# A Validation of the GOMS Model Methodology in the Development of a Specialized, Commercial Software Application

*Richard Gong*
Center for Ergonomics
The University of Michigan
1205 Beal Ave.
Ann Arbor, MI 48109-2117
rich.gong@um.cc.umich.edu

*David Kieras*
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, MI 48109-2110
kieras@eecs.umich.edu

**ABSTRACT**

A formal GOMS model approach was applied to the design and evaluation of the user interface for a specialized, commercial software application. This approach was able to identify significant usability problems embedded in the procedures by which users interact with the interface. A redesign of the interface based on the GOMS approach resulted in a 46% reduction in learning time and a 39% reduction in execution time during a formal evaluation, differences predicted by the GOMS analysis. Corrections to the GOMS time estimation techniques were necessary to obtain accurate (within 9%) predictions of absolute learning and execution times.

**KEYWORDS**: GOMS, analytical methods, interface design, usability, user testing, performance prediction

## INTRODUCTION

In spite of the increasing attention given to user interface design by software development organizations, it remains a difficult problem to successfully design and evaluate a new user interface. Perhaps the most widely cited set of usability design principles are those proposed nearly a decade ago by Gould and Lewis [8], who urged interface designers to focus early and continually on users, to conduct empirical testing of proposed interfaces with these users, and to iteratively design the interface based on the results of testing. In response to these admonitions, some software developers now claim to perform at least some iteration of the user interface based on user focus and testing [15].

While there are well known cases of successful design utilizing Gould and Lewis' usability principles (e.g., [7]), there continues to be a strong intuition within the HCI community of the need for analytical approaches to interface design and evaluation [3, 12], which at the very least, could work in tandem with empirical approaches in the iterative design loop, and at the most provide satisfactory design results when empirical testing is not practical.

## THE GOMS MODEL

The GOMS model concept, originally introduced by Card, Moran and Newell [4], is one of the most widely accepted analytical modeling concepts in the HCI community. The GOMS model describes the procedural knowledge required by a user to routinely interact with a software interface in terms of the user's goals, the operators or actions involved, the methods required to accomplish the user's goals, and the rules for selecting between competing methods. Kieras [13] described a methodology for a simplified type of GOMS analysis, based on the Kieras and Polson cognitive complexity work [2, 14] and using a relatively formal notation for GOMS models, NGOMSL. The goal of this GOMS methodology is to make GOMS modeling convenient for actual software development. Recent experience with NGOMSL (e.g., [5] and [20]) and other forms of GOMS modeling [9, 17] suggests that GOMS models can in fact be constructed and used effectively to design and evaluate software interfaces. However, to date there are no systematically documented and reported cases in which the NGOMSL methodology has been applied in an actual software development situation, with the costs and payoffs determined. The purpose of this paper is to provide such a case. An additional result of the evaluation is some refinements and corrections to the NGOMSL methodology.

## SPECIFIC DESIGN PROBLEM

The interface design problem concerns a specialized CAD program for the field of Industrial Ergonomics. This application requires the user to input a variety parameters to describe a manual materials-handling situation (such as lifting crates, pushing carts, etc.). The software then displays measures of the physical stress imposed by the job.

This application, known as the Three Dimensional Static Strength Prediction Program™ (3D for short) [21] has been commercially available for DOS computers through the University of Michigan for approximately 3 years, and has a registered user base of around 200 users.

The design problem of interest was to create a Macintosh version of the software. While porting an application to a different platform is hardly unusual, far fewer resources were available for this project than in most commercial software development enterprises. The entire responsibility for developing, coding, and evaluating the design of the new version was to fall upon a single person, the first author of this paper.

