Automating CPM-GOMS

Bonnie John[1], Alonso Vera[2], Michael Matessa[2], Michael Freed[2], and Roger Remington[2]

[1]Human-Computer Interaction Institute
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA
+1 412 268 7182
bej@cs.cmu.edu

[2]MS 262-4
NASA Ames Research Center
Moffett Field, CA 94035
+1 650 604 6294
avera@arc.nasa.gov

ABSTRACT

CPM-GOMS is a modeling method that combines the task decomposition of a GOMS analysis with a model of human resource usage at the level of cognitive, perceptual, and motor operations. CPM-GOMS models have made accurate predictions about skilled user behavior in routine tasks, but developing such models is tedious and error-prone. We describe a process for automatically generating CPM-GOMS models from a hierarchical task decomposition expressed in a cognitive modeling tool called Apex. Resource scheduling in Apex automates the difficult task of interleaving the cognitive, perceptual, and motor resources underlying common task operators (e.g. mouse move-and-click). Apex's UI automatically generates PERT charts, which allow modelers to visualize a model's complex parallel behavior. Because interleaving and visualization is now automated, it is feasible to construct arbitrarily long sequences of behavior. To demonstrate the process, we present a model of automated teller interactions in Apex and discuss implications for user modeling.

Keywords

GOMS, Apex, Task/User Modeling, Tool Support for Usability Evaluation.

INTRODUCTION AND MOTIVATION

One of the difficulties in developing human interfaces to complex systems is anticipating the response of users to the large space of possible system states and design options. Even extended empirical user testing can fail to uncover serious difficulties. It would be useful to have a computational representation of the user that would allow the designer to simulate user responses to a variety of situations and design options. Though the field is far from having a complete model of the user, several computational modeling approaches have been successful in making accurate predictions of user choices as well as task completion times (e.g., [15, 25, 29, 31, 32]). Of the several architectures available to model human users, the Goals, Operators, Methods, and Selection (GOMS) method [6, 21] has been the most widely used, providing accurate, often zero-parameter, predictions of the routine performance of skilled users in a wide range of procedural tasks [6, 13, 15, 27, 28].

GOMS is meant to model routine behavior. The user is assumed to have methods that apply sequences of operators and to achieve a goal. Selection rules are applied when there is more than one method to achieve a goal. Many routine tasks lend themselves well to such decomposition. Decomposition produces a representation of the task as a set of nested goal states that include an initial state and a final state. The iterative decomposition into goals and nested subgoals can terminate in primitives of any desired granularity, the choice of level of detail dependent on the predictions required.

Although GOMS has proven useful in HCI, tools to support the construction of GOMS models have not yet come into general use. Several tools have emerged from the research world, e.g., QGOMS [3], CATHCI [30], GLEAN [24]. All of these tools aid the modeler to some extent, but all have drawbacks that prevent them from being heavily used in design practice today [2]. In addition, none of them automate any part of the modeling process. However, limited demonstrations of the potential for automating some portions of GOMS modeling have been made [4, 16, 26].

We extend these promising directions with a tool that automates part of the GOMS modeling process using an existing computational architecture, Apex [9, 10]. Our work differs from that of our predecessors because Apex served not only as an implementation platform but provided new insights into GOMS modeling itself. In addition, the previous work focused on the higher-level members of the GOMS modeling family (KLM, CMN-GOMS and NGOMSL; [21]) whereas our use of Apex emphasizes the lowest-level GOMS modeling technique (CPM-GOMS). Employing reusable templates of behavior, our tool allows the modeler to specify procedural knowledge at a task-level and automates the translation of that knowledge into interleaved cognitive, perceptual and motor operators.

CHI 2002, April 20-25, 2002, Minneapolis, Minnesota, USA.

The next section will introduce key aspects of CPM-GOMS and the procedure for constructing such models by hand. Then we will describe Apex, the insights it afforded, and the procedure for constructing a CPM-GOMS model with that tool. Finally, we will present an example use of the tool and a comparison of the resulting CPM-GOMS model to user data.

CONSTRUCTING CPM-GOMS MODELS

John & Kieras [21] described four varieties of GOMS modeling techniques. Three make the assumption that all operators occur in sequence and usually do not contain operators below the activity level (e.g., type-string, move-and-click-mouse). These three are the original formulation by Card, Moran and Newell [5, 6] termed CMN-GOMS, the Keystroke-Level Model (KLM) also formulated by Card Moran and Newell [6], and NGOMSL [23]. The fourth, called CPM-GOMS [17, 18], uses operators at the level of the Model Human Processor (MHP, [6]) and assumes that operators of the **c**ognitive processor, **p**erceptual processor, and the **m**otor processor can work in parallel to each other subject to information-flow constraints. The first three techniques have been supported by research tools for about a decade, where modelers can draw hierarchical goal decomposition in a tree diagram (QGOMS, [3]), program it in a dedicated programming environment (GLEAN, [24]) or even automatically generate most of the model simply by demonstrating a task (CRITIQUE, [16]).

