# Estimating the Relative Usability of Two Interfaces: Heuristic, Formal, and Empirical Methods Compared

*Jakob Nielsen and Victoria L. Phillips**
Bellcore
445 South Street
Morristown, NJ 07962-1910
USA
nielsen@bellcore.com and vp0o+@andrew.cmu.edu
Electronic business card for first author can be retrieved by sending any message to the server at nielsen-info@bellcore.com

*) Current address: Carnegie Mellon University, Pittsburgh, PA 15213, USA

## ABSTRACT

Two alternative user interface designs were subjected to user testing to measure user performance in a database query task. User performance was also estimated heuristically in three different ways and by use of formal GOMS modelling. The estimated values for absolute user performance had very high variability, but estimates of the relative advantage of the fastest interface were less variable. Choosing the fastest of the two designs would have a net present value more than 1,000 times the cost of getting the estimates. A software manager would make the correct choice every time in our case study if decisions were based on at least three independent estimates. User testing was 4.9 times as expensive as the cheapest heuristic method but provided better performance estimates.

**Keywords:** Heuristic evaluation, heuristic estimation, GOMS, User testing, Usability, User performance, Absolute performance, Relative performance, Cost-benefit estimates.

## INTRODUCTION

One often needs to assess the relative usability of two or more user interfaces, designed to solve the same problem. Sometimes it will be sufficient to find which of the two is the most usable and then pick the best. In other cases, one will need to know *how much* better the preferred interface is before one can decide to use it. A typical situation, and the one that prompted the current study, is the case where one interface design already is implemented. Since there are obviously significant development costs associated with changing such an interface, one would like an estimate of the amount of money to be saved from the supposed increased usability of the alternative design.



Other examples where relative assessments of usability are needed include design teams considering alternative designs where one is expected to be more expensive than the other to implement, or cases where two alternative software packages are available for purchase, but at different prices. Common for all these cases is the basic phenomenon that the available choices have different costs associated with them, meaning that it is not possible to simply choose the best user interface. One needs to estimate whether the best interface will lead to savings that are sufficiently large to justify its higher cost. Of course, if the cheapest interface was also the best, there would be no need for any further analysis.

From a research perspective, we have a general interest in estimating the benefits of usability engineering in monetary terms [7]. Cost/benefit analyses are important assessment tools for proposed new usability engineering methods [4]. The *costs* of various usability methods are easy to measure, since they involve mostly adding up staff and subject hours and multiplying with the relevant loaded salary rates. The *benefits* of usability methods are much harder to assess, since one rarely fields two complete versions of the same product in order to measure real-world performance with the two versions (but see [3] for an example of a study doing just that and saving $2.4 million).

The study described in this paper takes a slightly simplified approach to the estimation of usability benefits. We have studied the estimation of the task time needed for expert users to perform a certain task with two alternative user interface designs. Given a task analysis of the work of the real users, one can convert such task time estimates into person-hours per year, by simply multiplying the time saved per transaction with the number of times per day each user is expected to perform the task as well as by the number of users doing the job. These person-hour estimates can then be converted into monetary terms as shown below.

A full cost/benefit analysis would include additional elements, such as estimates of learning times, error rates, and subjective user satisfaction (and its probable impact on

employee turnover etc.). One would also need to consider potential differences between usability as measured in the laboratory and usability as observed in real-world usage. Furthermore, of course, much software is produced for sale on the open market rather than for use in-house or by contractual clients, and the relation between customer savings and increased market share for the vendor is not well understood.

The relative importance of the different usability variables will differ depending on the system. For many applications, initial learnability is paramount as users will not be expected to use the system more than a few times. For other applications, such as the one giving rise to the current study, users keep using the same interface repeatedly, and their performance as expert users is the dominating factor in overall usability. For example, for our full system (with more interface elements than the one discussed here), the savings from improving the interface according to the results of a heuristic evaluation were estimated at about $40,000 from reduced training costs and about $500,000 from increased expert user performance in the first year alone [12].

The study described in this paper has therefore focused on expert user performance. We have studied two alternative interface designs for a single, quite specific task: That of looking up information in one out of several databases.

