of System	Interface Technology	Versions Tested	Subjects per Test	Overall Improvement
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 interface	3	9	882%
Hypertext	Workstation graphical user interface	3	10	41%

Table 1. ***Four case studies of iterative design.***

Home Banking

The home banking system was a prototype of a system to allow Danish bank customers access to their accounts and other bank services from a home computer using a modem. The interface was explicitly designed to explore the possibilities for such a system when customers owned 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, however. Prototyping aims at generating a running user interface much faster than standard programming by accepting some limitations on the generality of the code. As one such prototyping limitation, the system was implemented on a stand-alone personal computer without actual on-line access to bank computers. Instead, the user interface only provided access to a limited number of accounts and other pre-defined 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 made by users. Task performance is less important for this application than for many others, as the users will not be paid employees. Even so, 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 as a usability attribute. Subjective satisfaction may be the most important usability attribute, as usage of a home banking system is completely discretionary, and since the reputation for being "user friendly" (or rather, "customer friendly") is an important competitive parameter for a bank. User errors are also important, but most important of all is probably the avoidance of usage catastrophes, defined as situations where the user does not complete a task correctly. User errors are certainly unpleasant and should be minimized, but it is considerably worse when a user makes an error without realizing it, and thus without correcting it. Each test user was asked to perform a specified set of tasks with the system, accessing some of the dummy accounts that had been set up for the prototype. In general, it is important to pick benchmark tasks that span the expected use of the system since it would be easy for a design to score well on a single, limited task while being poor for most other tasks.

Four sets of tasks were used:

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.
   - 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.

2. 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.

3. Special tasks

- Given the electronic funds transfers you have set up, what will the balance be on August 12th? (Do not calculate this manually; find out through the system).
- You returned the stereo set to the shop, so cancel the remaining installment payments.

Table 2 shows the measures of task time for each of the four task types measured in the user testing of the home banking system. The best way of calculating a score for task efficiency would be to weigh the measured times for the individual sub-tasks relative to the expected frequency with which users would want to perform those sub-tasks in field use of the system. To do this perfectly, it would be necessary to perform true, contextual field studies of how consumers actually access their bank account 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 very introduction of the system changes the way the features are used.

Version	Basic Tasks (own account)	Transfers to Others' Accounts	Foreign Exchange	Special Tasks	Total Task Time
1	199	595	245	182	**1,220**
2	404	442	296	339	**1,480**
3	221	329	233	317	**1,090**
4	206	344	225	313	**1,088**
5	206	323	231	208	**967**

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

Color-coding indicates whether a redesign led to an improvement or a degradation in a usability metric: green cells show when a design was better than the previous iteration, and rose cells show designs that are worse than the previous iteration.

For the home banking example, it is likely that most users would access their own account 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 have been known to rely extensively on a foreign exchange feature, so a proper weighting scheme is by no means obvious. In this article, I will simply assume that all four tasks are equally important and should 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.

Regular errors and catastrophes were measured by having an experimenter count the number of each, observed as the users were performing the specified tasks. Subjective satisfaction was measured by giving the test users a two-question questionnaire after they had performed the tasks with the system:

How did you like using the bank system?

1. Very pleasant
2. Somewhat pleasant
3. Neutral
4. Somewhat unpleasant
5. Very unpleasant

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?

1. Definitely use the system
2. Likely use the system
3. Don't know
4. Likely contact the bank
5. Definitely contact the bank

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

Version	Subjective Satisfaction (1-5 scale; 1 best)	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	2.5	0.43
5	1.67	1.5	0.17

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. Color-coding indicates whether a redesign led to an improvement or a degradation in a usability metric: green cells show when a design was better than the previous iteration, and rose cells show designs that are worse than the previous iteration.***

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 which might have been due to the pleasing appearance of graphical user interfaces compared with the types of banking interfaces they were used to. One major problem was the lack of explicit error messages (the system just beeped when the users made an error), which made it hard for the 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. Furthermore, several menu options had names that were difficult to understand and were easy to confuse (for example the two commands "Transfer money" and "Move an amount"), and there were several inconsistencies. For example, users had to type in account numbers as digits only (without any dashes), even though dashes were used to separate groups of digits in listings of account numbers shown by the computer.

