Guerrilla HCI: Using Discount Usability Engineering to Penetrate the Intimidation Barrier

by Jakob Nielsen, 1994

This paper was one of the chapters in the book Cost-Justifying Usability (edited by Randolph G. Bias and Deborah J. Mayhew).

One of the oldest jokes in computer science goes as follows:

> **Q:** ***How many programmers does it take to change a light bulb?***
> **A:** ***None; it is a hardware problem!***

When asking how many usability specialists it takes to change a light bulb, the answer might well be four: Two to conduct a field study and task analysis to determine whether people really need light, one to observe the user who actually screws in the light bulb, and one to control the video camera filming the event. It is certainly true that one should study user needs before implementing supposed solutions to those problems. Even so, the perception that anybody touching usability will come down with a bad case of budget overruns is keeping many software projects from achieving the level of usability their users deserve.

1 The Intimidation Barrier

It is well known that people rarely use the recommended usability engineering methods [Nielsen 1993; Whiteside et al. 1988] on software development projects in real life. This includes even such basic usability engineering techniques as early focus on the user, empirical measurement, and iterative design which are used by very few companies. Gould and Lewis [1985] found that only 16% of developers mentioned all three principles when asked what one should do when developing and evaluating a new computer system for end users. Twenty-six percent of developers did not mention a single of these extremely basic principles. A more recent study found that only 21% of Danish software developers knew about the thinking aloud method and that only 6% actually used it [Milsted et al. 1989]. More advanced usability methods were not used at all.

One important reason usability engineering is not used in practice is the cost of using the techniques. Or rather, the reason is the perceived cost of using these techniques, as this chapter will show that many usability techniques can be used quite cheaply. It should be no surprise, however, that practitioners view usability methods as expensive considering, for example, that a paper in the widely read and very respected journal Communications of the ACM estimated that the "costs required to add human factors elements to the development of software" was $128,330 [Mantei and Teorey 1988]. This sum is several times the total budget for usability in most smaller companies, and one interface

evangelist has actually found it necessary to warn such small companies against believing the CACM estimate [Tognazzini 1990]. Otherwise, the result could easily be that a project manager would discard any attempt at usability engineering in the belief that the project's budget could not bear the cost. Table 1 shows the result of adjusting a usability budget according to the discount usability engineering method discussed below. The numbers in Table 1 are for a medium scale software project (about 32,000 lines of code). For small projects, even cheaper methods can be used, while really large projects might consider additional funds to usability and the full-blown traditional methodology, though even large projects can benefit considerably from using discount usability engineering.

Original usability cost estimate by [Mantei and Teorey 1988]	$128,330
Scenario developed as paper mockup instead of on videotape	- $2,160
Prototyping done with free hypertext package	- $16,000
All user testing done with 3 subjects instead of 5	- $11,520
Thinking aloud studies analyzed by taking notes instead of by video taping	- $5,520
Special video laboratory not needed	- $17,600
Only 2 focus groups instead of 3 for market research	- $2,000
Only 1 focus group instead of 3 for accept analysis	- $4,000
Questionnaires only used in feedback phase, not after prototype testing	- $7,200
Usability expert brought in for heuristic evaluation	+ $3,000
Cost for "discount usability engineering" project	$65,330

Table 1
Cost savings in a medium scale software project by using the discount usability engineering method instead of the more thorough usability methods sometimes recommended.

British studies [Bellotti 1988] indicate that many developers don't use usability engineering because HCI (human-computer interaction) methods are seen as too time consuming and expensive and because the techniques are often intimidating in their complexity. The "discount usability engineering" approach is intended to address these two issues. Further reasons given by Bellotti were that there were sometimes no perceived need for HCI and a lack of awareness about appropriate techniques. These two other problems must be addressed by education [Perlman 1988, 1990; Nielsen and Molich 1989] and propaganda [Nielsen 1990a], but even for that purpose, simpler usability methods should help. Also, time itself is on the side of increasing the perceived need for HCI since the software market seems to be shifting away from the "features war" of earlier years [Telles 1990]. Now, most software products have more features than users will ever need or learn, and Telles [1990] states that the "interface has become an important element in garnering good reviews" of software in the trade press.

As an example of "intimidating complexity," consider the paper by Karwowski et al. [1989] on extending the GOMS model [Card et al. 1983] with fuzzy logic. Note that I am not complaining that doing so is bad research. On the contrary, I find it very exciting to develop methods to extend models like GOMS to deal better with real-world circumstances like uncertainty and user errors. Unfortunately, the fuzzy logic GOMS and similar work can easily lead to intimidation when software people without in-depth knowledge of the HCI field read the papers. These readers may well believe that such methods represent "the way" to do usability engineering even though usability specialists

would know that the research represents exploratory probes to extend the field and should only serve as, say, the fifth or so method one would use on a project. There are many simpler methods one should use first [Nielsen 1992a, 1993].

I certainly can be guilty of intimidating behavior too. For example, together with Marco Bergman, I recently completed a research project on iterative design where we employed a total of 99 subjects to test various versions of a user interface at a total estimated cost of $62,786. People reading papers reporting on this and similar studies might be excused if they think that iterative design and user testing are expensive and overly elaborate procedures. In fact, of course, it is possible to use considerably fewer subjects and get by with much cheaper methods, and we took care to say so explicitly in our paper. A basic problem is that with a few exceptions, published descriptions of usability work normally describe cases where considerable extra efforts were expended on deriving publication-quality results, even though most development needs can be met in much simpler ways.

As one example, consider the issue of statistical significance. I recently had a meeting to discuss usability engineering with the head of computer science for one of the world's most famous laboratories, and when discussing the needed number of subjects for various tests, he immediately referred to the need for test results to be statistically significant to be worth collecting. Certainly, for much research, you need to have a high degree of confidence that your claimed findings are not just due to chance. For the development of usable interfaces, however, one can often be satisfied by less rigorous tests.

Statistical significance is basically an indication of the probability that one is not making the wrong conclusion (e.g., a claim that a certain result is significant at the $p<.05$ level indicates that there is a 5% probability that it is false). Consider the problem of choosing between two alternative interface designs [Landauer 1988]. If no information is available, you might as well choose by tossing a coin, and you will have a 50% probability of choosing the best interface. If a small amount of user testing has been done, you may find that interface A is better than interface B at the 20% level of significance. Even though 20% is considered "not significant," your tests have actually improved your chance of choosing the best interface from 50/50 to 4-to-1, meaning that you would be foolish not to take the data into account when choosing. Furthermore, even though there remains a 20% probability that interface A is not better than interface B, it is very unlikely that it would be much worse than interface B. Most of the 20% accounts for cases where the two interfaces are equal or where B is slightly better than A, meaning that it would almost never be a really bad decision to choose interface A. In other words, even tests that are not statistically significant are well worth doing since they will improve the quality of decisions substantially.