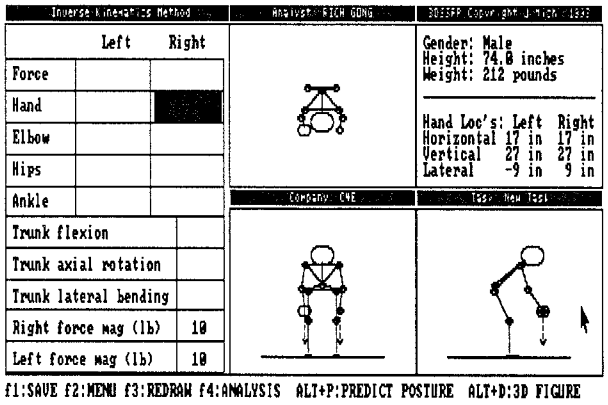
## DEVELOPMENT OF THE MACINTOSH INTERFACE

Following the advice of Gould and Lewis, potential users were consulted at the earliest stages of this project. The few available domain experts who were experienced Macintosh users were asked as to what specific interface behavior and features they would like to see in a Macintosh version. Beyond the expectation of duplicated functionality, few other concrete recommendations were expressed.

A working Macintosh interface was produced by transferring the essential organization and content of the user interface from the DOS application to the new Macintosh interface. Care was taken to adhere to Apple's style guide [1] while preserving the essential character of the commercially viable interface. Figures 1 and 2 show the main display of the commercial DOS application and the Macintosh interface; not shown are several additional windows and their associated controls.

Ideally, representative users would be recruited at this stage of the project to work through carefully selected task scenarios using the new interface while usability problems would be carefully monitored and recorded. The results from such an evaluation could then be used to modify the interface, and the representative users could then be called upon to re-evaluate the interface [8].

However, the actual circumstances of this development effort were far from ideal; the few available "representative users" of such specialized software (faculty, researchers, and advanced graduate students) were unwilling to commit the necessary time for participating in iterative cycles of user testing and redesign. Most of these users did agree, however, to informally evaluate the Macintosh interface on their own schedule over a three week period. Usability problems, interface feature requests, and bugs would be documented on standard forms by these users. While such a procedure would hardly be considered representative of a full empirical usability evaluation of an interface, we suspect that the circumstances which led to such a procedure are not uncommon in the development of specialized software.


Inverse Kinematics Method
Analyst: RICH GONG
3DSSPP Copyright U Mich [illegible]
Company: C4E
Task: New Task

| | Left | Right |
|---|---|---|
| Force | | |
| Hand | | ▇ |
| Elbow | | |
| Hips | | |
| Ankle | | |

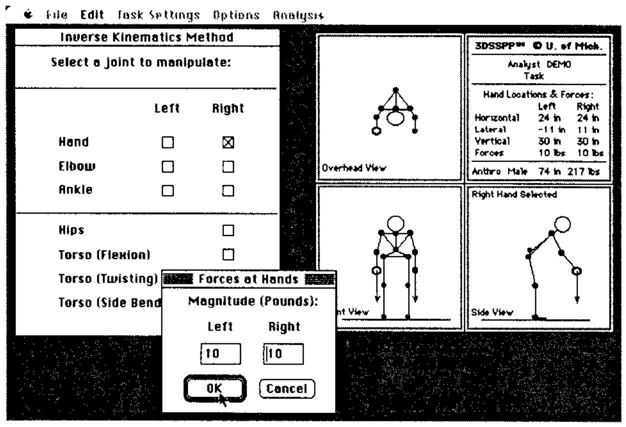
| | |
|---|---|
| Trunk flexion | |
| Trunk axial rotation | |
| Trunk lateral bending | |
| Right force mag (lb) | 10 |
| Left force mag (lb) | 10 |

```
Gender: Male
Height: 74.0 inches
Weight: 212 pounds

Hand Loc's: Left  Right
Horizontal 17 in  17 in
Vertical   27 in  27 in
Lateral    -9 in   9 in
```

```
f1:SAVE f2:MENU f3:REDRAW f4:ANALYSIS  ALT+P:PREDICT POSTURE  ALT+D:3D FIGURE
```