(One of the anonymous referees of this article questioned 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 dialogue element in the first design. Not only did this supposedly trivial problem occur for real in this case study, it occurred again for another designer in another design case I am currently studying. Also, several commercial applications have been released with similar problems.)

To reduce the number of catastrophes, confirming dialog boxes were introduced in version two to allow the user to check the system's interpretation of major commands before they were executed. 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 one had used three main menus: Functions, Accounts, and Information. The first menu included commands for ordering account statements, for initiating electronic funds transfers, and for viewing a list of funds transfers. The second menu included commands for viewing an account statement, for transferring money between the user's own accounts, and for viewing a list of the user's accounts and their current balance. The third menu included commands for showing current special offers from the bank and for accessing foreign currency exchange rates. The test had shown that the 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 two, 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 two showed that users had significantly fewer problems with finding commands in the menus and that they also 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 which they did not need to complete their current tasks. As can be seen from Table 3, version two did succeed in its major goal of reducing the very high levels of user errors and catastrophes from version one, but as shown by Table 2, this improvement was made at the cost of slower task completion for most tasks. Transfers to others' accounts were faster in version two due to the elimination of errors when users typed account numbers the way they were normally formatted.

For version three, the help system was completely restructured. Also, several of the error messages introduced in version two were rewritten to make them constructive and better inform users how they can correct their errors.

To further reduce the error rates, version four identified the user's own accounts by their account type (e.g. checking account or savings account instead of just their account number). Also, the dialog box for transferring money was expanded with an option to show the available balance on the account from which the transfer was to be made. Furthermore, a minor change was made to the confirmation dialog box for electronic funds transfers to prevent a specific error that had been observed to occur fairly frequently with users not entering the correct date for transfers that were to be scheduled for future execution. As mentioned above, the confirming dialog normally stated the date on which the transfer was to be made, but 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 listing a date made the message more concrete and easy to understand and also made it different from the message used for future transfers. Note how this iterative design further modified an interface change that had been introduced in version two.

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

In addition to the changes outlined here, a large number of smaller changes were made to the user interface in each iteration. Almost no user interface elements survived unchanged from version one to version five.
Table 4 finally shows the normalized values for the four usability metrics as well the overall usability computed as their geometric mean. It can be seen that most usability metrics improved for each iteration but that there also were cases where an iteration scored worse than its predecessor on some metric. As mentioned, many of these lower scores led to further redesigns in later versions. Also, there were many smaller instances of iterative changes that had to be further modified due to observations made during the user testing, even if they were not large enough to cause measurable decreases in the usability metrics.

Version	Efficiency (inverse task time)	Subjective Satisfaction	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%	+595%	**+155%**
5	+26%	+17%	+513%	+1,406%	**+242%**

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, relative to the baseline metrics for version 1.)***
Color-coding indicates whether a redesign led to an improvement or a degradation in a usability metric: green cells show when a design was better than the previous iteration, and rose cells show designs that are worse than the previous iteration.

A Cash Register System
The cash register system was a point of sales application for a chain of men's clothes stores in Copenhagen. A computerized cash register with a built-in alphanumeric screen allowed sales clerks to perform tasks like selling specific pieces of merchandise with payment received as cash, debit card, credit card, check, foreign currencies converted according to the current exchange rate, and gift certificates, as well as combinations of these, and also accepting returned goods, issuing gift certificates, and discounting prices by a specified percentage.