We were interested in comparing various methods for estimating user performance. We collected data for the two interfaces from heuristic estimates by usability specialists, from formal GOMS analyses of the interfaces, and from user testing. Some usability specialists may argue that user testing is the only true way of getting data about user performance, but alternative methods such as formal analysis or heuristic estimates have the advantage that they can be used even when the interfaces in questions have not yet been implemented, and thus cannot be tested. Also, heuristic evaluation has mostly been used for the finding of usability problems [12], so it is interesting to assess whether it can also be used to estimate user performance.

## THE TASK AND THE INTERFACES

The users' task in this study is that of issuing a query to one out of several databases on one or more data items such as telephone numbers, customer names, addresses, etc. For the purpose of the current study, we looked at queries on telephone numbers. Furthermore, we only studied queries on one or two telephone numbers, even though the actual application sometimes involves queries for more than two telephone numbers. The users will be working in a window system, and the information that is to be looked up in the databases can be assumed to be visible on the screen (possibly as the result of previous queries).

Two possible interface designs were considered. Design A was fairly simple to implement and had indeed been implemented in a prototype version of the full system. Design B was a potential alternative design that was arrived at in a debriefing session following a heuristic evaluation of Design A. Design B is harder to implement than Design A, since it requires the computer to parse the result of database queries performed on a remote host system and to understand the meaning of the text displayed on the screen. Also, Design A had the obvious advantage of already being implemented in the prototype.

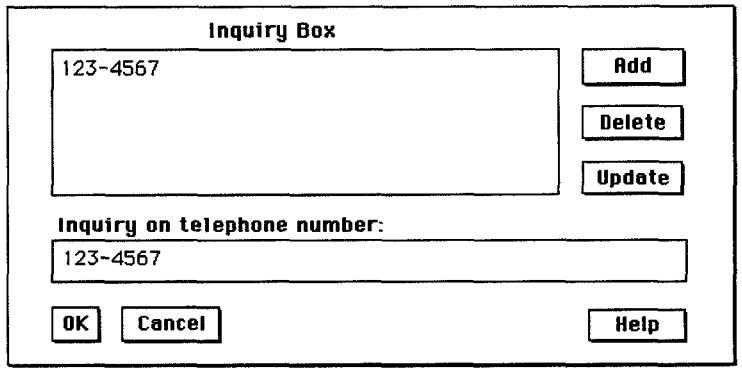


*Figure 1* *Dialog box used in Design A: The Dialog Box interface. In the state depicted here, the user has just input the telephone number "123-4567" and has clicked on the "Add" button. Figure shown at 60% of screen size.*

We should emphasize that the issue of interest here is not whether these two interfaces were optimally designed. It is very likely that they could be improved by iterative design using various usability engineering methods [11]. The question of interest for our study was how to assess expected expert user performance on two given designs.

### Design A: Dialog Box

To use this interface, the user first pulls down a menu from the menubar at the top of the screen. This menu contains the names of the databases and the user positions the mouse cursor over the name of the relevant database in this list. The user pushes down the mouse button and drags the mouse to the right to get a hierarchical submenu with the legal queries for the chosen database. This submenu contains an average of four alternatives, and the user moves the mouse until it highlights the option "query on telephone number." The user then releases the mouse button.

This causes a standard sized dialog box to appear in the middle of the screen as shown in Figure 1. The large field at the top is initially empty. This field will eventually list the telephone numbers to be submitted to the database as a query. The dialog box does not overlap the window with the list of telephone numbers. The user clicks in the input field (the one-line field below the prompt "Inquiry on telephone number") and types the telephone number. The user clicks on the "Add" button to add the number to the query. The figure shows the state of the dialog box after the user's click on "Add." If the query is for a single telephone number, the user then clicks on the "OK" button to submit the query.

If the query is for two telephone numbers, the user instead clicks in the input field and selects the previously typed telephone number by dragging the mouse cursor over the existing text in the input field. The user then types in the second number (thus replacing the previous selection in the input field) and clicks on the "Add" button. Finally, the user clicks on the "OK" button to submit both queries at once. In a similar manner, the user can issue queries for larger number of telephone numbers in a single dialogue.

### Design B: Pop-Up Menu

As previously mentioned, it can be assumed that the telephone number(s) in question is/are already on the screen.

To query for one number, the user moves the mouse cursor to the telephone number on the screen and presses down the mouse button. This causes a pop-up menu to appear over the number with one element for each database for which queries can be performed with telephone numbers as keys.