**Figure 1.** *User interface for DOS version*



Select a joint to manipulate:

| | Left | Right |
|---|---|---|
| Hand | ☐ | ☒ |
| Elbow | ☐ | ☐ |
| Ankle | ☐ | ☐ |
| Hips | | ☐ |
| Torso (Flexion) | | ☐ |
| Torso (Twisting) | | |
| Torso (Side Bend | | |

```
3DSSPP™ © U. of Mich.
Analyst DEMO
Task
Hand Locations & Forces:
             Left    Right
Horizontal   24 in   24 in
Lateral      -11 in  11 in
Vertical     30 in   30 in
Forces       10 lbs  10 lbs
Anthro Male  74 in   217 lbs
```

Magnitude (Pounds):

| Left | Right |
|---|---|
| 10 | 10 |

**Figure 2.** *Original Macintosh interface*

As might be expected, such an informal procedure yielded less than satisfactory results. While several usability problems were identified by the users, there was little consensus as to the severity of any single problem. At this point, a non-empirical approach to evaluating and redesigning the interface was sought.

## DEVELOPMENT OF THE REVISED INTERFACE

While one of the currently popular non-empirical approaches (such as Heuristic Evaluation [16]) might be a logical choice for a non-empirical approach, this project seemed to provide a unique opportunity to apply a formal GOMS model approach that is rarely attempted in practice, at least to the extent that would be required by this project. The formal GOMS model approach was thus chosen to aid in the evaluation and redesign of the Macintosh interface. The GOMS model evaluation of the interface was then conducted entirely by the designer and first author of this paper. The designer had previously worked with GOMS models [6] and consulted the guidelines in Kieras [13] where necessary.

The first step of the methodology is to identify users' high level goals. In this project, the designer identified these goals through contact with users during two separate training seminars for users of the commercial DOS version of the software. The designer then constructed a GOMS model of the original Macintosh interface. The GOMS model was then examined for usability problems according to the suggestions found in [4] and [13], which help to identify characteristics of interface methods which impede routine execution or ease of learning. Table 1 summarizes, at a high level, the nature of the usability problems identified in this analysis. Many of the problems contained multiple characteristics (see the example described below), but the table classifies each problem by its most salient characteristic.

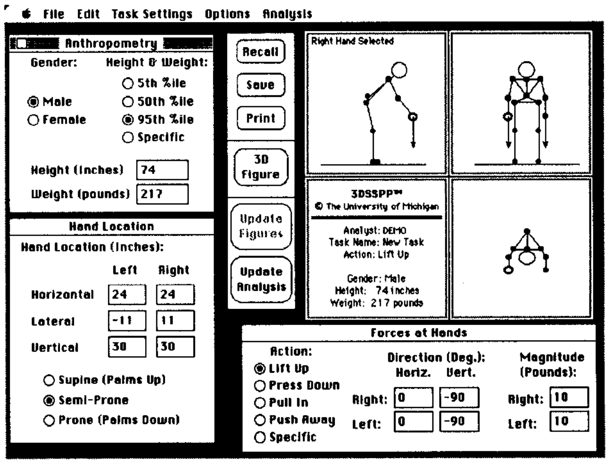
**Table 1.** *Usability problems identified by analysis of GOMS model*

| Problem characteristic identified by GOMS | Number |
|---|---|
| Frequent goal supported by inefficient method | 4 |
| Frequent goal not supported by any method | 1 |
| Similar goals supported by inconsistent methods | 2 |
| Method which relies heavily on long-term memory | 2 |
| Method which relies heavily on working memory | 2 |
| Method with multiple mouse and hand moves | 1 |

Twelve major usability problems were identified. A design solution was drafted for each problem based on the specific details provided by the GOMS model analysis. For example, most of the frequently accessed dialog windows were modal windows, a characteristic carried over from the DOS version. The GOMS analysis revealed that when performing a routine analysis, users must execute a lengthy series of steps to retrieve and dismiss these dialogs. The analysis further revealed that these steps are fraught with time-consuming actions such as mouse moves to access menu items, hand exchanges between the keyboard and mouse, and the retention of information from dismissed dialogues in short term memory.

