Designing and Using Human–Computer Interfaces and Knowledge Based Systems
edited by G. Salvendy and M. J. Smith
Elsevier Science Publishers B.V., Amsterdam, 1989 – Printed in the Netherlands

Usability Engineering at a Discount

Jakob Nielsen
Technical University of Denmark

Department of Computer Science; Building 344
DK-2800 Lyngby Copenhagen; Denmark
email: datJN@NEUVM1.bitnet

Abstract

The "discount usability engineering" method consists of scenarios, simplified thinking aloud, and heuristic evaluation and is intended to alleviate the current problem where usability work is seen as too expensive and difficult by many developers.

1. Introduction

Usability engineering [Whiteside et al. 1989] is the discipline of improving the usability of user interfaces in a situation of resource constraints. Many methods for usability engineering exist and are available for use *if* people want to use them. Unfortunately, the available evidence shows that many companies do *not* use such basic usability engineering techniques as early focus on the user, empirical measurement, and iterative design [Gould and Lewis 1983].

One important reason usability engineering is not used in practice is the cost of using its techniques. Or rather, the reason is the *perceived* cost of using those techniques, as this paper 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 recent paper in the widely read 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.

British studies [Bellotti 1988] indicate that many developers don't use usability engineering because HCI is seen as too time consuming and expensive and because the techniques are often intimidating in their complexity. This paper aims at addressing these two problems. 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 problems must be addressed by education and propaganda, but even for that purpose, simpler usability methods should help.

2. The "Discount" Method

We present a set of methods which is substantially cheaper in use than the more

extensive usability engineering methods usually advocated by most researchers. These methods may not be quite as scientifically valid as more advanced and complicated methods and they may not find quite as many of the usability problems in a given interface. But they stand a much better chance of actually being used in practical design situations in smaller companies and they should therefore be viewed as a way of serving the user community.

The "discount usability engineering" method is based on the use of the following techniques:

- scenarios [Nielsen 1987] and fast iteration
- small thinking aloud studies [Nielsen 1988]
- heuristic evaluation [Molich and Nielsen 1989]

This is really a mixture of two different methods. The first method is empirical and consists of the development of scenarios which are tested in small thinking aloud studies, often using only a single test subject. On the basis of the thinking aloud results, the scenario is changed using fast iteration and is then tested again. The other method is more analytical and is used in the inner loop of the design (i.e. during the fast iteration). Heuristic evaluation is used to let designers make informed judgements when they are trying to decide what changes to make in the scenario and how to translate the scenario into a complete design.

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).

Scenarios finally reduce *both* the level of functionality and the number of features and are 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 or in simple prototyping environments such as HyperCard [Nielsen 1989]. This is an additional savings compared to more complex prototypes requiring the use of advanced software tools.

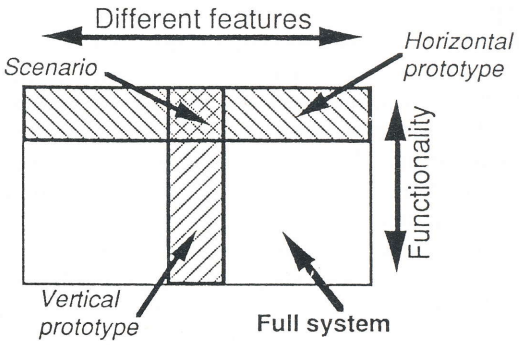

Figure 1.
The two dimensions of prototyping.