The user moves the mouse until the desired database is highlighted in the pop-up menu. The user then lets go of the mouse button. (The system knows which number to search on because the user pointed to it when calling up the menu).

To query for two numbers, the user repeats this entire interaction sequence for the second number. The second number was normally about five lines below the first number in the window. It is possible to submit the second query before seeing the result of the first one, and the result of the first query can be assumed to appear such that it does not overlap the telephone number needed for the second query.

## HEURISTIC ESTIMATES

Heuristic evaluation [9][13] is a usability inspection method [6] whereby a set of evaluators produces lists of usability problems in a user interface by going through it and noting deviations from accepted usability principles. As originally conceived, the heuristic evaluation method only involved the finding the usability problems and did not address other phases of the usability engineering lifecycle [8][11]. Subsequent work has shown how it can be extended to have the evaluators provide severity estimates for the usability problems after the lists of problems found by the individual evaluators have been aggregated [10][12].

Even severity estimates are still focused on usability problems (that is, characteristics of the interface) rather than overall user performance, even though they aim at assessing the impact of each usability problem on eventual user performance. We were therefore interested in assessing the extent to which heuristic evaluators could provide estimates of actual user performance.

Heuristic estimates were collected in three different conditions (cold, warm, and hot), varying in the extent to which the evaluators had access to actual running implementations of the interfaces. In all three conditions, evaluators were asked to estimate the time needed for expert users to issue queries on one as well as two telephone numbers with each of the two alternative interface designs. The evaluators were told to focus on expert user performance and disregard novice user performance as well as any learnability problems.

Common for all three conditions was that the evaluators had to estimate expert user performance based on their personal intuition and best guesses without having a chance to observe a real user or to perform any measurements. Of course, evaluators who had tried using one or both of the interfaces themselves could estimate their own performance and use it as a baseline for their projections of expert user performance, and several of them did so.

### Cold Estimates

"Cold" estimates were collected by sending each of 12 evaluators a written specification of the two interface designs, essentially corresponding to the descriptions above. The evaluators had not seen any running implementations of the designs. This condition corresponds to the case where a design team is considering two possible solutions, none of which has been implemented.

The mean time spent by the cold evaluators was 9.9 minutes. The total cost of collecting the cold estimates also includes the time needed to write up the specification, which took about two hours. Of course, there was a further cost associated with designing the two interfaces in the first place, but that cost should probably not be charged to the comparative evaluation study.

### Warm Estimates

The 10 evaluators in the "warm" condition were the same as those used for the original heuristic evaluation of the full system. They had spent about one hour each using a running prototype system implementing the Dialog Box interface as well as several other dialogue elements. After the heuristic evaluation, they were provided with a list of all the usability problems in the interface (both the ones they had identified themselves and the ones found by other evaluators) and were asked to rate the severity of each problem. Furthermore, they were provided with a written description of the Pop-Up Menu design as an alternative interface. This condition corresponds to the case where one design has been implemented and evaluated, and another is being considered based on the results of the evaluation.

There is no simple way to estimate the cost of collecting the warm estimates. Not only did the evaluators spend an hour each using the full system with the Dialog Box interface, but they also attended a one-hour briefing session explaining the full, complicated application and spent a further half hour estimating severities for the usability problems. Most of these activities were not strictly necessary for the purpose of estimating expert user performance, but they probably did provide the evaluators with additional insights into the issues surrounding the kind of dialogues they were later

asked to assess. We would guess that it would be possible to collect warm estimates in about 12.1 minutes per evaluator (the mean of the cold and hot times), if their only task was to estimate expert user performance and not to find usability problems in the interface. Furthermore, it would be necessary to have a research assistant available throughout the study to help the evaluators use the software.

### Hot Estimates

The 15 evaluators in the "hot" condition had used running versions of both interfaces before giving their estimates. In fact, they were given access to the same versions as those used for the user testing (described further below). For each interface, the evaluators were first given a demo of how to perform single and double number queries, and were then given a chance to use the interface themselves as long as they wished. An experimenter was present to help the evaluators in case they had problems using the interfaces. This condition corresponds to the case where two alternative applications are available for purchase, or where two alternative design solutions have both been prototyped.