The usability metrics of interest for this application again 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 last metric was especially important because the users would often be temporary sales staff taken on for sales or holiday shopping seasons where the shop would be very busy and such help requests would slow down the selling process and inconvenience shoppers. The values measured for these usability metrics for each of the five versions of the system are shown in Table 5.

Version	Time on Task (seconds)	Subjective Satisfaction (1-5 scale; 1 best)	Errors Made 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 5. ***Absolute values of usability metrics for the cash-register system. Subjective satisfaction was measured on a 1 to 5 scale, where 1 indicated the highest satisfaction. Color-coding indicates whether a redesign led to an improvement or a degradation in a usability metric: green cells show when a design was better than the previous iteration, and rose cells show designs that are worse than the previous iteration.***

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 above, very small error rates (such as one error per 170 tasks, which was the measured error rate for version two) are not really disastrous for an application like the one studied here, so a 0.1 increase in error rate has been represented in the table as a 10% degradation in usability. One can obviously argue whether another number would have been more reasonable, but as it turns out, the error rate for the last version was also zero, so the method chosen to normalize the error rates does not impact on the conclusion with respect to overall improvement due to the total iterative design process.

Version	Efficiency (inverse task time)	Subjective Satisfaction	Correct Use (inverse error frequency)	Help Avoidance (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%**

Table 6. ***Normalized improvements in usability parameters for the cash-register system. (Version 1 is the baseline and has zero improvement by definition. All numbers except version numbers are percentages, relative to the baseline metrics for version 1.) Color-coding indicates whether a redesign led to an improvement or a degradation in a usability metric: green cells show when a design was better than the previous iteration, and rose cells show designs that are worse than the previous iteration.***

The main changes across the iterations regarded the wording of the field labels that were changed several times to make them more understandable. Also, some field labels were made context sensitive in order to improve their fit with the user's task, and fields that were not needed for given tasks were removed from the screen when users were performing those tasks. Several shortcuts were introduced to speed up common tasks, and these shortcuts also had to be modified over subsequent iterations.

A Security Application

The security application [Ref. 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, and a main goal of the development of the new system was to ensure that the transition between the two systems happened without disrupting the users in the branch offices. Therefore, the most important usability attribute was the users' ability to successfully use the new security application to 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 was measured as time needed to complete twelve sign-ons on the system, success rate was measured as proportion of users who could sign on error free after the third attempt, and the users' subjective attitudes towards the system was measured as the proportion of test users who believed that the product was good enough to deliver without any further changes. Table 7 shows the result of measuring these metrics for the three versions of the interface using test users who had experience with the old system but not with the new interface.

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

Table 7. ***Absolute values of usability metrics for three iterations of a computer security application in a major computer company.***
Color-coding indicates whether a redesign led to an improvement or a degradation in a usability metric: green cells show when a design was better than the previous iteration, and rose cells show designs that are worse than the previous iteration.

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 users were able to complete their twelfth sign-on attempt in 7 seconds with the third iteration of the interface, indicating that there would not be much room for further improvement without changing the fundamental nature of the system. Table 8 shows the normalized improvements in usability for the security application over its three versions.

Version	Efficiency (inverse task time)	Success Rate	Subjective Satisfaction	Overall Usability Improvement
1	0	0	0	**0**

2	+139%	+350%	+448%	**+298%**
3	+718%	+400%	+2,214%	**+882%**

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, relative to the baseline metrics for version 1.)***
Color-coding indicates whether a redesign led to an improvement or a degradation in a usability metric: green cells show when a design was better than the previous iteration, and rose cells show designs that are worse than the previous iteration.

A Hypertext System
The hypertext system [Ref. 12] was an interface designed to facilitate access to large amounts of text in electronic form. Typical applications include technical reference manuals, online documentation, and scientific journal articles converted into online form. The system ran on a standard workstation using multiple windows and various other graphical user interface features such as the ability for the user to click on a word in the text with the mouse to find more information about that word.
For the use of this hypertext system for technical reference texts, the most important usability attributes were the search time for users to find information about their problems as well as the search accuracy (the proportion of times they found the correct information). Both of these attributes were measures by having test users answer 32 given questions about a statistics software package on the basis of a hypertext version of its reference manual. The results of these tests for the three versions of the hypertext system are shown in Table 9.