Unlike the first three GOMS methods, CPM-GOMS human performance predictions are constructed from primitives that include estimates of the time for the elementary cognitive, motor, and perceptual operations. These primitives are hypothesized to underlie actions such as typing a key or moving a mouse. Much of the power of CPM-GOMS to predict skilled behavior comes from its ability to model overlapping actions by interleaving cognitive, perceptual, and motor operators. Although it could be argued that CPM-GOMS has been the most economically successful of the GOMS methods (saving a telephone company $2 million per year [15]), it has had no dedicated tool support to date.

Crafting CPM-GOMS Models by Hand

CPM-GOMS models have traditionally been expressed in PERT charts, a representation familiar to project managers. Every operator is represented as a box (a task) with a duration (in milliseconds). If an operator must have information that is the output of another operator, then it is said to be dependent on that operator and must wait for it to complete before starting itself. Likewise, if two operators use the same processor of the MHP (e.g., cognitive processor, vision, or the right hand), one must wait for the previous to complete before starting. Thus, a CPM-GOMS model of a user's task consists of boxes with durations and dependency lines between them. Figure 1 shows a model of a person carefully moving a mouse to a target on the screen and clicking on that target.

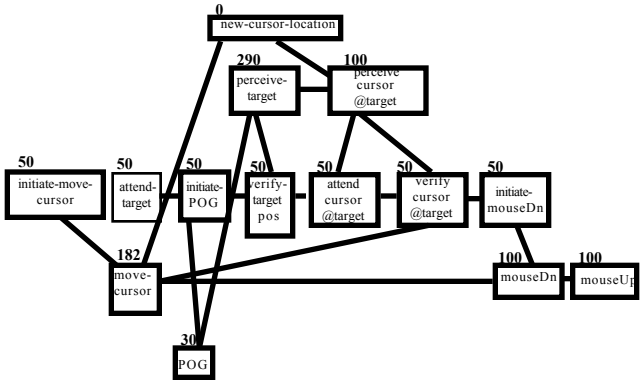


Figure 1: Model of carefully the cursor to a target and clicking the mouse button (adapted from [11]).

Procedure for constructing a CPM-GOMS model with MacProject

Models were created using MacProject, a project management tool originally produced by Apple, improved by Claris, and no longer commercially available. The key feature of MacProject that made it possible for CPM-GOMS models to be constructed is that pre-established patterns of operators could be stored in a library file and then copy-and-pasted into a new canvas, preserving all relevant information about the patterns (e.g., duration, dependencies, position on the page). These patterns, which we called "templates" [20], were of commonly recurring task-level activities in HCI. Each template was very short, some encompassed just a fraction of a second and others were up to several seconds. Templates exist for HCI tasks including typing, visually getting information from a screen (with or without eye-movements), pressing a single key, having a short conversation, etc. The pattern shown in Figure 1 for selecting a target is an example of a template.

To build a CPM-GOMS model, the modeler would start with a hierarchical goal decomposition, usually in the form of a CMN-GOMS model. This goal decomposition would continue until the leaves formed a sequence of keystroke-level activities necessary to complete the task. After completing the goal decomposition, the modeler would choose the templates that achieve the activities and copy and paste them into a blank workspace. The modeler then drew dependency lines between operators from adjoining templates that use the same processor, i.e., from one template's last cognitive operator to the next template's first cognitive operator, etc. Since each template was on the order of a second long, an interesting model would include scores of templates and be comprised of hundreds of MHP-level operators and their dependencies.

After copying the appropriate templates into the model, each operator in each template had to be given a unique name to allow the modeler to keep track of the model as they scrolled through many screens of MacProject. Furthermore, the modeler had to remember to fill in durations for some of the operator-boxes because the actual

duration of the operator varied with the task situation. Many modelers, novice and experienced alike, missed an operator or two in this step, an error that propagates and exacerbates problems throughout the rest of the modeling process.