The mean time spent by the hot evaluators was 14.2 minutes. This time included a few minutes to recover from system crashes in a few cases. It seems reasonable to include this time given that hot estimates require the use of running software and given that this software will often be in the early stages of development. Given the way we conducted the hot estimates, a research assistant was present during each session to help the estimators use the software, so there is a need for an additional 14.2 minutes of staff time per estimator. The time needed to develop the software implementing the two interfaces for the hot estimation was about 6 days. The same software was used for both the hot estimates and the user testing, but from a cost perspective, the development costs should be charged fully to both methods, since one would presumably use only one method for any given development project. It was also necessary to have a usability specialist spend one hour designing the study and the questionnaire given to the estimators.

### The Evaluators

For all three heuristic estimation conditions, the evaluators were usability specialists with an average of nine years of usability experience. Different evaluators were used in each condition. Previous studies have shown that usability specialists are better than non-usability specialists at finding usability problems through heuristic evaluation [9]. We would therefore expect non-usability specialists (e.g., developers) to provide somewhat worse estimates of user performance than those found in this study, but we do not have any data to support this conjecture.

It is important to note that the evaluators provided their user performance estimates independently of each other. In general, we recommend collecting data from heuristic evaluation without letting the evaluators talk together before they have provided their lists of usability problems and severity or performance estimates, in order to prevent any bias in the evaluations. It is obviously also possible to have heuristic evaluation performed by small teams working together [5].

## GOMS ANALYSIS

GOMS (Goals, Operators, Methods, and Selection rules) [2] is a method for the formal analysis of user interactions with computers. The method was explicitly designed for the purpose of estimating expert user performance disregarding learnability problems and user errors, so it seemed reasonable to use it for our study. Also, GOMS has proven quite popular in recent years in the formal modelling community [15], as evidenced by the large number of papers published about GOMS, extensions of GOMS, and applications of GOMS, again indicating that it would be a reasonable method for our study.

The GOMS analysis were performed by 19 upper-division undergraduate students in a human–computer interaction class as their second assignment using GOMS. The students were given the same specification of the two user interfaces as that used for the "cold" heuristic evaluators. They were asked to build GOMS models of the two interfaces based on the specification and to estimate the time needed for expert users to perform single and double telephone number queries. Just as for the heuristic estimates, each GOMS analysis was performed by an individual student working alone.

We should note that the quality of the GOMS models built by the students in this study was probably not as good as the quality of models that might be built by true GOMS experts with more extensive experience in the use of the method. On the other hand, even though better performance could probably be achieved by using GOMS experts, highly experienced GOMS experts are not readily available in most development organizations. If formal usability analysis becomes a successful approach in industry, it may be not be unrealistic to expect GOMS analyses to be performed by people with a medium level of expertise.

The mean time spent by each student to perform the GOMS analysis of the two interfaces was 108 minutes. The total cost of using the GOMS method would also include the time to produce the written specification (about two hours).

## USER TESTING

User testing was conducted with 20 test users who were students recruited from a local college or summer students involved in non-usability projects at Bellcore. All test users were required to be experienced mouse users, since previous experience shows that problems with mouse usage will otherwise dominate a user's performance.

The study design was a within-subjects experiment where all test users tried both interfaces such that half of the users (randomly assigned) used the Dialog Box interface first and the other half used the Pop-Up Menu interface first. The

| | N | Dialog Box 1 Query | Dialog Box 2 Queries | Pop-Up Menu 1 Query | Pop-Up Menu 2 Queries | St.Deviations as % of Means |
|---|---|---|---|---|---|---|
| Cold heuristic estimates | 12 | 20.3 (26.1) | 29.4 (30.4) | 8.0 (7.4) | 14.0 (15.1) | 108% |
| Warm heuristic estimates | 10 | 12.9 (10.7) | 21.4 (21.6) | 4.7 (2.7) | 9.1 (5.3) | 75% |
| Hot heuristic estimates | 15 | 13.8 (6.1) | 20.0 (9.5) | 6.1 (3.5) | 9.4 (5.5) | 52% |
| GOMS analyses | 19 | 16.6 (3.7) | 22.6 (5.4) | 5.8 (0.8) | 11.2 (1.9) | 19% |
| User testing | 20 | 15.4 (3.0) | 25.5 (4.7) | 4.3 (0.6) | 6.5 (0.9) | 17% |