2 The Discount Usability Engineering Approach

Usability specialists will often propose using the best possible methodology. Indeed, this is what they have been trained to do in most universities. Unfortunately, it seems that "le mieux est l'ennemi du bien" (the best is the enemy of the good) [Voltaire 1764] to the extent that insisting on using only the best methods may result in having no methods used at all. Therefore, I will focus on achieving "the good" with respect to having some usability engineering work performed, even though the methods needed to achieve this result are definitely not "the best" method and will not give perfect results.

It will be easy for the knowledgable reader to put down the methods proposed here with various well-known counter-examples showing important usability aspects that will be missed under certain circumstances. Some of these counter-examples are no doubt true and I do agree that better results can be achieved by applying more careful methodologies. But remember that such more careful methods are also more expensive -- often in terms of money, and always in terms of required expertise (leading to the intimidation factor discussed above). Therefore, the simpler methods stand a much better chance of actually being used in practical design situations and they should therefore be viewed as a way of serving the user community.

The "discount usability engineering" [Nielsen 1989b, 1990a, 1993] method is based on the use of the following three techniques:

- **Scenarios**
- **Simplified thinking aloud**
- **Heuristic evaluation**

Additionally, the basic principle of early focus on users should of course be followed. It can be achieved in various ways, including simple visits to customer locations.

2.1 Scenarios

Scenarios are a special kind of prototyping as shown in Figure 1. The entire idea behind prototyping is to cut down on the complexity of implementation by eliminating parts of the full system. Horizontal prototypes reduce the level of functionality and result in a user interface surface layer, while vertical prototypes reduce the number of features and implement the full functionality of those chosen (i.e. we get a part of the system to play with).

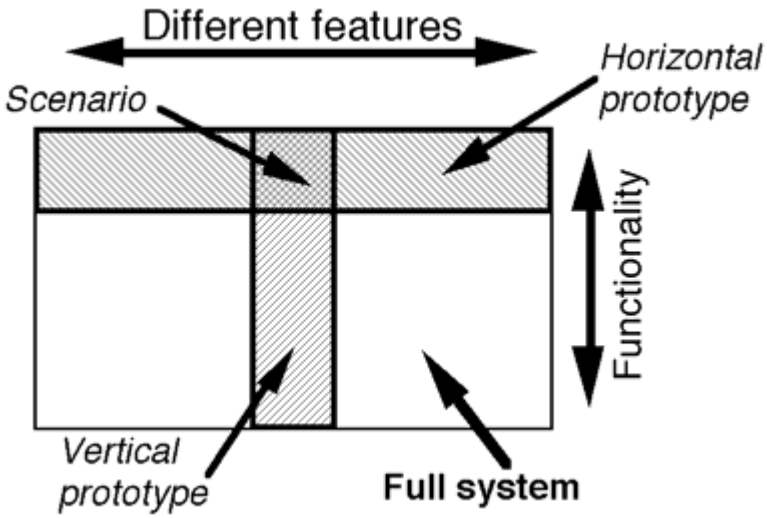

Figure 1
The concept of a *scenario* compared to vertical and horizontal prototypes as ways to make rapid prototyping simpler.

Scenarios take prototyping to the extreme by reducing both the level of functionality and the number of features. By reducing the part of interface being considered to the minimum, a scenario can be very cheap to design and implement, but it is only able to simulate the user interface as long as a test user follows a previously planned path.

Since the scenario is small, we can afford to change it frequently, and if we use cheap, small thinking aloud studies, we can also afford to test each of the versions. Therefore scenarios are a way of getting quick and frequent feedback from users.

Scenarios can be implemented as paper mock-ups [Nielsen 1990b] or in simple prototyping environments [Nielsen 1989a] that may be easier to learn than more advanced programming environments [Nielsen et al. 1991]. This is an additional savings compared to more complex prototypes requiring the use of advanced software tools.

2.2 Simplified Thinking Aloud

Traditionally, thinking aloud studies are conducted with psychologists or user interface experts as experimenters who videotape the subjects and perform detailed protocol analysis. This kind of method certainly may seem intimidating for ordinary developers. However, it is possible to run user tests without sophisticated labs, simply by bringing in some real users, giving them some typical test tasks, and asking them to think out loud while they perform the tasks. Those developers who have used the thinking aloud method are happy about it [Jørgensen 1989, Monk et al. 1993], and my studies [Nielsen 1992b] show that computer scientists are indeed able to apply the thinking aloud method effectively to evaluate user interfaces with a minimum of training, and that even fairly methodologically primitive experiments will succeed in finding many usability problems.

I have long claimed that one learns the most from the first few test users, based on several case studies. In earlier papers, I have usually recommended using between three and five test users per test as a way of simplifying user testing while gaining almost the same benefits as one would get from more elaborate tests with large numbers of subjects. Recently, Tom Landauer and I developed a mathematical model of the number of usability problems [Nielsen and Landauer 1993], and when plugging in typical budget figures from different kinds of user testing, we derived curves like the ones shown in Figure 2 for the ratio between the benefits of user testing and the cost of the test for medium-sized development projects. The curves basically show that the benefits from user testing are much larger than the costs, no matter how many subjects are used. The maximum benefit-cost ratio is achieved when using between three and five subjects, confirming my earlier experience.

