A Mathematical Model of the Finding of Usability Problems

Jakob Nielsen and Thomas K. Landauer
Bellcore
445 South Street
Morristown, NJ 07962-1910
USA
nielsen@bellcore.com and tkl@bellcore.com

Electronic business card for Nielsen can be retrieved by sending any email message to the server at nielsen-info@bellcore.com

ABSTRACT

For 11 studies, we find that the detection of usability problems as a function of number of users tested or heuristic evaluators employed is well modeled as a Poisson process. The model can be used to plan the amount of evaluation required to achieve desired levels of thoroughness or benefits. Results of early tests can provide estimates of the number of problems left to be found and the number of additional evaluations needed to find a given fraction. With quantitative evaluation costs and detection values, the model can estimate the numbers of evaluations at which optimal cost/benefit ratios are obtained and at which marginal utility vanishes. For a "medium" example, we estimate that 16 evaluations would be worth their cost, with maximum benefit/cost ratio at four.

Keywords: Usability problems, Usability engineering, Poisson models, User testing, Heuristic evaluation, Cost-benefit analysis, Iterative design.

INTRODUCTION

Both user testing [6][8][18][14][22] and heuristic evaluation [15][21] can be considered as interface debugging tests with respect to their positioning in the usability engineering lifecycle [16]. Their goal is to find and document as many usability problems in a user interface design as possible so that the problems can be corrected in future versions.

The two methods are quite different since user testing is based on bringing real users in and observing them as they interact with the system in order to perform a given set of tasks, whereas heuristic evaluation is based on having usability specialists judge a user interface on the basis of established usability principles. The mechanics of conducting these two kinds of evaluations are very different, and both methods have advantages and disadvantages. For example, user testing provides insights into the mindset and working methods of real users. Heuristic evaluation is easier to set up as there is no need to bring in users or to analyze interaction protocols, and exploits expert scientific and experiential knowledge of general problems and solutions.

From a general usability engineering perspective, however, user testing and heuristic evaluation have two important similarities. First, they are both debugging methods. Second, both involve aggregating results from multiple smaller evaluation studies. Even though one can in principle conduct user testing with a single test user and heuristic evaluation with a single evaluator, good practice calls for the use of several test users and evaluators. Typically, each test user or each evaluator identifies a certain list of usability problems, and these lists are then aggregated for however many test users or evaluators are employed for a given study.

Most current usability engineering work (e.g., [5][7][14][15][20][21][22]) in effect has a *post hoc* view of the actual evaluation process, analyzing the complete results of finished studies as a whole. In practical development environments, one will often be interested in looking at partial analyses of usability studies as they are performed with a view towards deciding when enough information has been gained. Especially when resources are tight and a discount usability engineering approach [12][13] is applied with an emphasis on using as few test users or heuristic evaluators as possible, one might be interested in a predictive model that can provide some estimate of the total number of usability problems in an interface, *during* the evaluation process even if all the problems have not been found yet.

DATASETS

The following analysis contains data from case studies of the empirical user testing of five user interfaces and the heuristic evaluation of six user interfaces. Table 1 lists all eleven studies and summarizes some of their pertinent char-

© 1993 ACM 0-89791-575-5/93/0004/0206...$1.50

System Tested [with publication reference]	Evaluation Method	Type of Interface	Number Subjs./ Evals.	Observed Usability Problems	Problems found by one evaluatn.	Model		
						λ fit	N fit	R^2
Office System (integrated spreadsheet etc.) [9]	User test	Personal computer, GUI	15	145	16%	.12	166	.996
Calendar program [22]	User test	Personal computer, GUI	20	40	36%	.32	39	.986
Word processor [17]	User test	Personal computer, GUI	24	9	30%	.30	9	1.00
Outliner (manipulate hierarchical structures) [17]	User test	Personal computer, GUI	30	14	28%	.26	14	.998
Bibliography database [24]	User test	Personal computer, CUI	13	29	31%	.29	24	.969
Teledata (info on airline departures) [21]	Heuristic evaluation	Videotex, CUI	37	52	51%	.48	50	.971
Mantel (hypothetical white pages system) [11]	Heuristic evaluation	Mainframe, CUI	77	30	38%	.34	26	.937
Banking system (transfer money between accounts) [15]	Heuristic evaluation	Voice response	31 19 14	16	22% 41% 60%	.21 .38 .58	12 16 16	.965 .991 .992
Savings (info on account balances and currency exchange) [21]	Heuristic evaluation	Voice response	34	48	26%	.26	40	.988
Transport (information to public on bus routes) [21]	Heuristic evaluation	Voice response	34	34	19%	.22	27	.995
Integrating system (internal telephone company application) [20]	Heuristic evaluation	Workstation, GUI	11	40	29%	.26	40	.995
Mean values				42	33%	.31	41	