***Table 1*** *Mean time estimates in seconds arrived at by different methods. Numbers in parentheses indicate the standard deviation of the estimates. The last column is averaged across the four data columns and indicates variability.*

within-subjects design was chosen since learning and transfer of learning was irrelevant for the study, whereas controlling for individual variability was important. Since learnability was not a concern in this study, the test users were instructed in the use of the interfaces and were given help as needed during their initial use. For each interface, the test users were first allowed to practice until they had reached a steady plateau of expert user performance, after which the measurement part of the experiment began. Users were presented with a sequence of randomly chosen tasks, randomly alternating one-, two-, and three-number queries (the latter were not used in the analysis). They were instructed to work as quickly as possible while avoiding errors. Users were given error messages whenever they made errors, and they were also shown a feedback curve tracking their transaction rate throughout the experiment. After their use of each interface, the users were given a two-question subjective satisfaction questionnaire to assess how well they liked the interface.

Erroneous transactions constituted 6.5% of the expert user mode transactions for the Dialog Box interface and 3.8% for the Pop-Up Menu interface. These errors were eliminated from the data before it was analyzed.

Each user test session (with two interfaces) took about two hours, involving both the test subject and an experimenter. The time needed to develop the software implementing the two interfaces for the user testing was about 6 days. There was also the need to have a usability specialist spend two hours designing the experiment as well as having a research assistant and a test user spend two hours on a pilot test.

*Subjective Satisfaction.* On a 1–5 scale, with 1 being best, users gave the dialog box interface a mean evaluation of 4.0 for pleasantness and 3.9 for efficiency. The pop-up menu interface was evaluated as 3.2 for pleasantness and 2.3 for efficiency. In other words, users strongly preferred the pop-up menu interface (the differences are statistically significant at $p<.01$ for both variables according to paired $t$-tests).

## COMPARING THE ESTIMATES

Table 1 shows the mean estimates for the four task-by-interface combinations for each of the five studies. A striking result is the extent to which all the estimation methods overestimate the time needed for a two-number query in the pop-up menu interface. Given the way the interface is defined, a two-number query is exactly identical to two single-number queries, so it may be natural for the estimators to more or less double their estimate from the one-number query. In actual use, however, repeating exactly the same sequence of actions a second time is faster than doing it the first time since some mental steps are eliminated* the second time around, so the ratio between two and one queries was measured as 1.5. Even though this phenomenon is known in the GOMS literature [14], the students doing the GOMS analyses were not familiar with this refinement of the model, and they arrived at a ratio of 1.9 between two- and one-number queries. The fact that extensive knowledge of modifications like this is needed for a complete GOMS analysis is one reason it is seen as intimidating by some developers [1]. The heuristic estimators apparently did not consider the faster execution of the repeated operations either, except for the ones in the hot condition who had tried the interface themselves and estimated a ratio of 1.6 between two- and one-number queries. The cold estimators' ratio was 1.8, and the warm estimators' was 1.9.

The user test results in Table 1 may be slightly misleading with respect to true user performance in the field. Expect field users may be faster at using the dialog box interface because they typically develop extremely fast keyboarding skills. Even though we trained our users to expert performance on the interface as such, we could obviously not train their keyboarding skills to the level of an experienced operator. However, reducing the user test times for the dialog box interface by one second per telephone number to be entered only reduces the relative advantage of pop-up menus from 74% to 72%. Reducing the time to type in each telephone number by two seconds still only reduces the rel-

* Potentially, a redesigned pop-up interface might be faster for repeated queries if some of the physical operators were eliminated: Making the database from the previous query the default menu choice would allow repeated queries simply by clicking the mouse on the telephone number, thus avoiding the need to select a menu option. We were not interested in iterative design for the purpose of the current study, however.