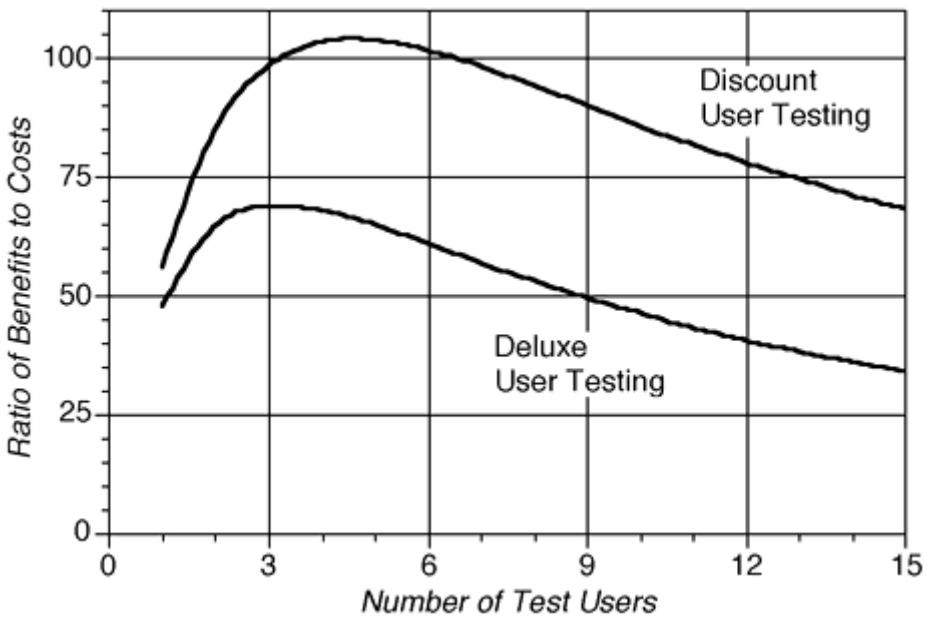

Figure 2
Cost-benefit trade-off curve for a "typical" project, varying the number of test users, using the model and average parameters described by Nielsen and Landauer [1993]. The curve shows the ratio of benefits to costs, that is, how many times the benefits are larger than the costs. For example, a benefit-to-cost ratio of 50 might correspond to costs of $10,000 and benefits of $500,000.

Besides reducing the number of subjects, another major difference between simplified and traditional thinking aloud is that data analysis can be done on the basis of the notes taken by the experimenter instead of by videotapes. Recording, watching, and analyzing the videotapes is expensive and takes a lot of time which is better spent on running more subjects and on testing more iterations of redesigned user interfaces. Video taping should only be done in those cases (such as research studies) where absolute certainty is needed. In discount usability engineering we don't aim at perfection anyway, we just want to find most of the usability problems, and a survey of 11 software engineers [Perlman 1988] found that they rated simple tests of prototypes as almost twice as useful as video protocols.

2.3 Heuristic Evaluation

Current user interface standards and collections of usability guidelines typically have on the order of one thousand rules to follow and are therefore seen as intimidating by developers. For the discount method I advocate cutting the complexity by two orders of magnitudes and instead rely on a small set of heuristics such as the ten basic usability principles (listed on a separate page).

These principles can be presented in a single lecture and can be used to explain a very large proportion of the problems one observes in user interface designs. Unfortunately it does require some experience with the principles to apply them sufficiently thoroughly [Nielsen 1992c], so it might be necessary to spend some money on getting outside usability consultants to help with a heuristic evaluation. On the other hand, even non-experts can find many usability problems by heuristic evaluation and many of the remaining problems would be revealed by the simplified thinking aloud test. It can also be recommended to let several different people perform a heuristic evaluation as different people locate different usability problems [Nielsen and Molich 1990]. This is another reason why even discount usability engineers might consider setting aside a part of their budget for outside usability consultants.

3 Validating Discount Usability Engineering

In one case, I used the discount usability engineering method to redesign a set of account statements [Nielsen 1989b]. I tested eight different versions (the original design plus seven redesigns) before I was satisfied. Even so, the entire project required only about 90 hours, including designing seven versions of twelve different kinds of statements (not all the forms were changed in each iteration, however) and testing them in simplified thinking aloud experiments. Most versions were tested with just a single user. To validate the redesign, a further experiment was done using traditional statistical measurement methods. It should be stressed that this validation was a research exercise and not part of the discount usability engineering method itself: The usability engineering work ended with the development of the improved account statements, but as a check of the usability

engineering methods used, it was decided to conduct a usability measurement of one of the new designs compared with the original design.

3.1 Experiment 1: Double Blind Test Taking Usability Measurements

The validation was done using a double blind test: 38 experimenters each ran four subjects (for a total of 152 subjects) in a between-subjects design. Neither the experimenters nor the subjects knew which was the original account statement and which was the new. The results which are reported in Table 3 show clear and highly statistically significant improvements in measurement values for the new statement with respect to the understandability of the information in the statement as measured by the average number of correct answers to four questions concerning the contents of the statement. The value had indeed been the usability parameter which had been monitored as a goal during the iterative design. Two other usability parameters which had not been considered goals in the iterative design process (efficiency of use and subjective satisfaction) were also measured in the final test, and the two versions of the statement got practically identical scores on those.

	Original design	Revised design	Significance of Difference
"Size of deposit"	79%	95%	$p<.01$
"Commission"	34%	53%	$p<.05$
"Interest rates"	20%	58%	$p<.01$
"Credit limit"	93%	99%	$p<.05$
Average correct	56%	76%	$p<.01$
Task time (sec.)	315	303	n.s. ($p=.58$)
Subjective satisfaction [1-5 scale]	2.8	3.0	n.s. ($p=.14$)

Table 3
Result of Experiment 1: a double blind test (N=152) comparing the original and the revised version of a bank account statement. The values measured are: How many of the subjects could correctly answer each of four questions about the contents of the statement (and the combined average for those four questions), the average time needed by subjects to review the statement and answer the questions, and the subjects' average subjective rating (scale: 1 [bad] to 5 [good]).
The rightmost column indicates whether the difference between the two account statements is statistically significant according to a *t*-test.

This study supports the use of discount usability engineering techniques and shows that they can indeed cause measurable improvements in usability. However, the results also indicate that one should be cautious in setting the goals for usability engineering work. Those usability parameters that have no goals set for improvement risk being left behind as the attention of the usability engineer is concentrated on the official goals. In this study, no negative effects in the form of actual degradation in measured usability parameters were observed but one can not always count on being so lucky.

3.2 Experiment 2: Recommendations from People without Usability

Expertise

Two groups of evaluators were shown the two versions of the account statement (without being told which one was the revised version) and asked which one they would recommend management to use. All the evaluators were computer science students who had signed up for a user interface design course but who had not yet been taught anything in the course. This meant that they did not know the usability heuristics which they might otherwise have used to evaluate the two versions.

Group A consisted of the experimenters from Experiment 1 (reported above) who had run two short experiments with each version of the account statement, while the evaluators in Group B had to make their recommendation on the basis of their own personal evaluation of the two versions. The results are reported in Table 4 and show a significant difference in the recommendations: Evaluators in Group A preferred the revised version 4 to 1 while evaluators in Group B were split equally between the two versions. This latter result is probably a reflection of the fact that the two versions are almost equally subjectively satisfying according to the measurement results reported in Table 3.

	Group A	Group B
	N=38	N=21
Recommends original	16%	48%
Recommends revised	68%	48%
No recommendation	16%	5%

Table 4
Result of Experiment 2: asking two group of evaluators to recommend one of the two versions of an account statement. In Group A, each person had first run an empirical test with four subjects, whereas the evaluators in Group B had no basis for their recommendation except their own subjective evaluation. The difference between the two groups is statistically different at the $p<.05$ level.