Table 1 *List of the user interface evaluations analyzed in this paper.*
Notes: GUI = Graphical User Interface, CUI = Character-based User Interface. The Banking *interface was evaluated by three different groups of evaluators: 31 non-specialists, 19 usability specialists, and 14 double specialists with expertise in both usability and the kind of interface being evaluated. The mean value of N-fit was calculated using the result from only one of these groups (N-fit=16).*

acteristics, as well as the results of fitting the Poisson model discussed in the next section to the data. Since the Banking System was evaluated by three independent groups of evaluators, it generated three datasets, so the total number of datasets is thirteen. The Office System was a collection of applications that together provided integrated support for standard professional work tasks, including a spreadsheet, a word processor, a file system, etc., but it was tested as a single system.

A POISSON MODEL OF THE FINDING OF USABILITY PROBLEMS

To develop the mathematical model, we make some basic assumptions about the finding of usability problems. In using the simple and well-understood probabilistic model of Poisson processes, we assume that the probability of finding any given usability problem in any given test is independent of the outcome of previous tests. Not only is the Poisson model well-behaved, but it has also been found to describe the finding of traditional programming errors ("bugs") [1][2] in software development projects under some conditions. This problem is similar to the "debugging" of user interface designs implied by usability engineering.

For some kinds of usability evaluation, the Poisson assumption seems quite reasonable. For example, heuristic evaluations are often conducted by having evaluators inspect the interface independently of each other. With this procedure it would seem obvious that the probability of having any given evaluator find any one specific problem would be independent of whether the other evaluators had found that problem.

For user testing, the finding of a usability problem depends of two factors: First, the subject has to experience the problem, and second, the experimenter has to realize that the user experienced the problem. To the extent that user tests are run according to classic experimental methodology, subjects can probably be considered independent of each other. Regarding the experimenter's role, however, the independence assumption may not hold in all cases. Some usability problems may not be easy to recognize, and the probability of having the experimenter find the problem in a test session where a user encounters the problem may therefore be dependent on whether the experimenter had seen (but not recognized) the problem before in previous test sessions. In spite of this potential difficulty, it would still seem likely that most aspects of the experimenter's finding of usability problems can be approximated by a Poisson model.

The discussion so far has concentrated on the finding of individual usability problems. In real projects, there are of course many usability problems in any given user interface, and we would like to have a model that accounts for the finding of several usability problems. As argued above, the finding of each individual usability problem probably follows a Poisson model. However, the parameters of the Poisson model will most likely be different for each usability problem. Nielsen found that severe problems were more likely to be found than less severe problems in heuristic evaluation, though not much more so [15], and Virzi found that severe problems were much more likely to be found than less severe problems by user testing [23].

For any given interface, some problems will be easy to find, being glaring design catastrophes and/or being encountered by almost all test users, whereas others will be more difficult to find, being more subtle and/or only encountered under special circumstances. However, adding two independent Poisson processes with respective rates A and B yields a Poisson process with rate A+B [3], so we can still model the finding of the set of usability problems with a single Poisson model, even though the different problems have different probabilities of being found. This additive property of the Poisson model assumes that detection of various usability problems are independent of each other. This assumption is a simplification, since some user difficulties do tend to co-occur.

Given the assumptions discussed above (that the finding of usability problems are independent of whether they have been found before and independent of each other), we can expect a Poisson model to describe the finding of usability problems. The number of usability problems that have been found at least once by i evaluators or subjects is

$$\mathrm{Found}(i) = N(1 - (1-\lambda)^i) \qquad \text{(EQ 1)}$$

where N is the total number of problems in the interface, and the parameter λ is the probability of finding the average usability problem when running a single, average subject, or using a single, average heuristic evaluator.* Table 1 shows the results of fitting this model to the data from the studies for 1 through 15 evaluators or subjects, using a least squares method. For the Integrating System, the fitted model was based on data for 1 through 11 evaluators, and for the Bibliography database, the fitted model was based on data for 1 through 13 test users. The fits are mostly very good as indicated by the high values of R^2 in Table 1, and as also seen in Figure 1, which shows the fitted model curves and the datapoints from the original datasets.

