Why GO

Bonnie E. John

In 1983 Stu Card, Tom Moran, and Allen Newell presented the concept of "GOMS" in their seminal book, The Psychology of Human-Computer Interaction. They suggested that, when designing a computer system, it is useful to analyze knowledge about how to perform a task in terms of four components: **Goals, Operators, Methods,** *and* **Selection rules**. *At the time of their proposal, I was a systems engineer at Bell Laboratories writing specifications for the functionality and user interface of small-business telecommunication systems. If I had heard this claim then, a raft of questions would have leapt to my mind: What is GOMS? What can it do? In what situations does it apply? How do you use it? Who can use it? Can it be trusted? Is it worth the bother of using it?*

A dozen years later, these same questions still exist for many people, for usability specialists who have not yet used GOMS, for their managers, and for computer system designers. After leaving Bell Laboratories to get a Ph.D. in cognitive psychology/human-computer interaction, I've spent the last twelve years using GOMS: reading about it, exercising it, and doing research to push the envelope of what it can do. This time around, in addition to asking the questions, I may be able to provide some of the answers.

MS?

GOMS is a method for describing a task and the user's knowledge of how to perform the task in terms of goals, operators, methods, and selection rules. Goals are simply the user's goals, as defined in layman's language. What does he or she want to accomplish by using the software? In the next day, the next few minutes, the next few seconds?

Operators are the actions that the software allows the user to take. With the original command-line interfaces, an operator was a command and its parameters, typed on a keyboard. Today, with graphic user interfaces, operators are just as likely to be menu selections, button presses, or direct-manipulation actions. In the future, operators will be gestures, spoken commands, or even eye movements. Operators can actually be defined at many different levels of abstraction but most GOMS models define them at a concrete level, like button presses and menu selections.

Methods are well-learned sequences of subgoals and operators that can accomplish a goal. The classic example describes deleting a paragraph in a text editor. Using a mouse, place the cursor at the beginning of the paragraph, hold the mouse button down, drag to the end of the paragraph, release, highlighting the paragraph, then hit the delete key. This complete sequence of actions is one "method." Another (less efficient) method: place the cursor at the end of the paragraph and hit the delete key until the paragraph is gone.

If there is more than one method to accomplish the same goal, then selection rules, the last component of the GOMS model, are required. Selection rules are the personal rules that users follow in deciding what method to use in a particular circumstance. For instance, in the paragraph-deletion example, if the paragraph is more than five characters long, then I will typically drag to highlight and hit the delete key. If the deletion is five characters or less, then I will place the cursor at the end and delete back. Thus, my personal selection rule depends on how long the paragraph is. Another user may have a different selection rule that depends on a different length of paragraph or even depends on other features of the task situation.

There are actually at least four different versions of GOMS in use today: the original formulation proposed by Card, Moran and Newell: and a simplified version they called the Keystroke-Level Model (KLM); a more rigorously defined version called NGOMSL proposed by David Kieras; and a parallel-activity version which I developed called CPM-GOMS. The original version was a loosely defined demonstration of how to express a goal hierarchy, methods and operators, and how to formulate selection rules. The KLM uses only keystroke-level operators, no goals, methods or selection rules. The analyst simply lists the keystrokes and mouse-movements a user must perform to accomplish a task an then uses a few simple heuristics to place "mental operators." NGOMSL presents a very well defined procedure for identifying all the GOMS components, expressed in a form similar to an ordinary computer programming language, and rules-of-thumb about how many steps can be in a method, how goals are set and teminated, and what information needs to be remembered by the user while doing the task. CPM-GOMS uses *cognitive*, *percpetual* and *motor* operators in a *critical path method* schedule chart (PERT chart) to show how activities can be performed in parallel. All of these techniques, however, are based on the same GOMS concept.

What Can GOMS Do?