After all templates are copied in, joined together serially, and customized to the task being modeled, they form a complete PERT chart for the task and MacProject displays the critical path (longest path) for task. At this point, the activities embodied in the templates are modeled as occurring in strict sequence. CPM-GOMS gets it predictive power by breaking the assumption of seriality thereby modeling the ability of highly skilled people to think ahead to the next step while completing the current step, essentially doing several things in parallel. To get this effect in CPM-GOMS models, at every juncture between two templates, the modeler had to consider whether to literally break the dependency line drawn in earlier, put an operator ahead of another operator, and reconnect the dependency lines appropriately. A full set of rules to dictate this step has never been articulated. The first consideration was whether there was sufficient slack time in the critical path to insert an operator belonging to a later template between two operators of the current template. However, deciding when it was appropriate to take advantage of that slack time was more of a craft than an engineering science, involving knowledge of the critical path, the task being modeled, psychology, and intuition. Furthermore, the breaking and reconnecting of scores of dependency lines also usually resulted in some errors of omission, which greatly affected the critical path of the final model.

Although a prose description does not do justice to the procedure, the previous paragraphs attempt to convey that crafting CPM-GOMS with MacProject was difficult, labor-intensive, tedious and error-prone. Add to this the fact that MacProject was not designed for tasks at the millisecond level (the modeler had to work in minutes and do time conversion and it did not have a big enough canvas for long tasks) and the process was also frustrating. Even experienced modelers would take hours to model each minute-long task and then put the model away for a few days and revisit it with "new eyes" to find the errors and inconsistencies. The resulting accuracy of the models, their predictive power, and the eventual clarity of presentation was worth the effort through several projects, but the process was always painful.

Automatically Generating CPM-GOMS Models with Apex

Apex is a computational architecture used to model human behavior in complex dynamic tasks. It incorporates a reactive planner [8] providing capabilities that are a superset of those needed to build GOMS models [11]. These capabilities allowed us to map the concepts of CPM-GOMS to those of Apex and implement CPM-GOMS models directly in Apex.

The Apex architecture

Resources. The Apex architecture includes the concept of resources, which map directly to the MHP's processors, and hence to CPM-GOMS models. Resources operate serially within themselves and are thereby occupied by a single task for the duration of that task. Apex currently has memory, vision, gaze, and right/left hand resources that map to the MHP's cognitive, perceptual, and motor processes. It also has facilities for including more resources as needed by more complex tasks. Apex allocates these resources and others to the tasks it is attempting to execute.

Hierarchical goal decomposition. The hierarchical goal structure of a GOMS model can be expressed in Apex with its Procedure Description Language (PDL). In PDL, a procedure (GOMS method) consists of a number of steps. PDL steps are decomposed hierarchically into procedures of simpler steps until those steps bottom out in primitive actions (GOMS operators) that occupy resources. The decision to perform a particular procedure is mediated by a selection operator (GOMS selection rules). The PDL language is similar to the implementation of NGOMSL in GLEAN [24]. However, PDL is closer in philosophy to CMN-GOMS in that it assigns no time to goal manipulation, only the execution of operators.

Step ordering. In PDL, steps can be assigned a strict serial ordering (like CMN-GOMS or NGOMSL) by explicitly setting the precondition of one step to be the completion of the preceding step. However, the Apex architecture also supports parallelism because if no explicit "waitfor" precondition is assigned, steps can run in parallel (subject to resource constraints). This default assumption of parallel activity is essential to CPM-GOMS models. Apex has a third possibility for ordering steps called *priorities*. In PDL, steps can be assigned a priority. When the step contends for use of a resource, its priority is compared to the priorities of other steps also contending for the same resource, and the task with the highest priority wins the resource. In terms of CPM-GOMS, this allows a sort of soft ordering of templates; task T2 should go after task T1 unless T1 is not using the resource required by T2, in which case T2 can take it.

Time assignment. Primitive actions are assigned durations that can be constants or a function of the environment. For example, the mouseDn action in Figure 1 is assigned an empirically determined value of 100 ms, while the move-cursor action is assigned a time calculated by Fitts's Law. The overall time to run several such actions is calculated by Apex, which takes into account when the actions start and what actions may be running in parallel at any particular time.