If we accept the statistical measurement results in Table 3 as defining the revised version as the "best," we see that Group A was dramatically better at making the correct recommendation than Group B was. This was in spite of the fact that each of the individuals in Group A had knowledge only of the experimental results from two subjects for each of the designs (the aggregate statistics were not calculated until after the recommendations had been made, so each evaluator knew only the results from the four subjects run by that individual).

So we can conclude that running even a small, cheap empirical study can help non-human factors people significantly in their evaluation of user interfaces. If we count the evaluators who did not make a recommendation as having a 50/50 chance of picking the right interface, this experiment shows that running just two subjects for each version in a small test improved the probability for recommending the best of two versions from 50% to 76%.

4 Cost-Benefit Analysis of Heuristic Evaluation: A Case Study

A cost-benefit analysis of heuristic evaluation includes two main elements: First estimating the costs in terms of time spent performing the evaluation, and second estimating the benefits in terms of increased usability (less the development costs for the redesign). Since these estimates involve some uncertainties, they will be converted into dollar amounts by using round numbers. Any given company will of course have slightly different conversion factors, depending on its exact financial circumstances.

The following case study regards a prototype user interface for a system for internal telephone company use which will be called the Integrating System in this chapter. The Integrating System is fairly complicated and understanding its details requires extensive knowledge of telephone company concepts, procedures, and databases. Since a detailed explanation is not necessary to understand the generally applicable lessons from the study, the Integrating System will only be outlined here.

Briefly, the Integrating System provides a graphical user interface to access information from several systems running on various remote computers in a uniform manner despite the differences between the backend systems. The Integrating System can be used to resolve certain problems when data inconsistencies require manual intervention by a technician because the computer systems cannot determine which information is correct. The traditional method for resolving these problems involves having the technician compare information across several of these databases by accessing them through a number of traditional alphanumeric terminal sessions. The databases reside on different computers and have different data formats and user interface designs, so this traditional method is somewhat awkward and requires the technicians to learn a large number of inconsistent user interfaces.

Performing this task involves a large amount of highly domain-specific knowledge about the way the telephone system is constructed and the structure of the different databases. Technicians need to know where to look for what data and how the different kinds of data are related. Also, the individual data items themselves are extremely obscure for people without detailed domain knowledge.

As a result of the heuristic evaluation of this interface with 11 evaluators (described in further detail in [Nielsen 1994b]), 44 usability problems were found. Forty of these problems are denoted "core" usability problems and were found in the part of the interface that was subjected to intensive evaluation, whereas the remaining four problems were discovered in parts of the interface that we had not planned to study as part of the heuristic evaluation.

4.1 Time Expenditure

As usual in usability engineering, the cost estimates are the easiest to get right. Table 5 accounts for the total time spent on the heuristic evaluation project in terms of person-hours. No attempt has been made to distinguish between different categories of professional staff. Practically all the person-hours listed in Table 5 were spent by usability specialists. The only exception is a small number of hours spent by development specialists in getting the prototype ready for the evaluation and in attending the debriefing session.

Assessing appropriate ways to use heuristic evaluation, 4 people @ 2 hours	**8**

Having outside evaluation expert learn about the domain and scenario	8
Finding and scheduling evaluators, 1.8 hours + 0.2 hours per evaluator	4
Preparing the briefing	3
Preparing scenario for the evaluators	2
Briefing, 1 system expert, 1 evaluation expert, 11 evaluators @ 1.5 hours	19.5
Preparing the prototype (software and its hardware platform) for the evaluation	5
Actual evaluation, 11 evaluators @ 1 hour	11
Observing the evaluation sessions, 2 observers @ 11 hours	22
Debriefing, 3 evaluators, 3 developers, 1 evaluation expert @ 1 hour	7
Writing list of usability problems based on notes from evaluation sessions	2
Writing problem descriptions for use in severity-rating questionnaire	6
Severity rating, 11 evaluators @ 0.5 hours	5.5
Analyzing severity ratings	2
Total	105

Table 5
Estimate of the total number of person-hours spent on the heuristic evaluation study described in this article. The estimate of "time to prepare the prototype" does not include the time needed for the initial task analysis, user interface design, or implementation of the prototype since these activities had already been undertaken independently of the heuristic evaluation.

Note that the time given for the preparation of the scenario covers only the effort of writing up the scenario in a form that would be usable by the evaluators during the evaluation. Considerable additional effort was needed to specify the scenario in the first place, but that effort was part of the general task analysis and design activities performed before the evaluation. Scenario-based design is a well-known method for user interface design [Carroll and Rosson 1990, Clarke 1991], so one will often be able to draw upon interaction scenarios that have been developed in previous stages of the usability lifecycle. Even so, we were probably lucky that the scenario developed for the present system could be used for the evaluation with such a small amount of additional effort.

The evaluation sessions were videotaped, and approximately eight hours were spent on mundane tasks like getting videotapes, learning to operate the video equipment in the specific usability laboratory used for the evaluation sessions, setting up and closing down the video equipment on each of the two days of the study, rewinding tapes, etc. This videotaping was not part of the heuristic evaluation as such, and the tapes were not reviewed for the purpose of arriving at the list of usability problems. The observers' notes were sufficient for that purpose. The videotapes were used to some extent in this research analysis of the study where an additional eight hours were spent reviewing details of some evaluation sessions, but since this use was not part of the practical application of the heuristic evaluation method, the time spent on the videotapes has not been included in Table 5.

It follows from Table 5 that the total number of person-hours spent on the evaluation can be determined by the formula

Equation 1:

time(*i*) = 47.8 + 5.2 *i*

where *i* is the number of evaluators. This formula is not exact for large values of *i*, since some of the effort devoted to room scheduling and to the analysis of the severity ratings is partly dependent on the number of evaluators and would change with large *i*s.

The cost estimate in (Equation 1) is probably larger than necessary for future heuristic evaluations. Major reductions in both the fixed and variable costs could be achieved by reducing the team of two observers to a single observer. This observer should be the person who is familiar with the application such that the observer can answer questions from the evaluators during the evaluation. Also, even though the observer should have a certain level of usability knowledge in order to understand the comments made by the evaluators, the observer need not be a highly skilled expert specializing in usability. A major difference between heuristic evaluation and traditional user testing is that an observer of a heuristic evaluation session is mostly freed from having to interpret user actions since the evaluators are assuming the task of explicitly identifying the usability problems. In contrast, the experimenter in a traditional user test would need a higher level of usability expertise in order to translate the subject's actions and difficulties into interface-related usability problems.