$1-\lambda$ is the probability of a usability problem remaining unfound for one more test given that it has not been found already. λ thus indicates the probability of finding a usability problem with one more person given that it has not been found yet. As can be seen from Table 1, this value corresponds fairly closely to the proportion of usability problems actually found by single heuristic evaluators or when running single subjects. The single-person values from the studies are larger than the fitted values, however. Such a discrepancy would be accounted for by a small positive correlation between the problems found by different people.

Similarly, Table 1 indicates that the model slightly underestimates the total number of usability problems N for most interfaces. The main exception is the Office System, where the model predicts 166 problems, 21 more than were found by running 15 subjects. Closer examination of the underlying data indicates that it is reasonable to assume that the Office System actually contains more usability problems than were found by the test with the 15 subjects. Out of the 145 problems that were found, as many as 76 were observed for only a single subject, thus providing some indication that there might be more remaining problems that would have been found by running additional subjects. The Office System actually comprised several different applications, so 15 subjects may simply not have been enough for an exhaustive test of this type of major integrated system. Overall, the model seems to slightly underestimate N, so applications of the model should preferrably rely on estimates of the number of *remaining* problems to be found rather than the total number of problems in the interface.

* The model as presented is simplified by assuming that all evaluations (subjects or heuristic evaluators) find exactly the mean number of usability problems. (That is, if one finds 10 and another 20, we act as if both found 15, though the overlap is greater between two sets of 15 than between a set of 10 and a set of 20.) This has the effect of somewhat underestimating the probability of finding a problem, λ, and consequently also somewhat overestimating the total number of usability problems, N. The model also ignores the likely positive correlation between some problems. This has an opposing effect. (Overall, we seem to be underestimating N). Relaxing these two assumptions would lead to a more complex model with more parameters, which would, therefore, require more data before predictions could be made. From our results, it appears that the simpler model is sufficiently accurate for practical purposes.