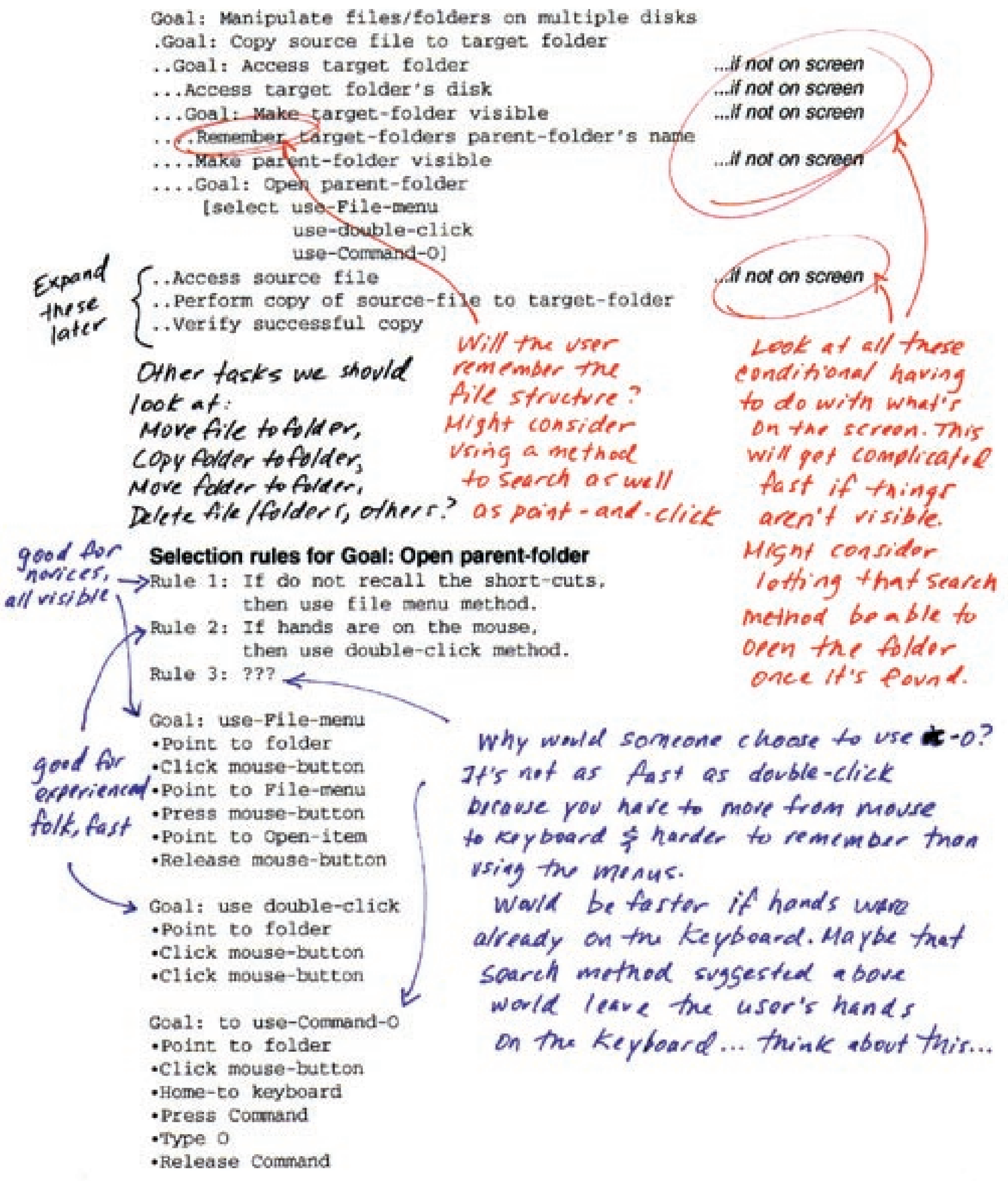
A GOMS analysis produces quantitative and qualitative predictions of how people will use a proposed system. If you are a consumer of computer systems, these predictions allow you to evaluate alternative systems before buying one. If you are a designer of computer systems, these predictions allow you to evaulate rival design ideas at the specification stage of design, even before a protoytpe is built. GOMS analyses also help focus design effort on those parts of a system that will cause users the most problems, and suggest better designs.