| | Absolute Advantage of Pop-Up Interface | | | | Relative Advantage of Pop-Up Interface | | | |
|---|---|---|---|---|---|---|---|---|
| | Estimate of absolute number of seconds saved | $\frac{Max}{Min}$ ratio | $\frac{Q_3}{Q_1}$ ratio | **N** to get standard error of 10% and 20% | Estimate of relative proportion of time saved | $\frac{Max}{Min}$ ratio | $\frac{Q_3}{Q_1}$ ratio | **N** to get standard error of 10% and 20% |
| Cold heuristic estimates | 27.6 (126%) | 18.6 | 2.9 | 159; 40 | 52% (26%) | 4.3 | 2.9 | 7; 2 |
| Warm heuristic estimates | 20.6 (146%) | 21.0 | 3.5 | 213; 53 | 51% (34%) | 2.7 | 1.7 | 12; 3 |
| Hot heuristic estimates | 18.1 (57%) | 7.8 | 3.0 | 32; 8 | 53% (27%) | 3.4 | 1.4 | 7; 2 |
| GOMS analyses | 11.1 (41%) | 5.3 | 1.9 | 17; 4 | 55% (21%) | 2.6 | 1.4 | 4; 1 |
| User testing | 15.0 (22%) | 2.5 | 1.3 | 4; 1 | 74% (4%) | 1.1 | 1.1 | 1; 1 |

***Table 2*** *Mean estimates of the advantage of pop-up menus in absolute terms (seconds saved per transaction) and relative terms (how large a proportion of the time that could be saved by going from the dialog box interface to the pop-up menu interface). Both estimates assume an even mix of one- and two-number queries. Numbers in parentheses after the value estimates indicate the standard deviation of the estimates as a proportion of the mean value.*

ative advantage of pop-up menus to 69%, thus confirming the general result of the user test. We will therefore use the user test results as the best indicator of "true" user performance, keeping in mind that field studies would probably show a slightly smaller advantage for the pop-up menus.

Table 2 shows the estimates of the absolute and relative advantage of the pop-up menu interface, assuming an even mix of one- and two-number queries. The table also shows three ways of assessing the spread of the estimates: standard deviation (in parentheses after the means), ratio between the largest and smallest estimate, and ratio between the third and first quartile (top 25% vs. bottom 25%). Estimates of relative performance are much tighter than are estimates of absolute performance, indicating a higher degree of agreement on the relative estimates. Thus, even though the estimators and analysts may disagree on exactly how fast users would be on the two interfaces, they agree somewhat on the proportional difference between the interfaces. Similarly, the standard deviation for the user test data is much smaller for the relative comparison than for the absolute comparison between the interfaces, basically taking advantage of the fact that individual differences are smoothed over by the within-subjects design of the experiment.

Since relative usability estimates have a smaller spread than absolute usability estimates, we would recommend relying on relative estimates whenever possible. If absolute estimates are needed, a larger number of estimates must be gathered to achieve a sufficiently small measurement error. The standard error can normally be estimated as the standard deviation divided by the square root of the number of observations. Using this estimator, Table 2 also shows the number of estimators/analysts/test users needed to achieve a standard error of 10% as well as 20% of the mean value for absolute and relative estimates of usability. It may not always be necessary to aim for a standard error of 10% of the mean estimate. The values listed for a standard error of 20% of the mean estimate thus provide a reasonable indication of the minimum number of estimates to acquire.

Experienced usability specialists might have been expected to provide better estimates than less experienced usability specialists. This is not the case, however, given two simplistic measures of usability expertise. The heuristic estimators were asked to state their usability experience in number of years (ranging from 1 to 30 years, mean=9, SD=6). They were also asked to state their experience with graphical user interfaces on a 1–3 scale (little, some, or extensive), with nobody giving their GUI experience as "little." Regressions between the experience measures and measures of estimation accuracy were all statistically insignificant, having very low correlation coefficients (typically R=.1). This is not to say that one might not be able to get better at performance estimates with some forms of experience, but just that other factors might be at play than simply the person's number of years of experience in the usability field in general. It is an interesting issue how to improve the estimation abilities of usability specialists as well as of regular developers. We believe that an important element in the acquisition of such skills is the observation of multiple user tests where real users use a variety of interface designs for a variety of tasks. We only have anecdotal evidence to support this conjecture, however. One supporting observation is the fact that our best heuristic estimate was provided by a usability specialist from a major computer vendor who had recently completed a set of empirical studies of input techniques.

## COST-BENEFIT ANALYSIS

A cost-benefit analysis is very difficult to perform for the problem discussed in this paper as there is no way to quantify the value of having estimates of a certain accuracy.