2.2. Simplified Thinking Aloud

Traditionally, thinking aloud studies are conducted with psychologists or user interface experts as ex-

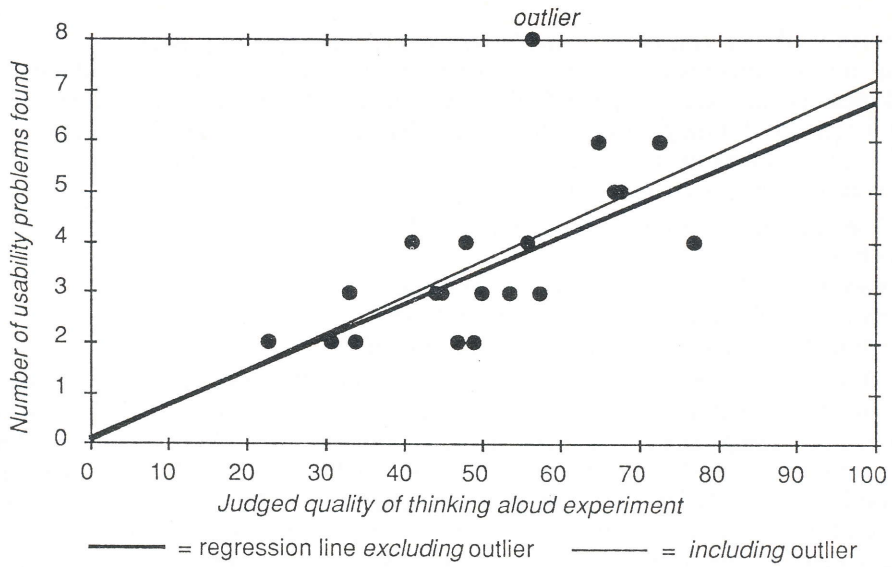

Figure 2.
Scatterplot of regression analysis of relation between the quality of a thinking aloud experiment (on a 0-100 scale) and the number of usability problems found in MacPaint v. 1 (out of a total of 8 problems). Regression equation excluding the outlier:

$$y = 0.067x + 0.115, \quad R^2 = 0.58$$

perimenters who videotape the subjects and perform detailed protocol analysis. This kind of method is certainly intimidating for ordinary developers and it is perhaps not surprising that recent studies of Danish computer companies and data processing department [Milsted et al. 1989] showed that only 6 % used the thinking aloud method. Those developers who *have* used the thinking aloud method seem [Jørgensen 1989] to be happy about it, however, and my studies [Nielsen 1988] 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.

Figure 2 summarizes the results from [Nielsen 1988] and shows that most of the experimenters were able to find about half of the usability problems in the program tested on the basis of doing thinking aloud studies of three subjects per group of experimenters. The thinking aloud studies discussed here were the first ones conducted by these experimenters and one would expect them to do better next time. It is interesting to extrapolate along the regression line to see what would happen in a 100% methodologically correct experiment: The estimate from Figure 2 is that such an experiment would find about 80 % of the total number of usability problems. Since this is a very nice proportion for a discount method, we conclude that discount thinking aloud studies normally should not use more than three experimental subjects. If there is time to run more subjects, it is probably better to change the user interface first on the basis of the usability problems identified from the first three studies.

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 in-

stead of by video tapes. Recording, watching, and analyzing the video tapes 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.

Simple and natural dialogue
Speak the user's language
Minimize user memory load
Be consistent
Provide feedback
Provide clearly marked exits
Provide shortcuts
Good error messages
Prevent errors

Table 1.
Usability heuristics after [Molich and Nielsen 1989].

2.3. Heuristic Evaluation

Current 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 we advocate cutting the complexity by two order of magnitudes and instead rely on a small set of heuristics such as the nine basic usability principles from [Molich and Nielsen 1989] listed in Table 1.

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, so it might be necessary to spend some money on getting an outside usability consultant to do 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. This is another reason why even discount usability engineers might consider setting aside a part of their budget for outside usability consultants.

Table 2 shows the final result of adjusting a usability budget according to the discount usability engineering method. The numbers in Table 2 are for a medium scale software project (about 32,000 lines of code). For small projects, even cheaper methods can be used as discussed in the examples below, while really large projects ought to devote sufficient funds to usability to allow use of the full-blown traditional methodology.

3. Two Examples: Redesigning Bank and Pension Statements

Two projects were conducted using the "discount usability engineering" method in practice. The first project concerned the improvement of a set of bank account statements and other computer printouts sent from a small Danish bank to its customers. The original designs were quite good as they had been developed by bank staff with knowledge of usability principles, but it was still felt that they could be improved. The second project concerned developing a printout of individualized projections of the result of investing in Individual Retirement Accounts (IRAs). Both sets of printouts were improved using the scenario technique, fast iteration, and cheap thinking aloud studies as well as usability heuristics.