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

Table 9. ***Absolute values of usability metrics for the hypertext system. (A version numbering scheme starting with version 0 has been used in other papers about this system, but for internal consistency in this document, I gave the first version the number 1 in this table.).***
Color-coding indicates whether a redesign led to an improvement or a degradation in a usability metric: green cells show when a design was better than the previous iteration, and rose cells show designs that are worse than the previous iteration.

A hypertext system for technical reference texts will obviously have to compete with text presented in traditional printed books. The developers of the hypertext system therefore conducted a comparative test where a group of test users were asked to perform the same tasks with the printed version of the manual. [Ref. 12] The search time for paper was 5.9 minutes with a search accuracy of 64%, indicating that paper actually was better than the initial version of the hypertext system. This example clearly shows that iterative design should not just try to improve a system with reference to itself, but should also aim at arriving 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 were based on encouraging users to use a search strategy that had proven effective by making it both easier to use and more explicitly available. For version 3, several additional changes were made, including an important shortcut that automatically

performed a second step that users almost always did after a certain first step in the initial tests. Both improvements illustrate the important ability of iterative design to 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 make an interface usable according to preconceived notions about how they will be used.

Version	Search Efficiency (inverse task time)	Search Accuracy	Overall Usability Improvement
1	0	0	**0**
2	+41%	+9%	**+24%**
3	+77%	+13%	**+41%**

Table 10. ***Normalized improvements in usability parameters for the hypertext system. (Version 1 is the baseline and has zero improvement by definition. All numbers except version numbers are percentages, relative to the baseline metrics for version 1.)***
Color-coding indicates whether a redesign led to an improvement or a degradation in a usability metric: green cells show when a design was better than the previous iteration, and rose cells show designs that are worse than the previous iteration.

Conclusions

The median improvement in overall usability from the first to the last version for the case studies discussed here was 165%. A study of 111 pairwise comparisons of user interfaces found that the median difference between two interfaces that had been compared in the research literature was 25%. [Ref. 11] Given that the mean number of iterations in the designs discussed here was three, one might expect the improvement from first to last interface to be around 95% based on an average improvement of 25% for each iteration. As mentioned, the measured improvement was larger, corresponding to an average improvement of 38% from one version to the next.

(Three iterations, each improving usability by 25%, lead to an improvement from version one to version four of 95% rather than 75%, due to a compounding effect similar to that observed when calculating interest on a bank account across several years.)

There is no fundamental conflict between the estimate of 25% average usability difference between interfaces compared in the research literature and 38% difference between versions in iterative design. Picking the best of two proposed interfaces to perform the same task is actually a very primitive usability engineering method that does not allow one to combine appropriate features from each of the designs. In contrast, iterative design relies much more on the intelligence and expertise of the usability specialist throughout the design process and allows designers to combine features from previous versions based on accumulated evidence of what works under what circumstances. It is thus not surprising that iterative design might lead to larger improvements between versions than that found simply by picking the best of two average design alternatives.
Of course, the 38% improvement between versions in iterative design should only be considered a rough estimate as it was only based on four case studies. Furthermore, there is a large variability in the magnitude of usability improvement from case to case, so one should not expect to realize 38% improvements in all cases. Finally, one should not expect to be able to sustain exponential improvements in usability across iterations indefinitely. If a very large number of iterations were to be used, usability improvements would probably follow a curve somewhat like Figure 1, since it is probably not possible to achieve arbitrary improvements in usability just by iterating sufficiently many times. It is currently an open research question how much usability can be improved and how

good an "ultimate user interface" can get, since practical development projects always stop iterating before perfection is achieved.