The solution to this multi-dimensional problem was guided by the problem characteristics identified. In the revised design of the interface, all of the frequently used dialogs became "modeless" dialogs, reducing the number of steps needed to complete a typical analysis by almost half, eliminating over 50% of the mouse moves in the model, and eliminating 90% of the information that the GOMS model described as having to be kept in the user's memory. An obvious potential tradeoff not addressed by the GOMS analysis is that such a redesign increases the density of the display space.

The redesigned interface is shown in Figure 3.



Gender: [✓ Male] Female

Height & Weight: 5th %ile 50th %ile [✓ 95th %ile] Specific

Height (inches) 74

Weight (pounds) 217

Hand Location

Hand Location (inches):

| | Left | Right |
|---|---|---|
| Horizontal | 24 | 24 |
| Lateral | -11 | 11 |
| Vertical | 30 | 30 |

Supine (Palms Up) [✓ Semi-Prone] Prone (Palms Down)

Analyst: DEMO
Task Name: New Task
Action: Lift Up

Gender: Male
Height: 74 inches
Weight: 217 pounds

Forces at Hands

Action: [✓ Lift Up] Press Down Pull In Push Away Specific

| | Direction (Deg.): Horiz. | Vert. | | Magnitude (Pounds): |
|---|---|---|---|---|
| Right: | 0 | -90 | Right: | 10 |
| Left: | 0 | -90 | Left: | 10 |

**Figure 3.** *Redesigned Macintosh interface*

## EMPIRICAL COMPARISON OF THE INTERFACES

While the GOMS evaluation resulted in what appeared to be a better interface, two research questions remained: Did the changes implemented as a result of applying the GOMS model methodology really improve the usability of the software interface? And if so, can a GOMS model predict the extent of the improvement? A formal experiment was thus planned to compare the two interfaces and provide the data to test the GOMS model predictions.

### Method

*Design.* Subjects recruited for the experiment were randomly assigned to one of two groups and were asked to complete a representative set of tasks using either the original or revised Macintosh version of the software. These conditions will be hereafter referred to as the "Original" and "Revised" conditions. Each group used only one of the two interfaces. To assess the subjective quality of "usability", two objective metrics were used, the metrics of learning time and practiced, errorless, execution time. Thus, if the usability of the software had been improved through using the GOMS methodology, there should be a decrease in the learning and execution times for users of the revised version compared to the users of the original version.

*Subjects.* Since the software assumes domain knowledge, subject selection and training were critical. Recall that one of the motivations for adopting a formal GOMS model method was the lack of domain experts who were willing to participate in multiple evaluations in an iterative design scheme. Therefore, subject recruitment was limited to undergraduate and graduate students in the Industrial Engineering Department of the University of Michigan who had completed at least an introductory course in Industrial Ergonomics. While training in the domain would still be significant, it would be greatly reduced by placing such a restriction on potential subjects. Twenty-one subjects participated in the experiment, with 10 assigned to the Original condition and 11 assigned to the Revised condition. All subjects were compensated for their time.

*Tasks.* A set of 15 representative analysis tasks were chosen from training and reference materials. Each task differed slightly in the specific parameters supplied by the users, such as worker size, type of exertion performed, and postural description. One task was used for a demonstration task, 7 of the tasks were used for practice, and the remaining 7 for actual data collection.

*Procedure.* Precise execution times were captured through the insertion of program code into each interface which time-stamped and recorded all user actions and system events in log files.

After satisfactorily completing a short training session in the requisite aspects of Industrial Ergonomics, subjects in each condition completed the demonstration task along with the set of seven practice tasks. Next they completed the set of seven analysis tasks and then repeated the same seven tasks in order to produce routine and practiced performance.