Assuming that the estimates were to be used as the basis for making a decision on what interface to implement in a

development project, the benefit would be the increased probability of making the correct decision. To get a rough idea of the benefits, let us assume that the time to implement the dialog box interface in the full system would be 200 hours and that the time to implement the pop-up menu interface in the full system would be 800 hours. Assuming the loaded cost of a software professional to be $100 per hour, the cost of the dialog box interface is $20,000 and the cost of the pop-up menu interface is $80,000. Let us furthermore assume that the system will have 3,000 users when it is released and that they will spend 2% of their working days performing queries, evenly divided among one- and two-element queries. These assumptions are reasonable given the underlying application that gave rise to this study.

Based on the above assumptions, a total of 3000 (users) × 236 (working days per year) × 0.02 (time doing queries) = 14,160 person-days per year would be spent issuing queries with the dialog box interface. Then the user test data indicates that the users would save 74% of this time, or about 10,480 person-days per year by using the pop-up menu interface, corresponding to $1,048,000 if a user costs $100 per day. Further assuming that the system release is slightly more than a year in the future, that the system is to be used for two years before an update can potentially be made to correct any design errors, and that the savings are to be discounted by 10% per year, gives a value of $870,000 for the first year's savings and $790,000 for the second year's saving. Thus, the present value of choosing the pop-up menu interface is $1,660,000. Deducting the additional implementation costs result in a final benefit of making the correct choice (the pop-up menu interface) of $1,600,000.

Assume for the sake of argument that a software development manager is only willing to approve a more expensive user interface element if its expected benefits are at least ten times larger than the expected incremental cost of implementing it. Event though the exact factor can be discussed, it would certainly be reasonable of the project manager to insist on a large number, since the implementation costs are certain (except for the ever-present risk of a cost overrun) and are to be spent here and now out of the manager's budget, whereas the reduced user costs are projected and will be accrued by the user organization over time. This assumption means that the software manager will make the correct decision whenever the pop-up menu design is estimated to save at least $600,000 (corresponding to a relative advantage of 30% for the pop-up menus).

Under these assumptions, the software manager would make the correct decision every time when relying on estimates from user testing of even a single test user, as well as when relying on any single warm heuristic estimate or any single GOMS analysis. For the cold heuristic estimates, the software manager would make the correct decision 92% of the time when relying on a single estimate, 98% of the time when relying on the mean of two estimates, and 100% of the time when relying on the mean of three estimates. For the hot heuristic estimates, the manager would make the correct decision 93% of the time when relying on a single estimate, and 100% of the time when using the mean of two.

| | *Cost formula (U.S. dollars)* | *Cost of a good relative estimate* |
|---|---|---|
| Cold heuristic estimates | 200 + 17N | $319 |
| Warm heuristic estimates | 800 + 38N | $1,256 |
| Hot heuristic estimates | 1300 + 42N | $1,594 |
| GOMS analyses | 200 + 90N | $560 |
| User testing | 1470 + 83N | $1,553 |

***Table 3*** *Cost estimates for the five methods. The last column indicates the cost to achieve a relative estimate with a standard error of 10% by setting N (the number of estimators/analysis/subjects) equal to the numbers listed in Table 2.*

The cost estimates in Table 3 have been arrived at under the following estimates for the loaded hourly cost of various types of staff: Usability specialist: $100. Research assistant: $25. Test user (student): $10. The cost estimates include fixed costs to set up the studies as indicated in the description of the procedure earlier in this paper as well as variable costs for each estimator/analyst/subject. Furthermore, the cost formulas for warm and hot heuristic estimates as well as user testing include half an hour of research assistant time per person for the scheduling of appointments. Cold heuristic estimates and GOMS analyses can be handled simply by mailing out the written interface specification to people who can deal with it at their own convenience. The GOMS analysis cost formula reflects only half of the time actually spent in the study, since we would assume that experienced analysts would be faster than the students used by us. We see that cold heuristic estimation is the cheapest method (but as indicated by Table 1 and Table 2 also the worst), and that hot heuristic estimation is the most expensive, being 5.0 times as expensive. User testing is almost as expensive as hot heuristic evaluation, being 4.9 times as expensive as cold heuristic evaluation.

Combining our estimates of the costs and the benefits, we find that the benefits are between 1,000 and 5,000 times greater than the costs, confirming the value of usability engineering, no matter what method is used.

## CONCLUSIONS

We have compared various ways of arriving at estimates of user performance with interface alternatives. User testing still seems to be the best method for arriving at such estimates, but one should remember that laboratory testing is not always a perfect predictor of field performance. User testing was also much more expensive than "cold" heuristic estimates and somewhat more expensive than GOMS analy-

ses, both based on interface specification. The main reason for the added expense of user testing was the implementation cost needed to get a running prototype of sufficient stability to allow for user testing.