This single change would result in the following, revised formula

Equation 2:

time(*i*) = 37.3 + 4.2 *i*

Transforming the time estimates in (Equation 1) or (Equation 2) to money estimates can be done fairly simply by multiplying the number of hours by an estimate of the loaded hourly cost of professional staff. Note that the salary and benefits costs of the professional staff are not sufficient, since additional costs are incurred in form of the computer equipment and laboratory space used for the test. To use round numbers, an estimated hourly loaded cost for professional staff of $100 translates into a total cost for the heuristic evaluation of $10,500 for the 105 hours that were actually spent.

4.2 Benefit Estimation

The only way to get an exact measure of the benefits of the heuristic evaluation would be to fully implement two versions of the user interface; one without any changes and one with the changes implied by the evaluation results. These two versions should then be used by a large number of real users to perform real tasks for sufficiently long time that the steady-state level of expert performance had been reached in both cases [Gray et al. 1992]. This process would provide exact measures for the differences in learning time and expert performance. Unfortunately, the version of the interface that was evaluated only exists in a prototype form with which one cannot do any real work, and it would be unrealistic to expect significant development resources to be invested in transforming this prototype to a final product with an identical user interface now that a large number of usability problems have been documented.

Alternatively, one could build a detailed economic work-study model of the different steps involved in the users' workday in order to assess the frequency and duration of each sub-task. One could then further use formal models of user interaction times to estimate the duration of performing each step with each of a set of alternative user interface designs [Gray et al. 1992]. Such an approach would provide fairly detailed estimates but would not necessarily be accurate because of unknown durations of the operations in the model. It would also be very time-consuming to carry out.

It is thus necessary to rely on estimates of the benefits rather than hard measurement data. To get such estimates, the 11 evaluators were asked to estimate the improvements in usability from fixing all the 44 usability problems identified by the heuristic evaluation. Usability improvements were estimated with respect to two usability parameters:

- **Reduction of learning time: How much less time would the users need to spend learning to use the system? Learning time considered as a usability parameter represents a one-time loss of productive time for each new user to learn the system, so any savings would be realized only once.**
- **Speedup in expert performance: Once the users have reached a steady state of expert performance, how much faster would they be able to perform their work when using a system with all the usability problems fixed than when using a system with all the problems still in place? Expert performance considered as a usability parameter represents a continued advantage for the use of the improved interface, so any savings would be realized throughout the lifetime of the system.**

Other usability parameters of interest include frequency of user errors and the users' subjective satisfaction, but these parameters were not estimated. Since several of the usability problems we found were related to error-prone circumstances, it is likely that the number of user errors would go down.

Ten of the 11 evaluators provided learning time estimates and all 11 provided expert speedup estimates. Histograms of the distribution of these estimates are shown in Figure 3. Nielsen and Phillips [1993] found that estimates of changes in user performance made by usability specialists were highly variable, as also seen in the figure here, but that mean values of at least three independent estimates were reasonably close to the values measured by controlled experiments.

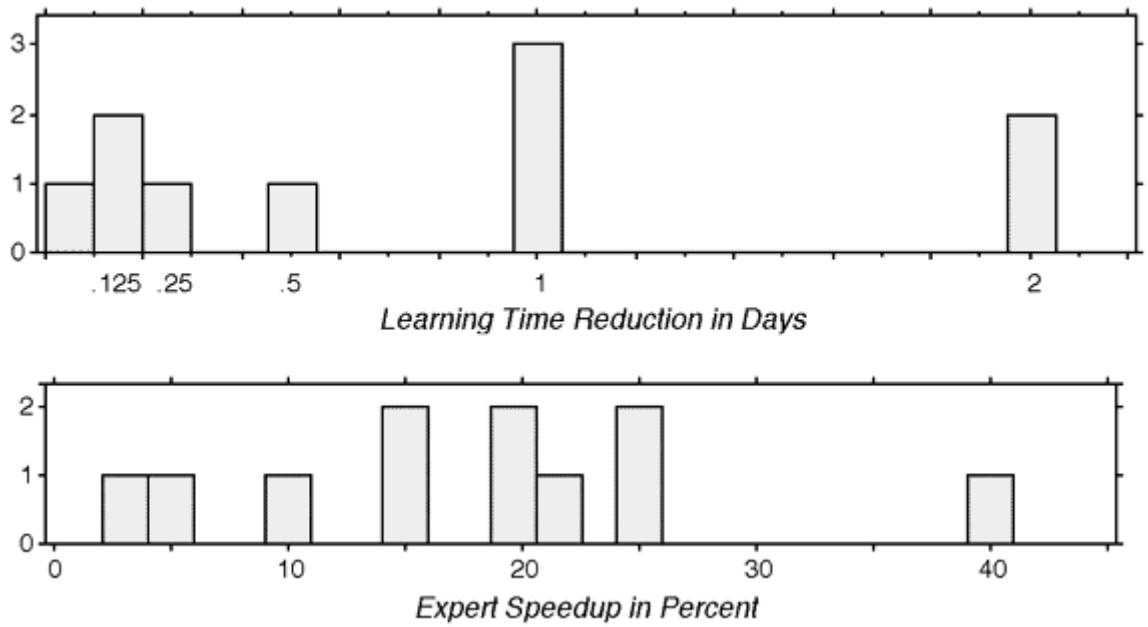

Figure 3
Histograms showing the distribution of the evaluators' estimates of savings in learning time (top) and expert performance speedup (bottom) for an interface fixing all the usability problems found in the heuristic evaluation. One evaluator did not provide a learning time estimate.

Given that the benefit estimates are based purely on subjective judgments of experts rather than on empirical evidence, it would seem prudent to be conservative in translating the evaluators' estimates into projected monetary savings. The mean values are 0.8 days for learning time reduction and 18% for expert speedup when all evaluators are considered, and 0.5 days and 16%, respectively, when excluding the perhaps overly optimistic outliers at 2 days and 40%. In order to be conservative, we will choose 0.5 days as our learning time reduction estimate and 10% as our expert speedup estimate.

The 10% expert speedup obviously only applies to time spent using the interface. Studies of the users indicate that they will spend about 1/3 of their time doing other tasks, 1/3 of their time performing the task without operating the user interface, and 1/3 of their time actually operating the interface. The 10% expert speedup thus corresponds to 3.3% of total work time.

Translating these estimates into overall savings can be done under the following assumptions: We assume that 2,000 people will be using the system. This is somewhat conservative given that about 3,000 people currently perform this job. Having 2,000 people each save 0.5 days in learning to use the system corresponds to a total of 1,000 user-days saved as a one-time saving. Furthermore, having 2,000 users perform their work 3.3% faster after having reached expert performance corresponds to 67 user-years saved each year the system is in use. Again to be conservative, we will only consider the savings for the first year, even though computer systems of the magnitude we are talking about here are normally used for more than one year. Sixty-seven user-years correspond roughly to 13,000 user-days saved. The total number of user-days saved the first year is thus about 14,000.