To make quantitative evaluations of systems or design ideas, GOMS can be used to predict execution time of tasks that skilled users are likely to perform. If a length of time can be estimated for each one of the operators described in a GOMS model, then all the times can be added up to get a prediction of performance time. Such predictions also work with a interface where people are performing several tasks at once. In a catalogue-sales interface, for instance, the sales person using the interface may also be talking on the phone with a customer. The sales person is talking and listening to the customer to give and get information, while at the same time he or she may be typing or using a mouse, waiting for information from the screen, and visually searching the screen. GOMS can predict the length of time to perform well-learned tasks that involve both sequential and parallel operators.

GOMS can also be used to predict how long it will take to learn to perform a task. Predictions can be made for the time needed to learn a set of methods by assigning an estimate of the time to learn each of the steps in the process. GOMS assumes that the user already knows how to perform the operators in isolation (e.g., that they know how to type or use a mouse), but that they do not know how to combine them into efficient methods or what selection rules might be useful to distinguish between methods. GOMS can predict this "method-learning" time, which is indeed the time most users require to learn a new application on their current workstation.

There is also some research showing how GOMS can be used to predict the frequency of errors that arise from forgetting. How much an interface requires the user to remember when performing a task will affect how often an error may be made and in what particular places errors may be more frequent.

These, then, are the types of predictions that GOMS can make: skilled-performance time, method-learning time, and the likelihood of memory errors, all before a system is bought or



```
Goal: Manipulate files/folders on multiple disks
.Goal: Copy source file to target folder
..Goal: Access target folder                          ...if not on screen
...Access target folder's disk                        ...if not on screen
...Goal: Make target-folder visible                   ...if not on screen
....Remember target-folders parent-folder's name
....Make parent-folder visible                        ...if not on screen
....Goal: Open parent-folder
     [select use-File-menu
             use-double-click
             use-Command-O]
..Access source file                                  ...if not on screen
..Perform copy of source-file to target-folder
..Verify successful copy
```

Expand these later

Other tasks we should look at:
Move file to folder,
Copy folder to folder,
Move folder to folder,
Delete file/folders, others?

Will the user remember the file structure? Might consider using a method to search as well as point-and-click

Look at all these conditional having to do with what's on the screen. This will get complicated fast if things aren't visible. Might consider letting that search method be able to open the folder once it's found.

Selection rules for Goal: Open parent-folder

```
Rule 1: If do not recall the short-cuts,
        then use file menu method.
Rule 2: If hands are on the mouse,
        then use double-click method.
Rule 3: ???
```

good for novices, all visible

good for experienced folk, fast

```
Goal: use-File-menu
•Point to folder
•Click mouse-button
•Point to File-menu
•Press mouse-button
•Point to Open-item
•Release mouse-button

Goal: use double-click
•Point to folder
•Click mouse-button
•Click mouse-button

Goal: to use-Command-O
•Point to folder
•Click mouse-button
•Home-to keyboard
•Press Command
•Type O
•Release Command
```

Why would someone choose to use ⌘-O? It's not as fast as double-click because you have to move from mouse to keyboard & harder to remember than using the menus.
Would be faster if hands were already on the keyboard. Maybe that search method suggested above would leave the user's hands on the keyboard... think about this...

built. These quantitative predictions can be used to calculate the human costs in a system: the costs to train people, the costs involved with using a system day in and day out, and the costs from possible errors and error-recovery. These cost-estimates can be examined in light of other costs of a system: capital costs to buy a new system, development costs to re-design and implement an old one, manufacturing and maintenance costs. Thus, the quantitative nature of GOMS predictions speaks the same language—dollars—that is common to everyone designing and buying hardware and software.

In What Situations Does GOMS Apply?

GOMS analysis applies to situations in which users will be expected to perform tasks that they have already mastered. In the psychology literature this is called having a cognitive skill, i.e., users are not problem solving, not hunting around for what they need to do next. They know what to do and all they have to do is act. There are many different types of cognitive skill in human-computer interaction. For instance, there are many single-user applications where the user tells the system what to do, then the system does it and tells the user what it has done. This is a user-paced, passive system and GOMS has been shown to work very well in this type of situation. GOMS has been applied to applications such as text and graphics editors, page layout, spreadsheets, information browsers, operating systems, ergonomic design systems, CAD systems, map digitizers, flight-management computers in commercial airplanes, oscilloscopes, programable television sets, and WWW pages. As you can see, there are many, many real-life situations in which GOMS modeling is applicable.