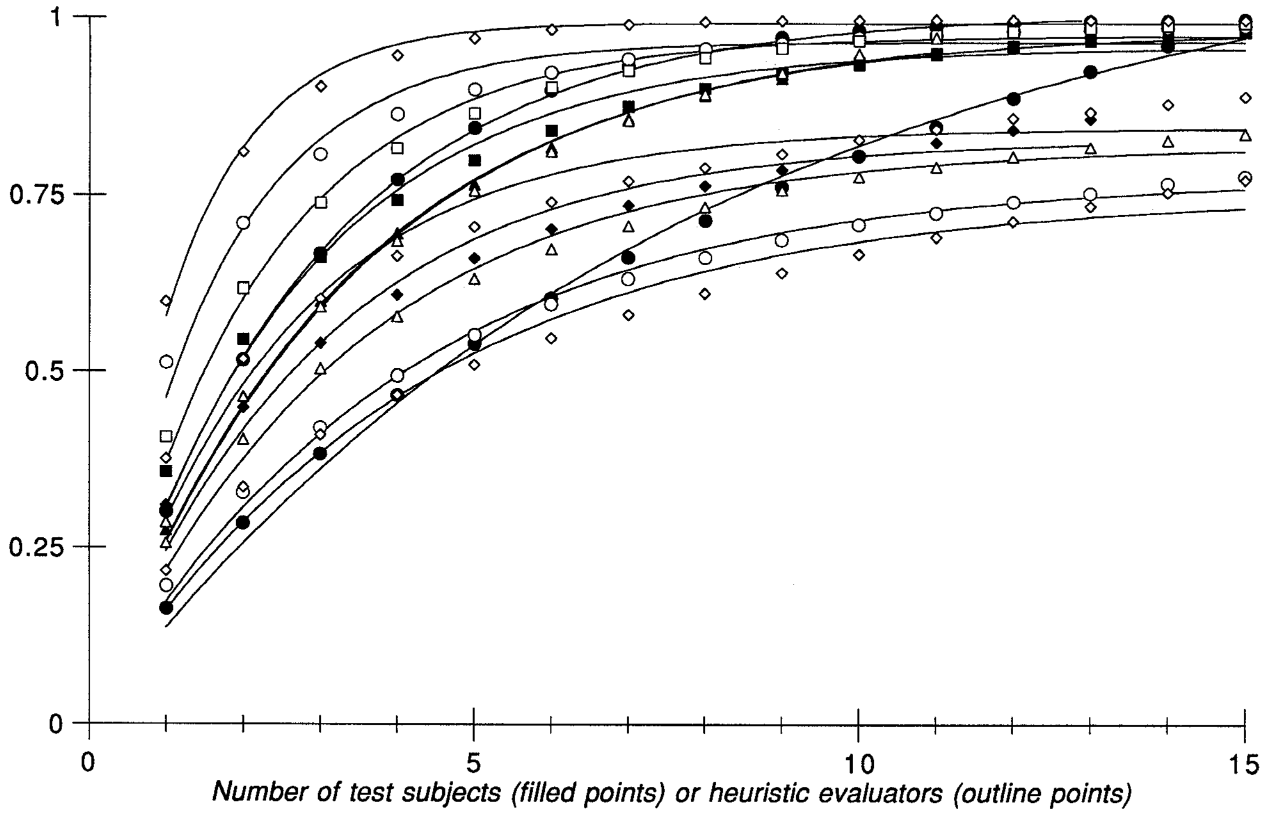

Figure 1 *Proportion of usability problems found with increasing numbers of subjects or evaluators for the interfaces in Table 1. The markers indicate the actual values from the studies and the lines indicate the fitted curves according to (EQ 1). The values from the various studies have been normalized to proportions rather than absolute number of problems to allow comparisons in a single figure.*

As can be seen from Table 1, λ varies substantially between studies. The exact value of λ in any given usability study will depend on

- The properties of the system and its interface.
- The stage of the usability lifecycle. For example, it might well be the case that usability problems are easier to find in an initial rough design with plenty of obvious problems than in a polished *n*'th iteration. Also, it might matter whether the interface has been fully implemented or only exists as a prototype or a paper design. For example, Nielsen [15] found that usability problems relating to missing features in a user interface were much harder to find by heuristic evaluation of paper mockups than of running prototypes.
- The evaluation method used. For example, an evaluation based on an analysis of logs of user interactions might require data from more users than a thinking aloud study would.
- The skills of the heuristic evaluators and the experimenters running a user test. For example, Nielsen [15] found that evaluators with usability expertise found many more problems in a heuristic evaluation than evaluators without such expertise and that evaluators with "double expertise" (both usability and the kind of interface being evaluated) found even more problems. Nielsen and Molich [11] found a positive correlation of .57 between the number of usability problems found by the same evaluators in two case studies of heuristic evaluation, indicating that some people tend to be better than others even within a given expertise category. As another example, Nielsen [14] found that the number of usability problems found in user testing using the thinking aloud technique had a positive correlation of .76 with the methodological quality of the test procedures used.
- Other factors, such as whether test users are representative of the actual user population.

ESTIMATING THE NUMBER OF PROBLEMS REMAINING TO BE FOUND

One use for this model would be as an aid to deciding when to stop testing. After two or more evaluations, one can easily estimate λ and *N*. It is impossible to derive estimates based only on the first subject or evaluator, as (EQ 1) has two unknowns, and therefore needs at least two datapoints to be estimated. Of course, one could still get a rough estimate of *N* if λ was assumed to be equal to some value that had previously been found to describe usability studies in a given organization. It is unknown, however, whether the variability in λ would in fact be smaller if one only consid-

Number of Test Users or Evaluators on which Estimate was Based	Standard Deviation of Estimated N (Expressed as a Proportion of Observed N)
2	42%
3	44%
4	21%
5	11%
6	9%
7	8%
8	6%
9	5%
10	5%

Table 2 *The spread of estimates of N made on the basis of various number of test users or evaluators, expressed as the standard deviation of the estimates. Values are given across the datasets.*

ered user interfaces of a given type developed in a single organization and evaluated by the same usability staff using the same methodology.

Once estimates for *N* and λ have been made, one has a rough idea of the number of usability problems yet to be found in the user interface. The model also indicates the likely number of these problems that will be found by the next test subject or heuristic evaluator. If a usability manager knows the approximate value of finding additional problems, it is then a simple matter to decide whether it would be worth the additional cost to continue the test.