Task execution times were then extracted from the time-stamped log files. This analysis included only the times for the second execution of the analysis tasks, corresponding to practiced performance. Errors were rare in this portion of the data and usually consisted of slips in key presses or mouse button clicks. The time users spent committing or recovering from these errors was identified in the log files and subtracted from the execution times.

## Results

*Observed execution times.* Table 2 shows the average time to complete the entire set of seven tasks. For practiced, errorless execution, users of the Revised interface completed the set of analysis tasks an average of 39% faster than users of the Original interface, which was highly statistically significant. The differences in execution time for each task was statistically significant. There was also a significant main effect of task, but no interface by task interaction. There were no significant differences in the quality (accuracy) of the ergonomic analysis completed by the subjects in each condition.

**Table 2.** *Mean observed execution times*

| Task | Original interface (seconds) | Revised interface (seconds) | Percent change |
|---|---|---|---|
| 1 | 71.0 | 40.6 | -43 |
| 2 | 82.6 | 53.3 | -35 |
| 3 | 70.8 | 40.1 | -43 |
| 4 | 55.3 | 36.4 | -34 |
| 5 | 57.4 | 39.6 | -31 |
| 6 | 82.4 | 51.1 | -38 |
| 7 | 74.1 | 41.6 | -44 |
| Mean | 70.5 | 43.2 | -39 |

*Predicted execution times.* Would the GOMS model have been able to predict the observed reductions in execution time? To answer this question, a GOMS model was constructed of the redesigned Macintosh interface. Predictions of task execution for each analysis task were then developed for each interface from the GOMS model using the method outlined by Kieras [13], which assigns time values to the various operators and method steps, and sums these values to produce a total time estimate.

Table 3 summarizes the predicted execution times, and the predicted percent improvement for the Revised interface over the Original interface. A comparison of Table 3 to Table 2 reveals that the predicted mean improvement of 40% closely matches the observed mean improvement of 39%. However, it can also be seen that absolute execution times are badly overpredicted (by a factor of more than two) consistently across the tasks.

**Table 3.** *Predicted execution times*

| Task | Original interface (seconds) | Revised interface (seconds) | Percent change |
|---|---|---|---|
| 1 | 152.5 | 94.5 | -38 |
| 2 | 199.5 | 111.5 | -44 |
| 3 | 185.2 | 102.6 | -45 |
| 4 | 130.6 | 87.4 | -33 |
| 5 | 139.8 | 97.8 | -30 |
| 6 | 191.0 | 108.6 | -43 |
| 7 | 180.5 | 105.7 | -41 |
| Mean | 168.4 | 101.2 | -40 |

*Learning times.* Predicted learning times above a baseline for each condition were determined by multiplying the number of new NGOMSL statements by a coefficient of 30 seconds per new statement, and long-term memory associations by a coefficient of 10 seconds per new long-term memory association. These values are recommended by Kieras [13] and are based on earlier studies of learning and transfer [2, 14]. The baseline is defined as the time required to work through a task when no new NGOMSL statements or long-term memory associations have to be learned. In a manner analogous to one of the earlier studies [2], the total training time for each condition was determined by summing times required by subjects to work through the demonstration task and the set of 7 practice tasks. The predicted and observed training times are summarized in Table 4.

**Table 4.** *Predicted vs. observed learning times*

| Condition | Predicted time over baseline (sec) | Observed time (sec) |
|---|---|---|
| Original | 1670 | 1601 |
| Revised | 900 | 871 |
| Difference | 770 | 730 |
| Improvement | 46% | 46% |

Although the observed differences in learning time between conditions were reasonably well predicted, the predictions of the absolute training times are close to the observed times only if the baseline time is assumed to be zero. However, such an assumption is unrealistic; even if no new learning takes place, a task will always require time for method execution.

### Effort required

An important result from this study is the effort expended by the designer when using the GOMS approach. Table 5 summarizes the days spent in each phase of the interface development. The table shows that for this particular design project, the time required for the GOMS method was reasonably small relative to the time required to code and debug both the original and revised interfaces.

