Cognitive Modeling for Human-Computer Interaction[1]

Bonnie E. John
Human-Computer Interaction Institute
and Departments of Computer Science and Psychology
Carnegie Mellon University
Pittsburgh, PA, USA, 15213

Phone: 412-268-7182 Fax: 412-268-1266
e-mail: bej@cs.cmu.edu

Abstract

The field of Human-Computer Interaction (HCI), whose goal is to make computers support human activity in much more satisfying ways than they currently do, has three main uses for cognitive modeling. A cognitive model can substitute for a human user to predict how users will perform on a system before it is implemented or even prototyped. A system can generate a cognitive model of the user currently interacting with the system in order to modify the interaction to better serve that user. Finally, cognitive models can substitute directly for people so groups of individuals can be simulated in situations that require many participants, e.g., for training or entertainment. This paper presents some instances of such models and the implications for GI design.

Le domaine de l'Interaction Homme-Machine, dont le but est d'améliorer le soutien que portent les ordinateurs aux activités humaines, peut utiliser la modélisation cognitive de trois manières différentes. Un modèle cognitif peut se substituer à un utilisateur humain pour prévoir comment les utilisateurs agiront avec un système avant qu'il ne soit implémenté ou même prototypé. Un système peut générer un modèle cognitif d'un utilisateur en train d'interagir avec le système afin de modifier l'interaction au bénéfice de cet utilisateur. Enfin, des modèles cognitifs peuvent se substituer directement à des utilisateurs, de sorte que des groupes d'individus peuvent être simulés dans des situations qui nécessitent beaucoup de participants, par exemple l'apprentissage ou les loisirs. Cet article présente des exemples de tels modèles et leur conséquences sur la conception d'interfaces graphiques.

Keywords: Cognitive modeling, GOMS, intelligent tutoring, intelligent agents

Introduction

A cognitive model is a computer program that behaves like a human being. It may emulate the perceptual, cognitive and/or motor processes people go through to complete a task. It may take the same amount of time that people take to perform a task. It may make the same type of errors people make. It may take the same amount of time and require the same type of experience to learn to perform a task. It may do the same inefficient fumbling for a solution to a difficult problem. In all, the point is to have the computer behave like a human, not simply to get the job done with the least effort or in the least time.

Cognitive models are used in psychological research in several ways. They serve as a vehicle for understanding human behavior; if you can program a computer to behave the same way, you have demonstrated a level of understanding more rigorous than the typical box-and-arrow diagrams that are also models of a sort. If your model is successful at producing human-like behavior under certain assumptions, you can hypothesize that different behavior will emerge under different assumptions, change those assumptions in the model and see how it behaves. Explorations with models in this way can then be used to design experimental conditions that are likely to show measurable effects. So, cognitive models are useful tools for psychologists, but why should graphic interface designers care about cognitive models?

[1] An earlier version of this talk was presented at the European Workshop on Cognitive Modelling. (14th-16th November 1996, Technical University of Berlin, Berlin Germany.)

Cognitive models for human-computer interaction (HCI), as opposed to those for psychological science, have a different goal. In general, they are used to make interfaces "better" for users. There are at least three different uses for cognitive models in service of this general goal.

- Predicting human behavior on proposed interactive systems,
- Modeling the user as a guide for adaptive interaction, and,
- Substituting models for other participants in group interactions.

Predicting Human Behavior on Proposed Interactive Systems[2]

The overall motivation for HCI cognitive modeling efforts is to provide *engineering models* of human performance. In the ideal, such models produce *a priori* quantitative predictions of performance at an earlier stage in the development process than prototyping and user testing. That is, they predict execution time, learning time, errors, and identify those parts of an interface that lead to these predictions, thereby focusing the designer on what to fix. They allow analysis at different levels of approximation so predictions appropriate to the design situation can be obtained with minimum effort. They are straight-forward enough for computer designers to use without extensive training in psychology, and these models are integrated enough to cover total tasks. Although HCI research has not yet reached this ideal, GOMS [4] is currently the most mature of engineering models, has many of these properties, and can be truly useful in real-world system development.