The accuracy of these estimates of *N* and λ will improve as more data becomes available. As shown in Table 2, the estimates are highly variable as long as they are based on data from two or three test users or heuristic evaluators, and they are reasonably tight when data is available from about six or more people. For *i* greater than two, the parameters can be estimated by least-squares curve-fitting, for which programs are available in many graphics and statistics packages. For *i*=2, the parameters can be estimated as follows (derived from (EQ 1) by simple algebraic manipulation):

$$\lambda = 2 - (\text{Found}(2) / \text{Found}(1)) \qquad \text{(EQ 2)}$$

$$N = \text{Found}(1) / \lambda \qquad \text{(EQ 3)}$$

where Found(*i*) indicates the number of different usability problems found after *i* evaluations. Both for curve fitting and for use of (EQ 2) and (EQ 3) it is recommended to calculate values for Found(*i*) that are independent of the particular sequence in which the evaluations were performed. For example, Found(1) should be the average number of usability problems found in a single evaluation and not just the number found by whatever evaluator or test subject happened to be the first. Similarly, Found(2) should be calculated as the mean number of problems found by all pairs of evaluations, Found(3) should be calculated based on all triplets, and so on. In practice, it is not necessary to consider all permutations of evaluations, but one should sample a reasonably large number of combinations.

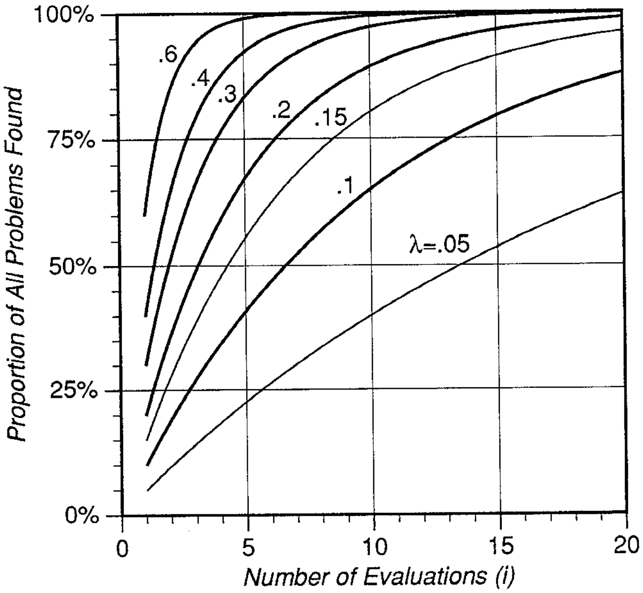

Figure 2 *Nomograph showing the proportion of usability problems found for various number of evaluations. Each curve represents a certain value of λ, as noted on the curves, from .05 (bottom curve) to .6 (top curve).*

Estimates of *N* are useful for development projects as they give an indication of how many usability problems might remain in an interface and how much additional work would be required to find them. Estimates of λ can help plan a testing program based on the expected shape of the curve of found usability problems. The nomograph in Figure 2 shows such curves over a common range of λ-values.

A PRIORI ESTIMATES OF THE OPTIMUM NUMBER OF EVALUATORS AND TEST USERS

The decision of when to stop trying to find usability problems will obviously depend on the characteristics of the individual development project, including especially the specific costs of each test user or heuristic evaluator as well as the probable savings to be realized from improved usability in the released software. Given this information, as well as information about the values of *N* and λ that are normally found in an organization's projects, one can also calculate rough a priori estimates of the amount of usability work that will be necessary. These estimates are obviously much less reliable than estimates made by fitting the model to measured data as it is accumulated from actual usability activities, but they can still be of some value for early planning.

This section provides examples of how a priori cost-benefit estimations can be made, using sample data estimated from

Project Size	*Heuristic Evaluation*	*User Testing*
Small	9	7
Medium-large	16	15
Very large	21	20

Table 3 *Estimates of the optimal number of heuristic evaluators and test users for our examples.*

Project Size	*Cost*	*Benefits*	*Benefit/ Cost Ratio*
Small	$9,400	$39,500	4.2
Medium-large	$13,600	$613,000	45
Very large	$33,200	$8,200,000	247

Table 4 *Cost–benefit analysis for using the optimal number of evaluators in a heuristic evaluation.*

our experience and the published literature. We will consider three sample projects, called small, medium-large, and very large. In reality, the magnitude of usability projects is not a simple, one-dimensional property. Several parameters influence the scope of usability activities, including the size of the interface (whether measured in lines of code or in number of screens, dialog boxes, or other interface elements), the number of expected users, and the duration, intensity, and possible mission-critical nature of their usage.