To value the total savings in monetary terms, we will assume that the cost of one user-day is $100, and to be conservative, we will assume that only half of the usability problems can actually be fixed, so that only half of the potential savings are actually realized. Furthermore, we need to take into account the fact that the savings in user time are not realized until the system is introduced and thus have a smaller net present value than their absolute value. Again to use round numbers, we will discount the value of the saved learning time by 20% and the value of the expert speedup in the first year by 30%. Learning time can be discounted by a smaller percentage as this saving is realized on day one after the introduction of the system. Using these conservative assumptions, we find one-year savings of $540,000.

Of course, the savings are not realized just by wishing for half of the usability problems to be fixed, so we have to reduce the savings estimate with an estimate of the cost of the additional software engineering effort needed to redesign the interface rather than just implementing the interface from the existing prototype. Assuming that the amount of software engineering time needed for this additional work is 400 hours, and again assuming that the loaded cost of a professional is $100 per hour, we find that the savings estimate needs to be reduced by $40,000. This expense is incurred here and now and thus cannot be discounted. Our final estimate of the net present value of improving the user interface is thus $500,000.

Still being conservative, we have not taken into account the value of the saved software engineering costs from not having to modify the system after its release. Assuming that the original user interface were to be fully implemented and released, is it very likely that the users would demand substantial changes in the second release, and it is well known

that making software engineering changes to a released system is much more expensive than making changes at a prototype stage of the software lifecycle.

The $500,000 benefit of improving the interface should be compared with the cost of the heuristic evaluation project, estimated at $10,500. We thus see that the benefit/cost ratio is 48. This number involves significant uncertainties, but is big enough that we do not hesitate to conclude that the heuristic evaluation paid off.

As a final comment on the cost-benefit analysis we should note that the "benefits" do not translate to an actual cash flow. Instead, they represent the avoidance of the penalty represented by the extra time the users would have had to spend if the prototype interface had been implemented and released without further changes. It is an interesting and an important management problem to find ways to properly represent such savings in the funding of software development.

4.3 Cost-Benefit Analysis of User Testing

After the heuristic evaluation exercise, additional user testing was performed on the same interface, running four test users. A major reason for using so many more heuristic evaluators than test users was that the users of this particular application were highly specialized technicians who were difficult to get into the lab, whereas it was reasonably easy to get a large number of usability specialists to participate in the heuristic evaluation session. Four new usability problems were found by the user testing which also confirmed 17 of the problems that had already been found by heuristic evaluation.

One can discuss whether the 23 core problems that were not observed in the user test are in fact "problems" given that they could not be seen to bother the real users. As argued elsewhere [Nielsen 1992b], such problems can indeed be very real, but their impact may just have too short a duration to be observable in a standard user test. Problems that have the effect of slowing users down for 0.1 second or so simply cannot be observed unless data from a very large number of users is subjected to statistical analysis, but they can be very real and costly problems nevertheless. Also, some problems may occur too infrequently to have been observed with the small number of users tested here.

The main cost of the user test activity was having two professionals spend 7 hours each on the running of the test and the briefing and debriefing of the test users. No time was needed for the traditionally time-consuming activity of defining the test tasks since the same scenario was used as that developed for the previous usability work. Additionally, half an hour was spent finding and scheduling the users for the test and two hours were spent on implementing a small training interface on which the users could learn to use a mouse and standard graphical interaction techniques like pull-down windows. These activities sum to a total of 16.5 person-hours of professional staff, or a cost of $1,650.

Furthermore, the four users and their manager spent essentially a full day on the test when their travel time is taken into account. Again assuming that the cost of one user-day is $100, and furthermore assuming that the cost of one manager-day is $200, the total cost of user involvement is $600. Adding the cost of the professionals and the users gives a total estimate of $2,250 as the cost of the user testing.

The $2,250 spent on user testing could potentially have been spent on additional heuristic

evaluation efforts instead. According to Equation 1, this sum corresponds to using 4.3 additional evaluators. Nielsen and Landauer [1993] showed that the finding of usability problems by *i* evaluators can be modelled by the prediction formula

Equation 3:

$$\text{ProblemsFound}(i) = N(1 - (1-l)^i)$$

For the core usability problems in the present study, the best-fit values for the parameters in this equation are N=40 and *l*=0.26. Increasing the number of heuristic evaluators, *i*, from 11 to 15.3 can thus be expected to result in the finding of about 1.1 additional usability problems. This estimate shows that the available additional resources do indeed seem to have been spent better on running a user test, finding four problems, than on potentially extending the heuristic evaluation further.

We have no systematic method to estimate the benefits of having found the four additional problems that were discovered by user testing. However, one easy way to arrive at a rough estimate is to assume that the average severity of the four new problems is the same as the average severity of the 17 problems that had already been found by heuristic evaluation. As part of the heuristic evaluation study, severity was measured on a rating scale, with each usability problem being assigned a severity score from zero to four, with higher scores denoting more serious problems. The sum of the severity scores for the original 44 usability problems was 98.41, and the sum of the severity scores for the 17 problems that were seen both in the user test and in the heuristic evaluation was 41.56. We can thus estimate the relative severity of the additional four problems as compared to the original problems as 4/17 xá41.56/98.41 = 0.099.

Knowing about the additional problems found by user testing would thus add 9.9% to the total potential for improving the interface. Furthermore, we might assume that the proportion of the new problems that can be fixed, the impact of fixing them, and the cost of fixing them are all the same as the estimates for the problems found by heuristic evaluation. Under these assumptions, the benefit of having found the additional four usability problems can be valued at $500,000 x 0.099 = $49,500.

Using these estimates, the benefit/cost ratio of adding the user test after the heuristic evaluation is 22. Of course, the benefits of user testing would have been larger if we had credited it with finding the problems that were observed during the user test but had already been found by the heuristic evaluation. We should note, though, that the cost of planning the user test would have been higher if the heuristic evaluation had not been performed and had confirmed the value of the usage scenario. Also, there is no guarantee that all the observed problems would in fact have been found if there had been no prior heuristic evaluation. Now, we knew what to look for, but we might not have noticed as many problems if the user test had been our first usability evaluation activity for this interface.