GOMS is a method for describing a task and the user's knowledge of how to perform the task in terms of ***G***oals, ***O***perators, ***M***ethods, and ***S***election 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.

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, programmable television sets, and WWW pages.

GOMS has also been shown to be valid in single-user, active systems, where the system changes in unexpected ways or other people participate in the accomplishing the task. There are GOMS models, for instance, of radar monitoring and of video games, where

[2] The following section draws heavily from my previous writing in [7] and [8]. Extensive references to GOMS research and validation can be found in those works.

the system throws new situations at the user at a maniacal pace, and GOMS models of telephone operators interacting with customers. The knowledge gathered by a GOMS analysis is sufficient to predict what a person will do in these seemingly unpredictable situations.

GOMS can be used both quantitatively and qualitatively. Quantitatively, it gives good predictions of performance time and learning time. So it can be used to help in a purchasing decision or to see if a proposed design meets quantitative performance requirements.

Qualitatively, GOMS can be used to design training programs, help systems, and even the system itself. Since a GOMS model is a careful description of the knowledge needed to perform a given task and it describes the content of task-oriented documentation. You need only 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. When GOMS discovers a frequent goal supported by a very inefficient method, then the design can be changed to include a more efficient method. If GOMS shows that there are goals not supported by any method at all, then new methods can be added. GOMS may also reveal where similar goals are supported by inconsistent methods, a situation in which users are likely to have problems remembering what to do, and show how to make the methods consistent.

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.[3]

Example: Project Ernestine[4]

In 1988, the telephone company serving New York and New England (NYNEX) considered replacing the workstations then used by toll and assistance operators, who handle calls such as collect calls, and person-to-person calls, with a new workstation claimed to be superior by the manufacturer. A major factor in making the purchase decision was how quickly the expected decrease in average work time per call would offset the capital cost of making the purchase. Since an average decrease of one second in work time per call would save an estimated $3 million per year, the decision was economically significant.

To evaluate the new workstations, NYNEX conducted a large-scale field trial. At the same time, a research group at NYNEX worked with me to use GOMS models in an effort to predict the outcome of the field trial. First, models were constructed for the current workstation for a set of benchmark tasks. They then modified these models to reflect the differences in design between the two workstations, which included different keyboard and screen layout, keying procedures, and system response time. This modeling effort took about two person-months, but this time included making extensions to the GOMS modeling technique to handle this type of task and teaching NYNEX personnel how to use GOMS. The models produced quantitative predictions of expert call-handling time for each benchmark task on both workstations, which when combined with the frequency of each call type, predicted that the new workstation would be an average of 0.63 seconds slower than the old workstation. Thus the new workstation would not save money, but would cost NYNEX about 2 million dollars a year.

This was a counter-intuitive prediction. The new workstation had many technically superior features. The workstation used more advanced technology to communicate with the switch at a much higher speed. The new keyboard placed the most frequently used keys closer together. The new display had a graphic user interface with recognizable icons instead of obscure alphanumeric codes. The procedures were streamlined, sometimes combining previously separate keystrokes into one keystroke, sometimes using defaults to eliminate keystrokes from most call types, with a net decrease of about one keystroke per call. Both the manufacturer and NYNEX believed that the new workstation would be substantially faster than the old one, by one estimate, as much as 4 seconds faster per

[3] 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.

[4] The details of this application of GOMS, both technical and managerial, can be found in [3] and [5].

call. Despite the intuition to the contrary, when the empirical field-trial data were analyzed, they supported the GOMS predictions. The new workstation was 0.65 seconds slower than the old workstation.

In addition to predicting the quantitative outcome of the field trail, the GOMS models explained *why* the new workstation was slower than the old workstation, something which empirical trials typically cannot do. The simple estimate that the new workstation would be faster was based on the greater speed of the new features considered in isolation. But the execution time for the whole task depends on how all of the components of the interaction fit together, and this is captured by the critical path in the GOMS model. Because of the structure of the whole task, the faster features of the new workstation failed to shorten the critical path.