Costs seem to be the easiest to estimate. In one analysis [20], heuristic evaluation was estimated as having fixed costs of between $3,700 and $4,800, with the variable cost of each evaluator being between $420 and $520. User testing using an extreme "discount usability engineering" approach with very little preparation or data analysis was estimated as having fixed costs of $2,600, with the variable cost of each test user being $410. Another analysis [10] did not consider heuristic evaluation, but estimated the fixed costs of user testing at $8,000 and the variable costs per test user at $2,000 (when updated to 1993 dollars).

Other studies have been less explicit in calculating the costs of usability engineering techniques, but one can convert information in one study [7] into the following estimates: The fixed cost of preparing to use heuristic evaluation was about $4,400, and the fixed cost of preparing to use user testing was about $3,400. The variable cost was about $900 per evaluator for heuristic evaluation and about $1,900 per test user for user testing. Another study [5] did not provide information about fixed versus variable costs, but estimated about $500 in total costs per heuristic evaluator and $3,000 in total costs per test user.

These estimates vary strikingly, which is understandable given the differences in user interfaces being evaluated and the methodologies being applied. For example, one would expect a test of a large system to take longer and thus be more expensive than a test of a small, walk-up-and-use system. For illustration, we will use fixed costs of $4,000 for heuristic evaluation and $3,000 for user testing and variable costs of $600 per evaluator for heuristic evaluation and $1,000 per test user for user testing, except for the "very large" project, where all costs will be assumed to be twice as large.

Project Size	*Cost*	*Benefits*	*Benefit/ Cost Ratio*
Small	$10,000	$37,900	3.8
Medium-large	$18,000	$613,000	34
Very large	$46,000	$8,200,000	178

Table 5 *Cost–benefit analysis for using the optimal number of test users in user testing.*

Benefits are harder to calculate. One analysis [20] conservatively estimated the mean benefit from having found a usability problem as $13,500, not including the software engineering savings from not having to change the interface in a maintenance release of the product. Another analysis [10] estimated the mean benefit from having found a usability problem as $19,300 (when updated to 1993 dollars). Both these estimates considered systems that were going to see fairly extensive use, and for the sake of simplicity, we will use $15,000 as the benefit estimate per usability problem found for such systems. For smaller systems that are going to be used less frequently or by fewer users (for example, much in-house software in medium-sized companies), we will use $1,000 as the benefit estimate per usability problem found.

For systems that are going to see extremely intensive use by very large numbers of users, the benefit of finding a usability problem can be considerably higher and reach into the millions of dollars [4]. Such large systems probably warrant individual analyses taking their special circumstances into account, but again for simplicity, we will use $200,000 as the benefit estimate per usability problem found.

For all systems, we will use the prediction formula in (EQ 1) with the mean values of N and λ from Table 1, N=41, and λ=.31. Of course, the actual values of N and λ will vary for any given development project, so the following results should be seen only as very approximate "rules of thumb" that can be used to arrive at rough estimates before the start of a project. It is recommended that project managers acquire estimates of the cost and benefit values from their own organizations and that they also refine the estimates of N and λ for their specific project as data becomes available