**Table 5.** *Effort required for project activities*

| Project phase | Type of design activity | Days |
|---|---|---|
| Original interface development | Conduct informal user survey | 6 |
| | Draft design objectives | 3 |
| | Code original Mac interface | 52 |
| GOMS analysis | Construct GOMS model | 4 |
| | Evaluate GOMS model | 11 |
| Revised interface | Draft design solutions | 8 |
| | Code interface revisions | 17 |
| Formal evaluation | Recruit and train subjects | 10 |
| | Run full formal evaluation | 18 |

While there is a common belief that a formal GOMS analysis is too laborious to be practical for real software development, the actual experience of this project appears to negate such a notion. The GOMS analysis required only 12% of the total project time and required only a fifth of the actual interface coding time. The GOMS analysis also required less than half the time as the informal user survey and formal testing combined.

## REFINING GOMS MODEL PREDICTIONS

### Refining execution time prediction techniques

While the design changes motivated by the GOMS analysis resulted in a significant reduction in practiced execution times (the relative amount of which could have been accurately predicted), the inability of the GOMS model to reliably predict absolute execution times was disturbing. However, the uniformity of the overprediction across interfaces and tasks suggests that there is a systematic cause to the overprediction.

One possibility is that GOMS models constructed by the interface designer inaccurately describe the methods used to accomplish goals. However, a careful examination of the log files revealed that the sequence of user actions conformed almost exactly to the sequence of steps in the GOMS models.

A second possibility is that the time values used for some of the GOMS model parameters are incorrect. The values used to develop the original predictions were recommended by Card, et al. [4] and Kieras [13]. Other empirical studies have found slight deviations from these recommendations for many of the parameters [18]. Analysis of the data from this experiment revealed that the values used for keystrokes, hand movements, and mouse-button clicks were reasonably accurate, but that the mouse move value, recommended as an average of 1.1 seconds, was consistently inaccurate. However, applying Fitts' law to the actual mouse moves in the interface results in the much smaller and more accurate values of 0.1 to 0.5 seconds.

A third possibility is that the serial execution assumption of Kieras' form of the GOMS model may be incorrect. Researchers who have been modeling tasks in other domains have found that some processes appear to occur in parallel, as postulated in the Model Human Processor [4, 9, 11]. The log files clearly suggest that for practiced execution, certain physical operators are executed in parallel with certain mental and perceptual operators required by Kieras' [13] methodology. This seems to hold in three common, highly practiced situations: The overlapped execution of the perceptual operator "locate" with a subsequent mouse move to a menu, the overlapped execution of the mental operator "verify selection" with the next mouse move or keystroke, and the execution of keystrokes and mouse moves by users while the computer display is being updated. The execution times of these pairs of steps was more accurately estimated by dropping the mental or perceptual step and simply using the physical operator in the remaining step.

Table 6 summarizes all of the corrections to the prediction rules described above.

**Table 6.** *Corrected rules for developing execution time predictions from NGOMSL methodology*

| |
|---|
| • Use mouse movement times calculated from Fitts' Law where possible. |
| • If a locate is followed by a mouse movement, and this is a well-practiced sequence, assign zero time for the locate. |
| • If a verification step is required, but is part of a well-practiced sequence, assign zero time to the verification (see [2]). |
| • If the display changes in response to a user input, and both the preceding and following user input are part of a well-practiced sequence, then assign zero time for the user waiting for the display to change. |

If execution time predictions are calculated using these corrections, the overprediction factor for each interface drops to an average of 9%. Figure 4 shows the observed and predicted execution times for both Macintosh interfaces using the corrected prediction rules. The predictions developed from these rules are reasonably accurate and usable for design purposes.