At the beginning of the bank statement redesign a small study of the vocabulary

Original usability cost estimate by [Mantei and Teorey 1988]	$ 128,330
Scenario developed as paper mockup instead of on video tape	- $ 2,160
Prototyping done in HyperCard instead of in professional software tool	- $ 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
Outside usability consultant for heuristic evaluation	+ $ 3,000
Cost for "discount usability engineering" project	**$ 65,330**

Table 2.
Cost savings in a medium scale software project by using the discount usability engineering method instead of the more thorough usability methods normally recommended.

used in the original statements was conducted. 26 words were listed on a one page questionnaire and 30 people rated each word on a 1-5 scale according to how well they knew its meaning. This is a typical "discount" method, since a more careful study would have asked the respondents to also write down their definition of each word because there is a large risk that people will say that they do know a word even if they don't. Indeed, it turned out that only 7 of the 26 words were rated at or below average understandability, and the thinking aloud studies later in the projects revealed problems with some of the other words also.

The method was used in spite of these problems because it was cheap: It was easy to get respondents since they were not intimidated by having to write down word definitions and since it took only about 5 minutes to answer the questionnaire. The result was of considerable value in the initial redesigns since it indicated the kind of words which should definitely be avoided, and furthermore the "hard" nature of the data made it easy to convince bank representatives that words which they felt to be very natural could be hard to understand. For example, the official abbreviation of the Danish currency, *DKK,* got the second lowest rating of the 26 terms tested. At the time of the test, the international standard for currency abbreviations was still new enough to be unknown and people only knew the traditional abbreviation, *Dkr.*

Several of the changes made during the redesign followed well-known usability heuristics, such as e.g.

- Consistency: Write all amounts in favor of the bank with a minus and all numbers in favor of the customer with a plus. To be user-oriented, use these signs rather than the reverse. For external consistency, write the prefix in front of the number instead of following the Cobol legacy of writing them at the end of the number.
- The *gestalt* law of proximity: Instead of writing "kr._____100", write "_____100 kr." (where the underscores represent spaces in the printout to allow room for larger amounts, and *kr.* is the standard abbreviation for the Danish currency, *kroner,* which can customarily be written either before or after the digits).

During empirical testing of the IRA designs, a new usability principle became apparent: *Enable users to check their own understanding.* To some extent this is a corollary of the general principle of providing feedback, but in the context of a non-interactive computer printout, feedback takes a special meaning. In reading the very

complicated IRA calculations, including strange taxation rules, projections of inflation and interest rates, etc., the test subjects naturally became quite cautious and wanted to check that they had understood the different items in the printout correctly. They typically did so by matching numbers between the different parts of the printout which they felt should correspond. In the first several versions of the design, this was not always immediately possible e.g. because a pension was listed in one table as amount available at retirement and in another table as sum of yearly payments (which is different because of accruing interest).

	Original design	*Revised design*	
"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.*
Subj. eval. [1-5]	2.8	3.0	*n.s.*

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 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.*

In later versions of the design, *enable users to check their own understanding* was used as a design heuristic and care was taken to have the same numbers appear in tables referring to the same concept (together with the calculation showing e.g. why the final amount in the table was larger or smaller because of interest or taxes).

In some of the versions of the IRA projection, a graphical representation was tested in addition to the traditional tables of numbers. The test subjects were very enthusiastic about these graphs and felt that they were much more easy to grasp than the tables. In spite of this, the thinking aloud studies showed that several subjects actually misinterpreted the graph.

In both projects empirical testing was done with the basic purpose of assessing the understandability of the computer printouts and finding where they should be improved. Simplified thinking aloud was used since it is ideal for these goals.