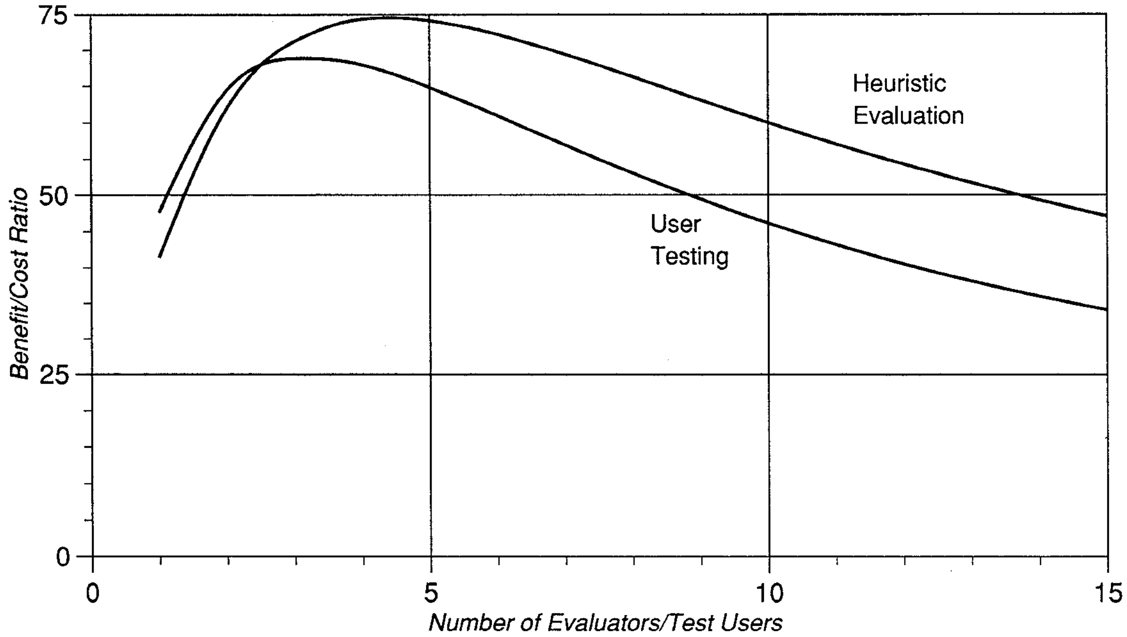

Figure 3 *Ratio between benefits and costs for using various numbers of heuristic evaluators and test users to find usability problems in a medium-large software project, as calculated using the various assumptions listed in the text.*

from their own usability activities. Of course, estimates for any specific project should also be refined as soon as a few evaluations have been performed for that project.

Given estimates for both costs and benefits as well as our prediction formula for the finding of usability problems, one can easily calculate the optimum number of heuristic evaluators or test users. The optimum number of evaluators or test users is the number for which the marginal value of the last evaluator or test user was higher than the marginal cost of that evaluator or user, but where the marginal value of one additional evaluator or user would be smaller than the marginal cost of that evaluator or user. Table 3 shows these optimal numbers. Table 4 then shows a cost–benefit analysis for using the optimum number of heuristic evaluators, and Table 5 shows a cost–benefit analysis for using the optimum number of test users.

These optimum numbers of evaluators/test users are much larger than obtained for our earlier "discount usability engineering" recommendation of using about five heuristic evaluators or test users [12]. One reason for the discrepancy is that discount usability engineering has as one of its goals to let people apply usability engineering methods on projects where budget or time constraints prevent them from using optimal methods. A second, and more fundamentally important, reason is that one would rarely evaluate a single user interface design to the bitter end without applying iterative design to fix usability problems found with the first few evaluators or test users. The estimates of the benefits of finding a usability problem assume that reasonable fixes will be introduced to the design, but by changing the design, one often introduces new usability problems. It is therefore likely to be a better strategy to evaluate initial iterations of a design less thoroughly.

Figure 3 shows the cost-benefit model for medium-large projects under our assumptions. It can be seen that the benefits are much larger than the costs both for user testing and for heuristic evaluation. The highest ratio of benefits to costs is achieved for 3.2 test users and for 4.4 heuristic evaluators. These numbers can be taken as one rough estimate of the effort to be expended for usability evaluation for each version of a user interface subjected to iterative design.

It seems reasonable to use our model also for iterative design [19], even though we have only tested it with data from single usability studies. Presumably, the usability problems found in each iteration of an iterative design process is a combination of previously unfound problems left over from earlier iterations and new problems introduced in the revised design. Given the Poisson assumption in our model, the probability of finding the previously unfound problems in the new iteration does not depend on how much testing has been conducted on previous iterations, so the model can be applied without modifications to the finding of all the problems, no matter whether they are new or old.

A model of iterative design should also account for the cost of producing the additional iterations. Unfortunately, such software development costs are extremely difficult to estimate. For the sake of argument, we will perform the calculations for a medium-large project, assuming that a new iteration can be produced for $20,000. This cost estimate might apply to a project that was still in an early stage and was being developed with a prototyping tool making changes reasonably easy to make. Taking $20,000 as an additional fixed cost for each usability study changes the model to have the highest ratio of benefits to costs at 6.7 test users and 7.9 heuristic evaluators. If iterations can be produced more cheaply, fewer users or evaluators should be used, and if iterations are more costly, more elaborate usability evaluations should be performed for each iteration.

CONCLUSIONS

We have established that a Poisson model describes quite well the finding of usability problems in user testing and heuristic evaluation. Such a model can therefore be used to predict the eventual number of problems that will be found by a usability study even as the study is in progress. Further work remains to be done to assess the preciseness of these predictions, but similar models have been successfully applied to the related problem of determining when to stop testing software for programming bugs [1][2].