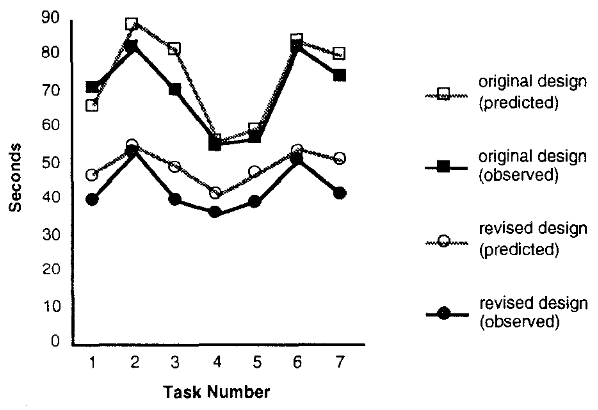


**Figure 4.** *Corrected predicted vs. observed times for the Original and Revised Macintosh interfaces*

### Revising learning time prediction coefficients

The predicted learning times compare favorably with the observed data only if the baseline is assumed to be zero. Therefore, new coefficients which reflect the conditions of learning and practice of this study are needed. If the baseline time is estimated using the practiced execution time of analogous tasks and subtracted from the total training time, then the remaining time for the demonstration and set of 7 practiced tasks can be fitted to the following revised coefficients: 17 seconds per new NGOMSL statement and 6 seconds per new long-term memory association. These coefficients reflect the conditions of learning and practice used in this study, which are more realistic than the experimental conditions of the earlier studies of learning and transfer [2, 14].

## CONCLUSIONS

This design project and the subsequent empirical evaluation demonstrate that an analytical modeling methodology such as the GOMS model method can and should have a role in the iterative cycle of software interface design and evaluation. This may be especially true when resources do not permit iterative design based on extensive user testing, or when the application domain is so specialized that the availability of users who can reliably test the software with realistic tasks is minimal. The time required to apply the GOMS methodology is reasonable, given the overall effort to initially design and modify an implementation.

Why use the GOMS model methodology as opposed to other non-empirical methodologies such as Heuristic Evaluation [16] or the Cognitive Walkthrough method [19]? The strengths of these approaches have been documented in case studies of their application (e.g., [10]). Nonetheless, based on the present study and the previously cited experiences (e.g., [17]), the GOMS model methodology should also be considered as an equally valid non-empirical methodology. Not only is it based on a widely accepted theoretical construct, but it can be applied, with a reasonable amount of effort, identify significant usability bottlenecks, provide guidance for design solutions, and produce useful quantitative predictions for design alternatives.

Because the informal evaluation approach initially applied in this project was not representative of a full empirical usability evaluation, the authors hesitate to draw any strong conclusions from this phase of the project. The purpose of this project was *not* to directly compare different approaches, but to demonstrate that a formal GOMS model approach can indeed provide useful design guidance for a realistic design problem. Designers seeking a non-empirical approach when circumstances prevent a full empirical evaluation should perhaps consider the GOMS model method as a viable approach.

If a GOMS model approach is to be applied to the development of alternative interface designs, how can usable engineering predictions of learning and execution be obtained? Based on the results of this study, it appears that the recipe suggested by Kieras [13] is a good starting point, but several modifications to the recipe are required to produce reliable predictions of absolute performance. For predictions of execution time, mouse movement times should be obtained using the appropriate Fitts' law equation, if the actual required movements can be determined. Mental and perceptual operators for fixed and frequent aspects of the interface can be assumed to overlap with the subsequent physical operator. To predict learning time for certain training situations, the revised coefficients for the number of new NGOMSL statements and long-term memory associations should be used. Baseline times, if needed, can be estimated from the practiced execution times of analogous tasks. Further research is required to verify all of these revised guidelines.