Thus, examination of the critical paths revealed situations in which the new keyboard design slowed down the call, why the new screen design did not change the time of the call, why the new keying procedures with fewer keystrokes actually increased the time of some calls, and why the more advanced technology communication technology often slowed down a call. The complex interaction of all these features with the task of the telephone operator was captured and displayed by GOMS in a way that no other analysis technique or empirical trial had been able to accomplish.

NYNEX decided not to buy the new workstations. The initial investment in adopting the GOMS technique paid off both in this one purchase decision, and by allowing NYNEX to make some future design evaluations in as little as a few hours of analysis work.

Modeling the User as a Guide for Adaptive Interaction

A second use of cognitive models in HCI is to model the knowledge or intentions of the user during interaction in order to serve that user better. There can be many applications of this, for example, active assistance, or intelligent agents attempting to divine the users' goals. But nowhere is it more clear than in educational systems that are attempting to teach students a cognitive skill. Early systems had an implicit "model" that if a screen had been presented, then the student had learned the material on the screen, or if a student solved a problem correctly, he or she reliably knew all the material necessary to solve that problem. If you've ever taught a class, you know that both these assumptions are poor ones. More modern systems use more complex models of human knowledge acquisition. For example, let us look at the "cognitive tutors" of the Advanced Computer Tutoring (ACT) Project

Example: Cognitive tutors.[5]

The purpose of the ACT Project is to develop computer-based tutors for mathematics and programming that assist students in problem solving. Their cognitive tutoring technology, based on a computational theory of thought called ACT-R [1], generate and follow the multiple possible solutions a student might attempt on any given problem and dynamically tailor instruction to each individual student and problem. Like a personal human tutor or coach, cognitive tutors observe student performance, identify strengths and weaknesses, and provide individualized, just-in-time instruction while students learn by doing.

Over the past decade, the ACT cognitive tutors have been experimentally tested by more than 2000 students and have been shown to dramatically accelerate student learning (students take 1/3 the time to master the material), increase test scores (by about a letter grade) and improve student motivation (as measured by third-party evaluators). These impressive results came only through the application of cognitive models to tutor-building.

The ACT tutors use a much more sophisticated architecture for their cognitive modeling than the GOMS models discussed above. They are based on the ACT-R cognitive architecture which embodies a theory of performance and learning, including forgetting and strengthening of knowledge. It even models the chances of "slips" and "lucky guessing" by the student.

As the student solves problems posed by the tutor, the system uses cognitive models in two ways. In a process called "model-tracing" it maps the student's solution onto a production-system model of ideal performance. On the basis of this mapping, the system does "knowledge-tracing" where it assesses how well the student knows the information in each production of the ideal model. It then adapts the problem selection to exercise those pieces of knowledge that the student has not yet mastered.

Participants in Group Interactions

A final use for cognitive models in HCI is as replacement for human participants in group interactions. Such systems can be used to train or rehearsing people in a domain that requires many

[5] The following section draws heavily from [2] and from the material on the Advanced Computer Tutoring Project homepage:
http://sands.psy.cmu.edu/ACT/tutor/tutoring.html
Additional publications can be found by following links on that page.

participants, e.g., high-level commanders in a simulated theater of war, or the personnel of who launch the NASA Space Shuttle. Furthermore, organizations would like to test out proposed changes in their work processes, technology, or organizational structure. In the latter case, this use of cognitive modeling becomes similar to the first example: predicting the effects of proposed changes in the task environment.

(Note that this is different from intelligent agents as participants in group interaction. These cognitive models are to emulate humans in important parameters (e.g., both functional response and time to respond) as opposed to doing a job without the expectation that it be done *as a human* would do it. Although intelligent agents may play an important role in the future of interactive systems, they are not constrained by the same requirements as cognitive models.)