GOMS has also been shown to be valid in single-user, active systems, where the system changes in unexpected ways. There are GOMS models, for instance, of radar monitoring and of video games, where the system throws new situations at the user at a maniacal pace. The knowledge gathered by a GOMS analysis is sufficient to predict what a person will do in these seemingly unpredictable situations.

GOMS can also describe the behavior of "surrogate users." The catalogue-sales situation mentioned above is one example of a surrogate user. The user of the interface is the employee on the telephone trying to help customers, but the overall task is really defined by the customer's needs. The employee, helping the customer through the entire task, is called the surrogate user. Extensive GOMS modeling in telephone-operator work has predicted quite precisely how customers and telephone operators work together to complete tasks.

GOMS, then, applies to tasks involving cognitive skill. This may initially appear like a limited field of applications that does not include creative, nonrepetitive tasks. But as the examples above illustrate, GOMS analysis goes well beyond the marked-up-manuscript text-editing for which it was developed. Any time an interface interferes with accomplishing a task, including very creative tasks, GOMS can be brought into play. For example, in the intellectually creative tasks of programming or visual design, a poorly designed window manager hinders creative work by being inefficient or intruding with system-imposed procedures that seem unnecessary to the actual task. GOMS can be useful in examining the routine portion of a creative task, allowing the designer to minimize the troublesome parts of the interface so the user can perform the real task instead of "interacting with the computer." Most creative tasks, from writing papers, to architectural design, to planning complex surgical operations, have some routine aspects of the work that could benefit from GOMS analysis.

How Do You Use GOMS?

GOMS can be used both quantitatively and qualitatively. Quantitatively, it gives good predictions of performance time and learning time. If, for instance, you have to choose between two systems, say you must decide whether your company should buy this or that package of office automation tools, you can apply a GOMS model. To do this, build a GOMS model of Application A and a GOMS model of Application B, and examine the quantitative predictions. Perhaps Application A has a lower up-front capital cost, but will be slower to perform frequent tasks. Perhaps Application B will be faster to perform tasks, but has a longer learning time. With these quantitative predictions, you can examine such trade-offs in the light of what is important to your company,

About the Author
Bonnie E. John
Human-Computer Interaction Institute and Departments of Computer Science and Psychology, Carnegie Mellon University
bej@cs.cmu.edu

The following references are a few classic, and some recent, works that span different uses of the different versions of GOMS.