Acknowledgments

The authors would like to thank James Lewis of the IBM Design Center/Human Factors, Robert Virzi of GTE Laboratories, and Peter Wright and Andrew Monk of the University of York for providing us with the detailed raw data underlying their published studies. We also thank Clare-Marie Karat of IBM U.S. Marketing and Strategy for clarifying details regarding fixed costs in her published study. The analyses of this data in this paper are solely the responsibility of the authors of the present paper and should not be taken as necessarily corresponding to the positions of these other authors or their organizations. The authors would also like to thank the anonymous *INTERCHI'93* referees for helpful comments on a previous version of this manuscript.

References

1. Dalal, S.R., and Mallows, C.L. (1988). When should one stop testing software? *J. American Statistical Association* **83**, 403 (September), 872–879.
2. Dalal, S.R., and Mallows, C.L. (1990). Some graphical aids for deciding when to stop testing software. *IEEE J. Selected Areas in Communication* **8**, 2 (February), 169–175.
3. Erhan, S. (1975). *Introduction to Stochastic Processes.* Prentice Hall, Englewood Cliffs, NJ. p. 87.
4. Gray, W.D., John, B.E., and Atwood, M.E. (1992). The precis of project Ernestine, or, an overview of a validation of GOMS. *Proc. ACM CHI'92* (Monterey, CA, 3–7 May), 307–312.
5. Jeffries, R., Miller, J.R., Wharton, C., and Uyeda, K.M. (1991). User interface evaluation in the real world: A comparison of four techniques. *Proc. ACM CHI'91* (New Orleans, LA, 27 April–2 May), 119–124.
6. 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.* Amsterdam: Elsevier Science Publishers, 743–750.
7. Karat, C., Campbell, R., Fiegel, T. (1992). Comparisons of empirical testing and walkthrough methods in user interface evaluation. *Proc. ACM CHI'92* (Monterey, CA, 3–7 May), 397–404.
8. Lewis, C. (1982). Using the 'thinking-aloud' method in cognitive interface design. *Research Report* **RC-9265**, IBM T.J. Watson Research Center, Yorktown Heights, NY.
9. Lewis, J.R., Henry, S.C., and Mack, R.L. (1990). Integrated office software benchmarks: A case study. *Proc. INTERACT'90 3rd IFIP Conf. Human–Computer Interaction* (Cambridge, U.K., 27–31 August 1990), 337–343.
10. 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.
11. Molich, R., and Nielsen, J. (1990). Improving a human-computer dialogue. *Communications of the ACM* **33**, 3 (March), 338–348.
12. Nielsen, J. (1989). 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.
13. Nielsen, J. (1990). Big paybacks from 'discount' usability engineering. *IEEE Software* **7**, 3 (May), 107–108.
14. Nielsen, J. (1992). 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. 69–82.
15. Nielsen, J. (1992). Finding usability problems through heuristic evaluation. *Proc. ACM CHI'92* (Monterey, CA, 3–7 May), 373–380.
16. Nielsen, J. (1992). The usability engineering lifecycle. *IEEE Computer* **25**, 3 (March), 12–22.
17. Nielsen, J. (1993). Estimating the number of subjects needed for a thinking aloud test. *Intl. J. Man–Machine Studies* **in press.**
18. Nielsen, J. (1993). *Usability Engineering*, Academic Press, San Diego, CA.
19. Nielsen, J. (1993). Iterative design of user interfaces. *IEEE Computer* **26** (to appear, probably in the July issue).
20. Nielsen, J. (1993). Heuristic evaluation. In Nielsen, J., and Mack, R.L. (Eds.), *Usability Inspection Methods. Book in preparation.*
21. Nielsen, J., and Molich, R. (1990). Heuristic evaluation of user interfaces. *Proc. ACM CHI'90* (Seattle, WA, 1–5 April), 249–256.
22. Virzi, R.A. (1990). Streamlining the design process: Running fewer subjects. *Proceedings of the Human Factors Society 34th Annual Meeting* (Orlando, FL, 8–12 October), 291–294.
23. Virzi, R.A. (1992). Refining the test phase of usability evaluation: How many subjects are enough? *Human Factors* **34**, 4 (August), 457–468.
24. Wright, P.C., and Monk, A.F. (1991). A cost-effective evaluation method for use by designers. *Intl. J. Man–Machine Studies* **35**, 6 (December), 891–912.