The median improvement from version 1 to version 2 was 45%, whereas the median improvement from version 2 to version 3 was "only" 34%. In general, one can probably expect the greatest improvements from the first few iterations as usability catastrophes get discovered and removed. It is recommended to continue beyond the initial iteration, however, since one sometimes introduces new usability problems in the attempt to fix the old ones. Also, user testing may turn up new interface strategies that would have to be refined through further iterations.
The projects discussed in this article included several cases where at least one usability attribute had reduced scores from one version to the next. Also, it was often necessary to redesign a given user interface element more than once before a usable version had been found. Three versions (two iterations) should probably be the minimum in an iterative design process. Of course, as in our cash register case study, the third version will sometimes test out as worse than the second version, so one cannot always follow a plan to stop after version three. The actual results of the iterative design and testing must guide the decisions on how to proceed in each individual project.

Update added 2006:
By way of comparison, across a large number of much newer Web design projects I studied, the average improvement in measured usability in a redesign was 135%.
See also my paper on parallel design for a way to kickstart an iterative design project by testing multiple alternative design suggestions at the same time during the first iteration.

Acknowledgments
This paper was written while the author was a member of the Applied Research Division of Bell Communications Research (Bellcore). Data on the home banking and cash register projects analyzed in this article was collected while the author was on the faculty of the Technical University of Denmark and these systems do not represent Bellcore products. The author would like to thank his students, Jens Rasmussen and Frederik Willerup, for helping collect data on these projects. The author would like to 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. Bury, K.F. The iterative development of usable computer interfaces. In Proceedings of *IFIP INTERACT'84 International Conference on Human-Computer Interaction* (London, U.K., 4-7 September 1984), 743-748.
2. Buxton, W., and Sniderman, R. Iteration in the design of the human-computer interface. In *Proceedings of the 13th Annual Meeting of the Human Factors Association of Canada*, 1980, pp. 72-81.
3. Gould, J.D., and Lewis, C.H. Designing for usability: Key principles and what designers think. *Communications of the ACM* **28**, 3 (March 1985), 300-311.
4. Tesler, L. Enlisting user help in software design. *ACM SIGCHI Bulletin* **14**, 3 (January 1983), 5-9.
5. Nielsen, J. The usability engineering life cycle. *IEEE Computer* **25**, 3 (March 1992), 12-22.
6. Nielsen, J. *Usability Engineering*. Academic Press, San Diego, CA, 1993.
7. Karat, C.-M. Cost-benefit analysis of iterative usability testing. In *Proceedings of IFIP INTERACT'90 Third International Conference on Human-Computer Interaction* (Cambridge, U.K., 27-31 August 1990), pp. 351-356.
8. Good, M., Spine, T.M., Whiteside, J., and George, P. User-derived impact analysis as a tool for usability engineering. In *Proceedings of ACM CHI'86 Conference on Human Factors in Computing Systems* (Boston, MA, 13-17 April 1986), pp. 241-246.

9. Gray, W.D., John, B.E., Stuart, R., Lawrence, D., and Atwood, M.E. GOMS meets the phone company: Analytic modeling applied to real-world problems. In *Proceedings of IFIP INTERACT'90 Third International Conference on Human-Computer Interaction* (Cambridge, U.K., 27-31 August 1990), pp. 29-34.
10. Carroll, J.M., and Campbell, R.L. Softening up hard science: Reply to Newell and Card. *Human-Computer Interaction* **2**, 3 (1986), 227-249.
11. Nielsen, J., and Levy, J. Measuring usability - preference vs. performance. *Communications of the ACM* **37**, 4 (April 1994), 66-75.
12. Egan, D.E., Remde, J.R., Gomez, L.M., Landauer, T.K., Eberhardt, J., and Lochbaum, C.C. Formative design-evaluation of SuperBook. *ACM Transactions on Information Systems* **7**, 1 (January 1989), 30-57.