Reference	Version of GOMS	Type of System	Predictions	Uses of Analyses	Task Domain
Original version of GOMS					
Card S., Moran, T., and Newell, A. (1983). The psychology of human–computer interaction. Hillsdale, NJ: Lawrence Erlbaum Associates.	Original GOMS	single-user/ passive	operator sequence, performance time	method verification	text editor, operating system, VLSI CAD
Lerch, F.J., Mantel, M.M., & Olson, J.R. (1989). Translating ideas into actions: Cognitive analysis of errors in spreadsheet formulas. In Proceedings of CHI, 1989, 121 - 126. New York: ACM.	Original GOMS	single-user/ passive	working memory errors	comparison of systems, method verification	spreadsheet
Ekenon, J. & Palmiter, S.L. (1991). Designing help using a GOMS model: An information retrieval evaluation. Human Factors,33, 2, pp. 185 - 204.	Original GOMS	single-user/passive	learning time	documentation design, method verification	HyperCard ™
John, B.E., Vera, A.H., & Newell, A. (1994). Toward real-time GOMS: A model of expert behavior in a highly interactive task. Behavior and Information Technology, vol 13, no. 4, pp.255 - 267.	Original GOMS	single-user/ active	operator sequence	method verification	videogame
Keystroke Level Model					
Card, S., Moran, T., and Newell, A. (1983). The psychology of human-computer interaction. Hillsdale, NJ: Lawrence Erlbaum Associates.	KLM	single-user/ passive	performance time	method verification	text editor
Lane, D.M., Napier, H.A., Batsell, R.R., & Naman, J.L. (1993). Predicting the skilled use of hierarchical menus with the Keystroke-Level Model. Human - computer Interaction, 8, 2, pp. 185 - 192.	KLM	single-user/ passive	performance time	method verification	spreadsheet
Haunold, P. & Kuhn, W. (1994). A keystroke level analysis of a graphics application: Manual map digitizing. In Proceedings of CHI, 1994 (Boston, MA, USA, April 24 - 28, 1994). New York: ACM, pp. 337 - 343.	KLM	single-user/ passive	performance time	redesign justification	map digitizer
NGOMSL					
Kieras, E.E. (1988). Towards a practical GOMS model methodology for user interface design. In M. Helander (Ed.). The handbook of human - computer interaction (pp.135 - 138). Amsterdam: North-Holland.	NGOMSL	single-user/ passive	performance time, learning time, working memory errors	method explanation	text editor
Bovair, S., Kieras, D.E., & Polson, P.G. (1990). The acquisition and performance of text editing skill. A cognitive complexity analysis. Human–Computer Interaction, 5, 1 - 48.	NGOMSL	single-user/ passive	learning time, performance time	method verification	text editor
Gong, R. & Kieras, D. (1994). A Validation of the GOMS Model Methodology in the Development of a Specialized, Commercial Software Application. In Proceedings of CHI, 1994 (Boston, MA, USA, April 24 - 28, 1994). New York: ACM, pp.351 - 357.	NGOMSL	single-user/ passive	operator sequence, performance time, learning time	redesign, method verification	ergonomic CAD
CPM-GOMS					
Gray, W.D., John, B.E., & Atwood, M.E. (1993). ``Project Ernestine: A validation of GOMS for prediction and explanation of real-world task performance." Human–Computer Interaction, 8, 3, pp. 207 - 209.	CPM-GOMS	surrogate user	performance time	comparison of systems, method verification	telephone operator workstation
Chuah, M.C., John, B.E., & Pane, J. (1994). Analyzing Graphic and Textual Layouts with GCMS: Results of a Preliminary Analysis in Proceedings Companion of CHI, 1994 (Boston, MA, USA, April 24 - 28, 1994). New York: ACM, pp. 323 - 324.	CPM-GOMS	single-user/ passive	performance time	comparison of systems, method verification	airline schedule
GOMS Tools					
Kieras, D.E., Wood, S.D., Abotel, K., & Hornof, A. (1995). GLEAN: A Computer-Based Tool for Rapid GOMS Model Usability Evaluation of User Interface Designs. To appear in UIST95 Proceedings.	NGOMSL	single-user/ passive	performance time, learning time, working memory errors	redesign, method verification	ergonomic CAD

GOMS Reviews (all versions of the technique)

Olson, J.R. & Olson, G.M. (1990). The growth of cognitive modeling in human—computer interaction since GOMS. Human—Computer Interaction, 5, 1221 - 265.
John, B.E. & Kieras, D.E. (in preparation). ``The GOMS-family of Analysis Techniques: Tools for Design and Evaluation."
A more extensive technical work reviewing GOMs research andpractice, with complete references, can be gotten from the author. Send a request to bej@cs.cmu.edu.

Jakob Nielsen on Using a Version of the Keystroke-Level Model

In an interview with interactions, Jakob Nielsen, presently a Distinguished Engineer at Sun Microsystems and formerly at Bellcore, discussed how he used a GOMS model to analyze a financial implication, in terms of users' time, of adding buttons to a graphic interface. Sun was working on a new World Wide Web site and needed to make decisions about whether or not to add new buttons to the Home Page, and if so how many.

Nielsen's specific goal was, in fact, to justify why it might not be sound to add new buttons. In his case, he needed to be able to explain the expense involved when a user has an option, but elects not to use it. What does it cost in time and money for a user to scan a screen, to see a button, and then decide not to use the button? This cost then has to be weighed against the benefits of adding that particular button. In the Sun Web site, of three potential buttons to be added, two were eventually removed and only one remained in the final design.

Choosing it for its simplicity, Nielsen used a customized version of the Keystroke-Level Model of GOMS to estimate the time it takes to scan, see, and reject a button. Then he multiplied this estimated time by estimated users in a year and estimated value of their time to derive the estimated cost per year.