If the user test were to be credited with all 17 duplicate problems as well as the four new ones, taking the higher-than-average severity of the seventeen problems into account, the benefit of the user test would be valued at $260,500. Of course, this amount would be the benefit from the user test only if no prior heuristic evaluation had been performed. Therefore, it would seem reasonable to charge this hypothetical analysis of the user test with some of the costs that were in fact spent preparing for the heuristic evaluation. Specifically, referring to Table 5, we will add the costs of assessing the appropriate way to

use the method, having the outside evaluation expert learn about the domain and scenario, preparing the scenario, and preparing the software, as well as half the time spent writing the problem descriptions (since about half as many problems were found). These activities sum to 24 hours, or an additional cost of $2,400, for a total estimated cost of running the user test without prior heuristic evaluation of $4,650. This translates into a benefit/cost ratio of 56.

To provide a fair comparison, it should be noted that the benefit/cost ratio of performing the heuristic evaluation with only four evaluators would have been 53. This number is larger than the benefit/cost ratio for the full evaluation since more previously unfound usability problems are identified by the first evaluators than by the last, as shown by (EQ 3). Furthermore, the heuristic evaluation provided severity estimates that can be used to prioritize the fixing of the usability problems in the further development process, and the availability of this data probably adds to the actual value of the method as measured by delivered usability. If the time spent on the debriefing and severity ratings is deducted from the time spent on the heuristic evaluation, the benefit/cost ratio for the full eleven evaluators becomes 59 and the ratio for four evaluators becomes 71.

Thus, within the uncertainty of these estimates, it appears that user testing and heuristic evaluation have comparable cost-benefit ratios, and that doing some of each may have additional value.

5 The Evolution of Usability Engineering in Organizations

Two of the fundamental slogans of discount usability engineering are that "any data is data" and "anything is better than nothing" when it comes to usability. Therefore, I often advocate an approach to usability that focuses on getting started to use a minimum of usability methods. Even so, there are many projects that would benefit from employing more than the minimum amount of discount usability methods. I used the term "guerrilla HCI" in the title of this chapter because I believe that simplified usability methods can be a way for a company to gradually build up its reliance on systematic usability methods, starting with the bare minimum and gradually progressing to a more refined lifecycle approach.

Based on observing multiple companies and projects over the years, I have arrived at the following series of steps in the increased use of usability engineering in software development.

1. **Usability does not matter. The main focus is to wring every last bit of performance from the iron. This is the attitude leading to the world-famous error message, "beep."**
2. **Usability is important, but good interfaces can surely be designed by the regular development staff as part of their general system design. This attitude is symbolized by the famous statement made by King Frederik VI of Denmark on February 26, 1835: "We alone know what serves the true welfare and benefit of the State and People." At this stage, no attempt is made at user testing or at acquiring staff with usability expertise.**
3. **The desire to have the interface blessed by the magic wand of a usability engineer. Developers recognize that they may not know everything about usability,**

so they call in a usability specialist to look over their design and comment on it. The involvement of the usability specialist is often too late to do much good in the project, and the usability specialist often has to provide advice on the interface without the benefit of access to real users.

4. GUI panic strikes, causing a sudden desire to learn about user interface issues. Currently, many companies are in this stage as they are moving from character-based user interfaces to graphical user interfaces and realize the need to bring in usability specialists to advise on graphical user interfaces from the start. Some usability specialists resent this attitude and maintain that it is more important to provide an appropriate interface for the task than to blindly go with a graphical interface without prior task analysis. Even so, GUI panic is an opportunity for usability specialists to get involved in interface design at an earlier stage than the traditional last-minute blessing of a design that cannot be changed much. (Update added 1999: these days, this stage is often characterized by Web Panic Strikes. It's the same phenomenon and should be treated the same way.)
5. Discount usability engineering *sporadically* used. Typically, some projects use a few discount usability methods (like user testing or heuristic evaluation), though the methods are often used too late in the development lifecycle to do maximum good. Projects that do use usability methods often differ from others in having managers who have experienced the benefit of usability methods on earlier projects. Thus, usability acts as a kind of virus, infecting progressively more projects as more people experience its benefits.
6. Discount usability engineering *systematically* used. At some point in time, most projects involve some simple usability methods, and some projects even use usability methods in the early stages of system development. Scenarios and cheap prototyping techniques seem to be very effective weapons for guerrilla HCI in this stage.
7. Usability group and/or usability lab founded. Many companies decide to expand to a deluxe usability approach after having experienced the benefits of discount usability engineering. Currently, the building of usability laboratories [Nielsen 1994a] is quite popular as is the formation of dedicated groups of usability specialists.
8. Usability permeates lifecycle. The final stage is rarely reached since even companies with usability groups and usability labs normally do not have enough usability resources to employ all the methods one could wish for at all the stages of the development lifecycle. However, there are some, often important, projects that have usability plans defined as part of their early project planning and where usability methods are used throughout the development lifecycle.

This model is fairly similar to the series of organizational acceptance stages outlined by Ehrlich and Rohn [1994] but was developed independently. Stage 1-2 in the above list correspond to Ehrlich and Rohn's skepticism stage, stage 3-4 correspond to their curiosity stage, stage 5-6 correspond to their acceptance stage, and stage 7-8 correspond to their partnership stage.

Many teachers of usability engineering have described the almost religious effect it seems to have the first time students try running a user test and see with their own eyes the difficulties perfectly normal people can have using supposedly "easy" software. Unfortunately, organizations are more difficult to convert, so they mostly have to be conquered from within by the use of guerrilla methods like discount usability engineering that gradually show more and more people that usability methods work and improve products. It is too optimistic to assume that one can move a development organization

from stage 1 or 2 in the above model to stage 7 or 8 in a single, sweeping change. In reality, almost all usability methods are extremely cheap to use compared to the benefits they provide in form of better and easier to use products, but often we have to start with the cheapest possible methods to overcome the intimidation barrier gradually.

Acknowledgments

The author would like to thank Raldolph Bias, Tom Landauer, and Janice Rohn for helpful comments on an earlier version of the manuscript.

230 Tips for User Testing

For practical advice on user testing, see also our report 230 Tips and Tricks for a Better Usability Test.

References

Apple Computer (1987). Human Interface Guidelines: The Apple Desktop Interface. Addison Wesley, Reading, MA.

Apple Computer (1992). Macintosh Human Interface Guidelines. Addison Wesley, Reading, MA.

Bellotti, V. (1988). Implications of current design practice for the use of HCI techniques. In Jones, D.M. and Winder, R. (Eds.), People and Computers IV, Cambridge University Press, Cambridge, U.K., 13-34.

Boehm, B. W. (1981). Software Engineering Economics. Prentice-Hall, Englewood Cliffs, NJ.

Card, S. K., Moran, T. P., and Newell, A. (1983). The Psychology of Human-Computer Interaction, Lawrence Erlbaum Associates, Hillsdale, NJ.