In the design of bank account statements, we tested 8 different versions (the original design plus 7 redesigns) before we were satisfied. Even so, the entire project required only about 90 hours, including designing 7 versions of 12 different kinds of bank statement (not all the forms were changed in each version, however) and testing them in thinking aloud experiments. The IRA projection was developed in 11 versions, 6 of which were tested with either one or two subjects. This project required about 60 hours.

4. Validation of the Redesign

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 bank 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.

4.1. Experiment 1: Double Blind Test Taking Usability Measurements

The validation was done using a double blind test: 38 experimenters each ran 4 subjects (for a total of 152 subjects) in a between-subjects design. Neither the experimenters nor the subjects knew which was the original bank 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 on the usability parameter which had been the goal during the iterative design (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). Two other usability parameters which had not been considered goals in the iterative design process (efficiency of use and subjective satisfaction) were also measured and the two versions of the statement got practically identical scores on those.

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 which 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 probably not always count on being so lucky.

4.2. Experiment 2: Recommendations from People without Usability Expertise

Two groups of evaluators were shown the two versions of the bank statement (without being told which one was the revised version) and asked which one they would recommend the bank 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 e.g. the usability heuristics from Table 1 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 bank 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 prefer the revised version while evaluators in Group B are split equally between the two versions. This latter result is probably a reflection of the fact that the two versions are almost equally subjectively

	Grp. **A** *N=38*	Grp. **B** *N=21*
Recommends original	16 %	48 %
Recommends revised	68 %	48 %
Can't recommend either	16 %	5 %

Table 4.
*Result of Experiment 2: asking two groups of people to recommend one of the two versions (original or revised) of a bank account statement. In Grp. **A**, each person had first run an empirical test with four subjects, while the people in Grp. **B** had no basis for their recommendation other than their own subjective evaluation.*
The difference between the two groups is statistically significant at the $p<.05$ *level.*

satisfying according to the measurement results reported in Table 3.

If we accept the statistical measurement results in Table 3 as defining the revised version as the "best", we see that Group A was drastically more able to make the correct recommendation than Group B. This was in spite of the fact that each of the individuals in Group A had knowledge only of the experimental results from the four subjects run by that individual (the aggregate statistics were not calculated until after the recommendations had been made).

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 eliminate the evaluators who did not make a recommendation, 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 81%.

5. Conclusions

The discount usability engineering method is significantly cheaper than traditional methods but even so it seems to identify most of the usability problems which can be found by the expensive methods. It should be stressed that the discount method is *not* as good as the traditional methods: Using traditional methods *will* give you more information in many cases. But on the other hand, using the discount method is much, much better than using *no* usability methods at all. Even if the result is not perfect, the resulting product will still have been improved significantly. In practice, especially in smaller companies or in smaller projects, there often is just no other feasible alternative than to rely on the discount usability method if any usability work is to be done.

References

Bellotti, V.: 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, UK, 1988, pp. 13-34.

Gould, J.D. and Lewis, C.: Designing for usability: Key principles and what designers think, *Proc. ACM CHI'83* (Boston, MA , 12-15 December 1983), pp. 50-53.

Jørgensen. A.H.: Using the thinking-aloud method in system development, *Proc. Third Intl. Conf. Human-Computer Interaction* (Boston, MA, 18-22 September 1989).

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

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

Molich, R., and Nielsen, J.: Improving the human-computer interface: What designers know about traditional interface design, manuscript submitted for publication 1989.

Nielsen, J.: Using scenarios to develop user friendly videotex systems, *Proc. NordDATA'87 Joint Scandinavian Computer Conference* (Trondheim, Norway, 15-18 June 1987).

Nielsen, J.: Evaluating the thinking aloud technique for use by computer scientists, *Proc. IFIP W.G. 8.1. Intl. Workshop on Human Factors of Information Systems Analysis and Design* (London, UK, 28-29 July 1988).

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

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

Whiteside, J., Bennett, J., and Holtzblatt, K.: Usability engineering: Our experience and evolution, in M. Helander (Ed.): *Handbook of Human-Computer Interaction,* North-Holland, 1989.