Simplicity was important to Nielsen for several reasons. He had no time to actually test users; he needed to be able to justify his concern about unneeded buttons for a meeting held less than 24 hours later. But even if he had been able to attain a very accurate number by actual testing of the time required for the user to scan, see, and reject, Nielsen knew the precision would be wasted because of the level of uncertainty of the other numbers he would use to complete his final estimate of cost. He was using educated guesswork to estimate the number of users per year and the estimated value of their time, both of which are prone to wide variations. For instance, will the design only last a year or will it last a significantly longer or shorter time, impacting the estimated number of users connecting before the redesign? Who are the users and why are they using the site? They could range between what Nielsen calls "tourists" brousing online in their spare time to highly paid professionals for whom every minute counts. Confronted with the task, then, of needing to very quickly establish a ballpark estimate, Nielsen concluded a GOMS analysis of estimated performance time was the only realistic approach.

Nielsen also needed to have a a clear and simple explanation of how he reached his conclusions. Although expert in many other areas, the colleagues to whom he would present his justification were not necessarily HCI experts or cognitive scientists. GOMS helped him explain his analysis without detailed technical discussion. Nielsen also noted that his use of GOMS analysis, which is outside his own specialization in discount usability techniques, added third-party credibility to his presentation.

Nielsen points out the downside of his quick use of a modified GOMS Keystroke-Level Model and rough estimates of users and the value of their time; his estimate of a half million dollars to reject a Web-site button is only ballpark and certainly not statistically significant. But the upside underscores the importance of flexibility: "We got the project done, and while it was still relevant."

Nielsen believes that this need for flexibility—particularly human flexibility— will continue to be the key component in interface design and usability in the future. Some tools will be automated, like GOMS or other emerging analysis methods, but new ideas and applications occur so quickly in the field of HCI that these tools will probably only address a small subset of the problems that will surface. In order to enhance flexibility (which Nielsen notes is the distinguishing characteristic of the experienced professional versus the more rigid novice), HCI will not only need new software. Usability labs and support facilities also will need to be firmly in place so that ideas can be tested and analyses performed quickly, helping give HCI professionals as sophisticated a toolbox as possible. Nielsen emphasizes that "servicing is extremely important. If you have to start negotiating every time you need something, you can no longer focus on the problem at hand."

and what is relevant to your user-group or task situation. This is exactly how NYNEX arrived at a choice of telephone-operator workstations (documented by Wayne Gray, myself, and Mike Atwood in our 1993 paper).

GOMS also can be used quantitatively to evaluate design ideas, not just existing systems. You do not need to have a running system to use a GOMS model so you can use it at very early stages in the design process. For instance, you may be considering a re-design of an existing system and perhaps want to ask, "What if we changed it to do this?" If a system is already in use, it is usually very costly to replace it with something else. The development costs are almost always substantial, as is the retraining time, and both may be prohibitive. To justify redesign and transfer to a new system, it is important to be able to estimate the costs involved with training in the short run and the performance savings in the long run. If you have a GOMS model of the existing system it is very easy to change the model to reflect a design idea and quickly obtain quantitative predictions on just how much learning and performance time the idea either saves or costs. These quantitative predictions can then be used to justify or reject the cost of redesign. Peter Haunold and Warner Kuhn's 1994 justification of the cost of redesigning a system for digitizing maps is an excellent example of this process.

Qualitatively, GOMS also can be used to design training programs and help systems. The GOMS model is a careful description of the knowledge needed to perform a given task and thus it describes the content of task-oriented documentation. You only need to tell the new user what the goals are, what different methods could be used to achieve them, and when to use each method (selection rules). This approach has been shown to be an efficient way to organize help systems, tutorials, and training programs as well as user documentation.

GOMS models can also be used qualitatively to literally redesign a system. Richard Gong gives us a clear example of using GOMS directly for design. His design task involved an ergonmic CAD system that analyzed the physical aspects of a job to see if there might be injury to the worker. For instance, if workers on an assembly line must lift car batteries to install them, then this CAD system defines at what heights the conveyers and the cars should be positioned so that workers can move from place to place without incurring back injuries. The CAD system was implemented on one platform and was being ported to a Macintosh.