Carroll, J. M., and Rosson, M. B. (1990). Human-computer interaction scenarios as a design representation. Proc. HICSS-23: Hawaii International Conference on System Science, IEEE Computer Society Press, 555-561.

Clarke, L. (1991). The use of scenarios by user interface designers. In Diaper, D., and Hammond, N. (Eds.), People and Computers VI, Cambridge University Press, Cambridge, U.K. 103-115.

Ehrlich, K., and Rohn, J. (1994). Cost-justification of usability engineering: A vendor's perspective. In Bias, R.G., and Mayhew, D.J. (Eds.), Cost-Justifying Usability. Academic Press, Boston, MA.

Gould, J. D., and Lewis, C. H. (1985). Designing for usability: Key principles and what designers think. Communications of the ACM 28, 3 (March), 300-311.

Gray, W. D., John, B. E., and Atwood, M. E. (1992). The precis of project Grace, or, an overview of a validation of GOMS. Proc. ACM CHI'92 (Monterey, CA, 3-7 May 1992), 307-312.

Jørgensen, A.H. (1989). Using the thinking-aloud method in system development. In Salvendy, G. and Smith, M.J. (Eds.), Designing and Using Human-Computer Interfaces and Knowledge Based Systems. Elsevier Science Publishers, Amsterdam, 743-750.

Karwowski, W., Kosiba, E., Benabdallah, S., and Salvendy, G. (1989). Fuzzy data and communication in human-computer interaction: For bad or for good. In Salvendy, G. and Smith, M.J. (Eds.), Designing and Using Human-Computer Interfaces and Knowledge Based Systems. Elsevier Science Publishers, Amsterdam, 402-409.

Landauer, T. K. (1988). Research methods in human-computer interaction. In Helander, M. (Ed.), Handbook of Human-Computer Interaction. North-Holland, Amsterdam, The Netherlands. 543-568.

Mantei, M. M., and Teorey, T.J. (1988). Cost/benefit analysis for incorporating human factors in the software lifecycle, Communications of the ACM 31, 4 (April), 428-439.

Milsted, U., Varnild, A., and Jørgensen, A.H. (1989). Hvordan sikres kvaliteten af brugergr¾nsefladen i systemudviklingen ("Assuring the quality of user interfaces in system development," in Danish), Proceedings NordDATA'89 Joint Scandinavian Computer Conference (Copenhagen, Denmark, 19-22 June), 479-484.

Molich, R., and Nielsen, J. (1990). Improving a human-computer dialogue, Communications of the ACM 33, 3 (March), 338-348.

Monk, A., Wright, P., Haber, J., and Davenport, L. (1993). Improving Your Human-Computer Interface: A

Practical Technique. Prentice Hall International, Hemel Hempstead, U.K.

Nielsen, J. (1989a). Prototyping user interfaces using an object-oriented hypertext programming system, Proc. NordDATA'89 Joint Scandinavian Computer Conference (Copenhagen, Denmark, 19-22 June), 485-490.

Nielsen, J. (1989b). Usability engineering at a discount. In Salvendy, G., and Smith, M.J. (Eds.), Designing and Using Human-Computer Interfaces and Knowledge Based Systems, Elsevier Science Publishers, Amsterdam. 394-401.

Nielsen, J. (1990a). Big paybacks from 'discount' usability engineering, IEEE Software 7, 3 (May), 107-108.

Nielsen, J. (1990b). Paper versus computer implementations as mockup scenarios for heuristic evaluation, Proc. INTERACT'90 3rd IFIP Conf. Human-Computer Interaction (Cambridge, U.K., 27-31 August), 315-320.

Nielsen, J. (1992a). The usability engineering life cycle. IEEE Computer 25, 3 (March), 12-22.

Nielsen, J. (1992b). Evaluating the thinking aloud technique for use by computer scientists. In Hartson, H. R. and Hix, D. (Eds.), Advances in Human-Computer Interaction Vol. 3, Ablex, Norwood, NJ. 75-88.

Nielsen, J. (1992c). Finding usability problems through heuristic evaluation. Proc. ACM CHI'92 (Monterey, CA, 3-7 May), 373-380.

Nielsen, J. (1993). Usability Engineering. Academic Press, Boston, MA.

Nielsen, J. (1994a). Usability laboratories. Behaviour & Information Technology 13, 1.

Nielsen, J. (1994b). Heuristic evaluation. In Nielsen, J., and Mack, R.L. (Eds.), Usability Inspection Methods. John Wiley & Sons, New York, NY.

Nielsen, J., and Landauer, T. K. (1993). A mathematical model of the finding of usability problems. Proc. ACM INTERCHI'93 Conf. (Amsterdam, the Netherlands, 24-29 April), 206-213.

Nielsen, J., and Levy, J. (1994). Measuring usability - preference vs. performance. *Communications of the ACM* 37, 4 (April), 66-75.

Nielsen, J., and Molich, R. (1989). Teaching user interface design based on usability engineering, ACM SIGCHI Bulletin 21, 1 (July), 45-48

Nielsen, J., and Molich, R. (1990). Heuristic evaluation of user interfaces, Proc. ACM CHI'90 (Seattle, WA, 1-5 April), 249-256.

Nielsen, J., and Phillips, V. L. (1993). Estimating the relative usability of two interfaces: Heuristic, formal, and empirical methods compared. Proc. ACM INTERCHI'93 Conf. (Amsterdam, the Netherlands, 24-29 April), 214-221.

Nielsen, J., Frehr, I., and Nymand, H. O. (1991). The learnability of HyperCard as an object-oriented programming system, Behaviour and Information Technology 10, 2 (March-April), 111-120.

Perlman, G. (1988). Teaching user interface development to software engineers, Proc. Human Factors Society 32nd Annual Meeting, 391-394.

Perlman, G. (1990). Teaching user-interface development, IEEE Software 7, 6 (November), 85-86.

Telles, M. (1990). Updating an older interface, Proc. ACM CHI'90 (Seattle, WA, 1-5 April), 243-247.

Thovtrup, H., and Nielsen, J. (1991). Assessing the usability of a user interface standard, Proc. ACM CHI'91 (New Orleans, LA, 28 April-2 May), 335-341.

Tognazzini, B. (1990). User testing on the cheap, Apple Direct 2, 6 (March), 21-27. Reprinted as Chapter 14 in *TOG on Interface*, Addison-Wesley, Reading, MA, 1992.

Voltaire, F. M. A. (1764). Dictionnaire Philosophique.

Whiteside, J., Bennett, J., and Holtzblatt, K. (1988). Usability engineering: Our experience and evolution. In Helander, M. (Ed.), Handbook of Human-Computer Interaction, North-Holland, Amsterdam, 791-817.