Heuristic estimates were better in the hot condition where estimators had access to running versions of the two interfaces, than in the cold condition based on specifications only. Since hot heuristic estimation was slightly more expensive than user testing, one might as well perform user testing if running interfaces are available. Heuristic estimation may have advantages in cases where the running software is too unstable for use in a user test or in cases where only one of the interface alternatives has been implemented.

In general, estimates of the relative advantage of one interface over the other were much better than estimates of the absolute time needed by users to perform various tasks. GOMS analyses and heuristic estimation were about equal for relative estimates, so heuristic estimation might be recommended based on its lower costs. GOMS analyses were superior for absolute estimates, however.

As a final conclusion, the results from this study show that performance estimates from both heuristic estimation and GOMS analyses are highly variable, with the absolute estimates from heuristic estimators being extremely variable. It is therefore highly recommended not to rely on estimates from a single usability specialist. Instead, one should gather estimates from several specialists, use the mean value, and keep in mind that the true value is likely to be somewhat different. Even so, performance estimates provide reasonable ball-park values that can be used as the basis for cost-benefit decision-making in a usability engineering process, and they would have led to the correct decision 100% of the time in the case study presented here, assuming that at least three independent estimates were collected.

It would be interesting to consider semi-formal estimation methods that might combine the best of the GOMS and heuristic approaches for some kind of "back of the envelope" type estimation procedures. It is likely that one of the reasons for the lower variability of the GOMS estimates was the use of a single, specified procedure for arriving at the estimates, and a less formal procedure might still be of some help at getting better estimates.

This study has focused on expert user performance, even though other usability parameters such as learnability are more important for the success of many user interfaces. In principle, one can use most of the same methods (except GOMS) to estimate system learnability. Unfortunately, there is no way of extrapolating the relative accuracy of the methods for this purpose from the current study.

## Acknowledgments

The authors would like to thank Erik Nilsen of Lewis and Clark College for collecting data from his students on their GOMS analyses of the two interfaces. We also thank Professor Nilsen's students for participating in the study. The analyses and conclusions based on this data in this paper is solely the responsibility of the authors of the present paper, and does not necessarily correspond to the positions of study participants. The authors would like to thank Rob Fish, Erik Nilsen, and the anonymous referees for helpful comments on previous versions of the manuscript.

## References

1. 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.
2. Card, S.K., Moran, T.P., and Newell, A. (1983). *The Psychology of Human–Computer Interaction.* Erlbaum Associates, Hillsdale, NJ.
3. 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.
4. 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.
5. Karat, C., Campbell, R., and 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.
6. Mack, R.L., and Nielsen, J. (1993). Usability inspection methods. *ACM SIGCHI Bulletin* **25**, 1 (January).
7. 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.
8. Nielsen, J. (1992a). The usability engineering life cycle. *IEEE Computer* **25**, 3 (March), 12–22.
9. Nielsen, J. (1992b). Finding usability problems through heuristic evaluation. *Proc. ACM CHI'92* (Monterey, CA, 3–7 May), 373–380.
10. Nielsen, J. (1992c). Reliability of severity estimates for usability problems found by heuristic evaluation. In *Digest of Posters and Short Talks, ACM CHI'92 Conference* (Monterey, CA, 3–7 May), 129–130.
11. Nielsen, J. (1993a). *Usability Engineering*, Academic Press, San Diego, CA.
12. Nielsen, J. (1993b). Heuristic evaluation. In Nielsen. J., and Mack, R.L. (Eds.), *Usability Inspection Methods,* book under preparation.
13. Nielsen, J., and Molich, R. (1990). Heuristic evaluation of user interfaces. *Proc. ACM CHI'90* (Seattle, WA, 1–5 April), 249–256.
14. Olson, J.R., and Nilsen, E. (1988). Analysis of the cognition involved in spreadsheet software interaction. *Human–Computer Interaction* **3**, 4, 309–349.
15. Olson, J.R., and Olson, G.M. (1990). The growth of cognitive modeling in human–computer interaction since GOMS. *Human–Computer Interaction* **5**, 2&3, 221–265.