Gong performed a GOMS analysis in support of the Macintosh port. He found that the GOMS model showed where the interface needed to be re-designed, as well as providing evaluation of design ideas. For instance, when GOMS revealed a frequent goal supported by a very inefficient method, then Gong changed the method to a more efficient one. If GOMS showed that there were goals not supported by any method at all, then new methods were added. GOMS also revealed where similar goals are supported by inconsistent methods, a situation in which users are likely to have problems remembering what to do, and showed how to make the methods consistent.

GOMS analysis also show how some methods rely very heavily on the user's prior knowledge. In these cases, the designer then needs to judge whether typical users of the system will know in advance what is needed or whether the interface should be changed to provide more guidance to the users. GOMS also highlights methods that force the user to remember something very specific to a particular interaction. For instance, an insurance system might require that a long claim number be read off one screen and then entered into a field on another screen. Again, this is a situation in which people are apt to have difficulty and can be easily eliminated by redesign once an analysis identifies the problem. A designer can handle most of these difficulties if he or she knows where they are. Thus, GOMS models contribute directly to design by first identifying particular problems, and then allowing very rapid evaluation of new ideas.

GOMS, then, can be used to evaluate systems or design ideas, focus design effort, suggest redesigns, and even structure user documentation. Most uses of GOMS reported in the HCI literature are about a full GOMS analysis done on an entire system (e.g., Gong's work), but this is only because these uses are described in primarily research papers. GOMS can also be used to analyze a few particular aspects of a system as required by the design situation. For instance, Jakob Nielsen tells us how he used GOMS to answer questions about buttons on a World Wide Web page (see sidebar). Likewise, an analyst might decide to use GOMS on the half-dozen most common tasks performed on an application, to optimize performance of those specific tasks, to minimize their learning time, and to write tutorial materials. GOMS is certainly not an all-or-nothing tool; it can be used as needed, and in conjunction with other usability methods.

Who Can Use GOMS?

The examples above demonstrate several uses of GOMS, but what type of training is necessary to understand and use it? Actually, the answer to this depends on the version of GOMS you choose and the design questions you need to answer. People have learned to do NGOMSL and CPM-GOMS analyses from one-day CHI tutorials or industrial short-courses. Several academic GOMS researchers have successfully taught the whole family of techniques in a few lectures to CS undergraduates in introductory HCI classes. Other people suceesfully learn and use GOMS models from the published literature. For instance, Haunold and Kuhn, who are civil engineers, provide a good example of learning and using the Keystroke-Level Model in their work on digitizing maps. They are not psychologists. They needed a particular solution to a potentially inefficient aspect of their digitizing task, so they used GOMS models because they found them in the literature, they were able use them, and there was a pay off. Likewise, Jakob Nielsen showed that undergraduates taking their first course in HCI were able to predict the time needed to execute a database look-up using the Keystroke-Level Model just as well as usability experts who had an average of nine-years' professional experience. On the other extreme Sharon Irving, Peter Polson, and J. E. Irving required a deep psychology background to help interpret the analysis when they used GOMS models on a flight-control computer. The ways in which people are using GOMS to analyze problems and obtain information are clearly very diverse. There is no one right way that must be rigorously adhered to and a spectrum of backgrounds and procedures can be found in the literature.

Can GOMS Be Trusted?

The answer to this question is "yes"! In the last decade, HCI researchers have very carefully tested and re-tested the predictions of GOMS models, and reported these results in refereed conferences and journals, so designers can trust the method. Many studies give rigorous laboratory verification of the predictions made from GOMS models on a large number of products. There have been several studies that use real-world data to verify performance-time predictions of GOMS models. There has also been work with realistic training situations that show the value of GOMS-inspired training programs and help systems.*

Thus, for those aspects of tasks that involve cognitive skill, the validity of GOMS is no longer a question. Designers can use the method with confidence that it will produce valid predictions of skilled performance time and learning time that no longer need to be tested against empirical data.

Is GOMS Worth the Bother?