Example: Soar-based intelligent forces[6]

Military organizations have always had the need to train and practice their people in the processes of war. Previously, this took the form of "war games" staffed with many, many personnel. Recently, both the technology of actual engagement (e.g., that tactical air fighting often occurs out of range of visual sighting, and the pilot only sees the information given to him or her through the interfaces in the cockpit) and the technology of simulation have evolved to make a simulated theater of war (STOW) a possibility. If simulated agents could be created to emulate human behavior with sufficient fidelity, then realistic training and rehearsal becomes cost-effective. Not only can people practice on the current technology, but the effects of proposed changes in doctrine, tactics, or weapon systems can be assessed.

To these ends, the major goal of STOW-97 (October 29-31, 1997) was to demonstrate that it was possible to generate high-fidelity behavior for a large-scale distributed entity-level simulation of a complete theater battle, such as the Iraq/Kuwait war in the early 1990's. In addition, STOW-97 was a full-fledged operational training exercise involving active duty personnel (approximately 1000 humans) and simulated forces, both semi-autonomous and "intelligent" (approximately 50,000 simulated entities).

Based on a theory of cognition called Soar [9] (as complex as ACT-R, but different in its details), TacAir-Soar provides competent pilot-level behavior for a wide range of air roles, including defensive counter air, offensive counter air, close air support, and the different elements of strike and interdiction packages (air-to-ground attack, escorts, suppression of enemy air defense, and sweeps). Additionally, TacAir-Soar can be used in a number of support and coordination roles for air missions, including airborne early warning, refueling, forward air control, and reconnaissance. In late October of 1997, TacAir-Soar provided the behaviors for all fixed wing aircraft entities and missions for STOW-97. With more than 5,000 rules and 500 operators, TacAir-Soar is one of the largest real-time "expert" systems ever developed.

RWA-Soar provides competent pilot-level behavior for a rotary-wing attack mission and a combined rotary-wing transport and escort mission. It also provides automated commanders for companies of attack helicopters. In late October of 1997, RWA-Soar provided attack, transport, and escort missions for STOW-97.

TacAir-Soar and RWA-Soar are distinguished from other entity-level computer-generated forces by being truly autonomous. As with human forces, they are initially provided with exercise, operations, and mission information. They then plan and fly their missions, respond to threats, and communicate via simulated radios with other synthetic and human forces, all without any human intervention other than the standard interactions defined by the environment and doctrine.

STOW-97 consisted of 48 hours of continuous operations (7 AM ET October 29 to 7 AM ET October 31, 1997). There were a total of 722 FWA scheduled missions plus numerous ground-alert missions for the 48 hours (and an as yet undocumented number of RWA missions). The missions varied in length from one and a half hours to eight hours, with the median being three hours. At any one time, there were from 30 to 80 planes airborne. The planes were run on approximately 25 Pentium Pro's (P6's) with 4-6 vehicles/machine.

Tambe and his colleagues (1995, after STOW-E a smaller-scale STOW with 2,000 participating entities) compiled a list of the requirements for automated pilots. Many of these requirements emphasize the cognitive-plausibility aspects of these systems.

•*Goal-driven and knowledge-intensive behavior.* including planning, executing a plan, and replanning as needed.

[6] The following section draws heavily from [10] and from the material on the Soar-IFor web pages at the University of Southern California and the University of Michigan:

http://www.isi.edu/soar/soar-ifor-project.html

http://ai.eecs.umich.edu/ifor/index.html

Additional publications can be found by following links on those pages.

•*Reactivity* to changes in the environment, mitigated by a memory for internal state.

•*Real-time performance* to fully participate in mixed-entity simulations

•*Conformance to human reaction times and limitations*. Super-human abilities (e.g., faster reaction time, or unlimited memory) or sub-human abilities (e.g. reaction times too slow to keep a jet aloft in high-stress maneuvers) will give an unrealistic behaviors.

•*Overlap of performance of multiple high-level tasks.*

•*Multi-agent coordination and communication*. These entities must be able to coordinate their actions with the other entities, both human and simulated, in order to work as a team.

•*Agent Modeling*. Especially opponent modeling done concurrently with flying the mission!