## REFERENCES

1. Apple Computer, Inc. (1987). *Human interface guidelines.* Reading, MA: Addison-Wesley.
2. Bovair, S, Kieras, D. E., and Polson, P. G. (1990). The acquisition and performance of text editing skill: A cognitive complexity analysis. *Human-Computer Interaction,* 5(1), pp. 1-48.
3. Butler, K. A., Bennett, J., Polson, P., and Karat, J. (1989). Report on the workshop on analytical models: Predicting the complexity of human-computer interaction. *SIGCHI Bulletin,* 20 (4), pp. 63-79.
4. Card, S., Moran, T., and Newell, A. (1983). *The psychology of human-computer interaction.* Hillsdale, NJ: Lawrence Erlbaum Associates.
5. Endestad, T., and Meyer, P. (1993). GOMS analysis as an evaluation tool in process control. Technical Report HWR-349, OECD Halden Reactor Project, Instituut for Energiteknikk, Halden, Norway.
6. Gong, R., and Elkerton, J. E. (1990). Designing minimal documentation using a GOMS model: A usability evaluation. *Proceedings of CHI '90,* pp. 99-106. New York: ACM.
7. Gould, J. D. , Boies, S. J., Levy, S., Richards, J. T., and Schoonard, J. (1987). The 1984 Olympic Message System: A test of behavioral principles of system design. *Communications of the ACM,* 30, pp. 758-759.
8. Gould, J. D., and Lewis, C. H. (1983). Designing for usability — Key principles and what designers think. *Proceedings of CHI '83,* pp. 50-53. New York: ACM.
9. Gray, W. D., John, B. E., & Atwood, M. (1992). The precis of Project Ernestine, or an overview of a validation of GOMS. *Proceedings of CHI'92,* pp. 307-312. New York: ACM.
10. Jeffries, R., Miller, J. R., Wharton, C., and Uyeda, K. (1991). User interface evaluation in the real world: A comparison of four techniques. *Proceedings of CHI '91,* pp. 119-124. New York: ACM.
11. John, B. E. (1988). *Contributions to engineering models of human-computer interaction.* Doctoral dissertation, Carnegie Mellon University, Pittsburgh, PA.
12. Karat, J., and Bennett, J. (1989). Modeling the user interaction methods imposed by designs. Technical report RC 14649. IBM T. J. Watson Research Center, Yorktown Hts., NY.
13. Kieras, D. E. (1988). Towards a practical GOMS model methodology for user interface design. In M. Helander (Eds.), *Handbook of human-computer interaction,* pp. 135-57. New York: North-Holland.
14. Kieras, D. E., and Polson, P. G. (1985). An approach to the formal analysis of user complexity. *International Journal of Man-Machine Studies,* 22(4), pp. 365-394.
15. Mulligan, R. M., Dieli, M., Nielsen, J., Poltrock, S., Rosenberg D., and Rudman S. E. (1992). Designing usable systems under real world constraints: A practitioner's forum. *Proceedings of CHI '92,* pp. 149-152. New York: ACM.
16. Nielsen, J. (1992). Finding usability problems through heuristic evaluation. *Proceedings of CHI '92,* pp. 373-80. New York: ACM.
17. Nielsen, J. and Phillips, V. L. (1993). Estimating the relative usability of two interfaces: Heuristic, formal and empirical methods compared. *Proceedings of INTERCHI '93,* pp. 214-221. New York: ACM.
18. Olson, J. R., and Olson, G. M. (1990). The growth of cognitive modeling in human-computer interaction since GOMS. *Human-computer interaction,* 5(1), pp. 221-265.
19. Polson, P. G., Lewis, C., Rieman, J., and Wharton, C. (1992). Cognitive walkthroughs: A method for theory-based evaluation of user interfaces. *International Journal of Man-Machine Studies,* 36(5), pp. 741-73.
20. Steinberg, L. S., & Gitomer, D. H. (1992). Cognitive task analysis, interface design, and technical troubleshooting. *Proceedings of the 1993 International Workshop on Intelligent User Interfaces,* pp. 185-191. New York: ACM.
21. The University of Michigan (1990). *The Three Dimensional Static Strength Prediction Program™.* Ann Arbor: The University of Michigan.