Why bother learning a new analysis technique like GOMS? Why not just run laboratory usability studies or use some of the new "discount methods" like Heuristic Evaluation or Cognitive Walkthrough? There are several reasons to "bother."

Sometimes it is very important to know how a system will be used in the long run with very skilled people. GOMS models allow these kinds of predictions to be made whereas many empirical methods do not (e.g., the typical one-hour, think-aloud usability study in a lab). Using GOMS, designers and users can make quantitative predictions about skilled behavior without having to train people, sometimes for months, to get them up to highly skilled performance. Often this predictive approach is the only one that is really viable.

Sometimes it is economically important to

*** There has been much less work with the prediction of errors, so I recommend that designers without formal psychological education not use GOMS models to predict errors until the validation research matures.]**

get quantitative predictions like the ones GOMS can provide. Haunold and Kuhn's map-digitizing situation demonstrates this point. They were able to predict a 73% reduction in performance time of a routine task that translated into a cost savings of $730,000. NYNEX's telephone-operator workstation is an another example in which every second counts because of the sheer scale of people doing the same tasks so many times each day. Again, GOMS models predicted a cost increase of about $2 million per year had NYNEX chosen to buy the less efficient workstation. Thus, it is of particular economic importance in some situations to have good predictions, and GOMS can provide them whereas laboratory usability studies or discount methods cannot.

GOMS has also been shown to find usability problems not found through normal development or other means of analysis. The goals and methods definitions in GOMS require the analyst to think about the structure of the task as the user sees it. Consistencies and inconsistencies emerge that are often not seen when other analysis techniques or program-development tools are used that do not focus on the task structure. In addition, the GOMS focus on task structure and user's knowledge provides a basis for documentation and tutorial design. Observation of users may help debug existing documentation but does not provide guidance for the overall design, and other analysis techniques in the literature have not yet been used to improve documentation.

Finally, when you ask, "Why should I bother?", you need to know how much of a bother using GOMS actually is. There is growing evidence that using GOMS is neither overwhelmingly difficult nor time consuming. For instance, Gong kept diaries of working times, starting with the original interface development where he conducted informal user surveys, drafted design objectives, and coded the interface. He clocked constructing and evaluating the GOMS model, revising the interface, and doing a formal usability evaluation in which trained subjects were recruited and a full, formal evaluation was run. Doing the GOMS analysis took 12% of the total project time and required only a fifth of the actual interface coding time. These times are relatively small, especially since the interface was improved so that learning time decreased almost 50% and skilled exectution-time decreased 40%. In addition, constructing and using GOMS analysis took only half of the time of doing the user-surveys and formal user-study. Applying the GOMS analysis, then, was well worth the "bother." Also, computer-based tools are being developed for GOMS analysis that promise to make constructing models easier, to automate the more tedious aspects, and to make playing "what-if" with them especially easy.

Conclusion

In conclusion, I would like to ask, and answer, one more question. Given all this evidence that GOMS is a useful approach to usability analysis, then "Why *NOT* GOMS?" Is it a panacea, a silver bullet? Of course not. There are good reasons not to use GOMS in many design situations. If you are designing a system that will only have novice users, like a kiosk at the Olympics, then predictions of skilled behavior are not necessary. Better to run a few representative users through a laboratory usability test or use Cognitive Walkthrough. (But look carefully: Coming up with the action sequences that form the basis for a Cognitive Walkthrough is almost the same as doing a Keystroke-Level Model.) If you are designing a system that has no predecessor, you may not know why or how people will ultimately use it. In this case, predicting what task goals people may have could be pure speculation and GOMS models might bear no resemblance to actual behavior when the system is in use. Finally, if you are designing a leading-edge technology system where the operators are not yet well-understood, like gesture-recognition or some forms of virtual reality, much of the learning and performance will be in these new, unpredictable perceptual/motor operators. In that case, the methods-learning GOMS can predict would not be a major component of learning time.

Certainly, GOMS is *not* the right usability analysis technique for *every* design situation. Just as certainly, however, GOMS *is* the right usability analysis technique for *many* design situations.

"Why GOMS?" was originally presented at CHI '94, Boston, MA, April 24 - 28th as the discussion of a paper session.