•*Temporal reasoning*. Participating in a real-time simulation requires temporal reasoning at least for planning and executing plans, timing communications, and recognizing opponents' plans.

•*Explanation*. These agents need to be able to explain their behavior after the fact for postmission debriefing. They need to accept questions and generate answers, both in verbal and graphical form, as needed, to best explain their assessment of the situation and actions.

•*Maintenance of episodic memory* used for the explanation capability (an agent cannot explain its behavior if it cannot remember it), but also to help with agent modeling, i.e., tracking the movements of an opponent through time to reveal its intentions.

What do these applications of cognitive models mean for GI design?

Each of these uses of cognitive models provide opportunities for GI designers and also pose challenges to the GI design community.

Predictive models

These models reliably predict quantitative measures of skilled performance (i.e., time to execute tasks) and relative method-learning time of WIMP interfaces. They have been used as a design tool for real systems (see 11 cases in John & Kieras, 1996) and could be used more widely with great potential benefit. Tutorial materials exist (e.g., presented at the CHI and HFES conferences and in the Handbook of Human-Computer Interaction [6]); this is a plea to use them.

However, the tool-support for doing these models is relatively meager, and confined to research labs. This is a plea to do something about that! Apply all GI's knowledge of tools for programming languages to tools for cognitive modeling, at a production-quality level to help bring this technique into routine use.

Cognitive Tutors

Again, this technique works for teaching people procedures and could be directly applied to help systems and tutorials for GI products. But, like the GOMS models above, the tools for creating them are not yet production quality, making the effort to produce cognitive tutors more costly than the six-month or year turn around on new releases of GI software warrant (as opposed to the unchanging rules of algebra or geometry, for example). Better tools would help. An intriguing possibility is to combine the need for tools for predictive models with cognitive tutoring tools, because a GOMS analysis is very close to the knowledge analysis necessary for a cognitive tutor.

Participants in Group Interactions

Finally, this last application of cognitive modeling opens whole new domains for GI, rather than contributing to the current way we do business. Again, better tools are needed, but specific challenges include the communication between multiple agents and debriefing explanation capability. Please bring all your experience with visualization and communication to bear on this problem.

References

1. Anderson, J. R. (1993) Rules of the mind. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.
2. Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995) Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences, 4*(2), pp. 167-207.
3. Atwood, M. E., Gray, W. D., & John, B. E. (1996) Project Ernestine: Analytic and empirical methods applied to a real-world CHI problem. In M. Rudisill, C. Lewis, P. B. Polson, and T. D. McKay (Eds.), *Human-Computer Interface Design: Success Stories, Emerging Methods and Real-World Context* (pp. 101-121). San Francisco: Morgan Kaufmann.
4. Card, S. K., Moran, T. P., & Newell, A. (1983). *The psychology of human-computer interaction.* Hillsdale, NJ: Lawrence Erlbaum Associates.
5. Gray, W. D., John, B. E., & Atwood, M. E. (1993) Project Ernestine: Validating a GOMS analysis for predicting and explaining real-world task performance. *Human-Computer Interaction, 8,* pp. 237-309.
6. Helander, M. G., Landauer, T. K., & Prabhu, P. V. (1997) *Handbook of Human-Computer Interaction,*

Second Completely Revised Edition. Amsterdam, Netherlands: North-Holland.

7. John, B. E. (1995) Why GOMS? *interactions, vol. 2*, no. 4. pp. 80-89.
8. John, B. E. & Kieras, D. E. (1996) Using GOMS for user interface design and evaluation: Which technique? *ACM Transactions on Computer-Human Interaction, 3* (4), pp. 287-319.
9. Newell, A. (1990) *Unified Theories of Cognition.* Canbridge, MA: Harvard University Press.
10. Tambe, M ., Johnson, W. L. Jones, R. M., Koss, F., Laird, J. E., Rosenbloom, P. S., & Schwamb, K. Intelligent Agents for Interactive Simulation Environments, *AI Magazine*, Spring 1995, pp. 15-39.