Expressing CPM-GOMS Templates in PDL

CPM-GOMS templates can be straightforwardly expressed in PDL. For example, the PERT chart template shown in Figure 1 is expressed in PDL in Figure 2. Each box (operator) in Figure 1 is a step in PDL, labeled "c" for cognitive, "p" for perceptual, and "m" for motor. Dependency lines that go between rows are expressed as explicit "waitfors" in the PDL. For example, the move-cursor motor operator (m1) waits for the initiate-move-cursor cognitive operator (c1). Dependency lines in a row of CPM-GOMS operators are implemented at the next lower

```
(procedure
 (index (slow-move-click ?target))
 (step c1 (initiate-move-cursor ?target))
 (step m1 (move-cursor ?target)
                    (waitfor ?c1))
 (step c2 (attend-target ?target))
 (step c3 (initiate-eye-movement ?target)
                    (waitfor ?c2))
 (step m2 (eye-movement ?target)
                    (waitfor ?c3))
 (step p1 (perceive-target-complex ?target)
                    (waitfor ?m2))
 (step c4 (verify-target-position ?target)
                    (waitfor ?c3 ?p1))
 (step c5 (attend-cursor-at-target ?target)
                    (waitfor ?c4))
 (step w1 (WORLD new-cursor-location ?target)
                    (waitfor ?m1))
 (step p2 (perceive-cursor-at-target ?target)
                    (waitfor ?p1 ?c5 ?w1))
 (step c6 (verify-cursor-at-target ?target)
                    (waitfor ?c5 ?p2))
 (step c7 (initiate-click ?target)
                    (waitfor ?c6 ?m1))
 (step m3 (mouse-down ?target)
                    (waitfor ?m1 ?c7))
 (step m4 (mouse-up ?target)
                    (waitfor ?m3))
 (step t1 (terminate)
                    (waitfor ?m4)))
```

Figure 2. PDL code for the CPM-GOMS template shown in Figure 1.

level below the code in Figure 2 where the primitive operators are assigned to their resources. That is, both the move-cursor operator and the mouse-down operator are assign to the right-hand resource; since that resource can only do one operator at a time a dependency emerges from Apex's architecture.

Notice in Figure 2 that neither c1 (initiate-move-cursor) nor c2 (attend-target) wait for the completion of any step in this template. This is theoretically appropriate because when selecting a target with a mouse a skilled user can start to point before she starts to look at the target, or start to look before she starts to point. The PDL code enforces no dependency between these two cognitive operators; resource constraints will automatically pick the most appropriate operator at run time.

Articulating CPM-GOMS Template Interleaving Rules

By attempting to express CPM-GOMS templates in PDL and create a complete model of an HCI task, we were able to articulate for the first time reliable rules for appropriately interleaving CPM-GOMS operators. These rules depend on templates like the one in Figure 2, where the operators occupy resources assigned in PDL code below the level of the template and inherit priorities from the goal decomposition code above the template.

The details of how these rules work are beyond the scope of this paper. However, roughly, they allow a momentarily free resource to be seized by an operator from a lower-priority template (i.e., later in the sequence) if no operator from a higher-priority template is ready to request it. If the lower-priority operator can complete before an operator from a higher-priority template requests the resource, that lower-priority operator has successfully interleaved. If it cannot complete before the resource is requested, then it is terminated and reset and it must re-compete for the resource at its next opportunity.

The decision process about how to interleave is completely different for the Apex architecture than for the human modeler using MacProject. The modeler uses knowledge of the entire timecourse of the task encoded in the critical path of the PERT chart, while Apex makes it selections at runtime with no foreknowledge of what other operators are waiting for resources. Despite the differences in decision-making mechanisms, the resulting Apex models interleave operators just as MacProjects models do when created by experts in CPM-GOMS.

Procedure for constructing a CPM-GOMS model with Apex

The first step in creating a CPM-GOMS model with Apex is the same as doing it by hand: create a CMN-GOMS goal decomposition. In Apex, this decomposition is formalized in PDL code instead of just being jotted down on paper or being typed into a word processor. As can be seen in Figure 3, the syntax of PDL code is sufficiently lightweight that this formalization is not a crushing burden.

Both methods depend on previously set-up templates of reusable skills like pointing with a mouse or typing. These reusable templates take the form of PERT charts when using MacProject and PDL code when using Apex. These templates are coded by researchers in cognitive modeling not by system designers modeling a particular interface and task set. Thus, the psychological science is "built in" to the templates by experts in psychology and human modeling so that they can be used easily by non-expert modelers. In addition, in Apex, the psychologists also provide the lower-level code assigning operators to actual and virtual resources, which the modeler never need see.

When the PDL goal decomposition reaches the level of the templates, they simply call the appropriate template as a step in the PDL code. Next, the modeler runs the model using Apex's GUI, Sherpa (Figure 3). By default, Sherpa produces a textual trace of the model, showing the time when each operator starts and completes. However, at the press of a button, Sherpa converts that trace into a PERT chart. The resulting PERT chart contains all the operators, their durations, and their dependencies without any further input from the modeler. No bookkeeping, no deleting or drawing dependency lines, no difficult thinking about interleaving.

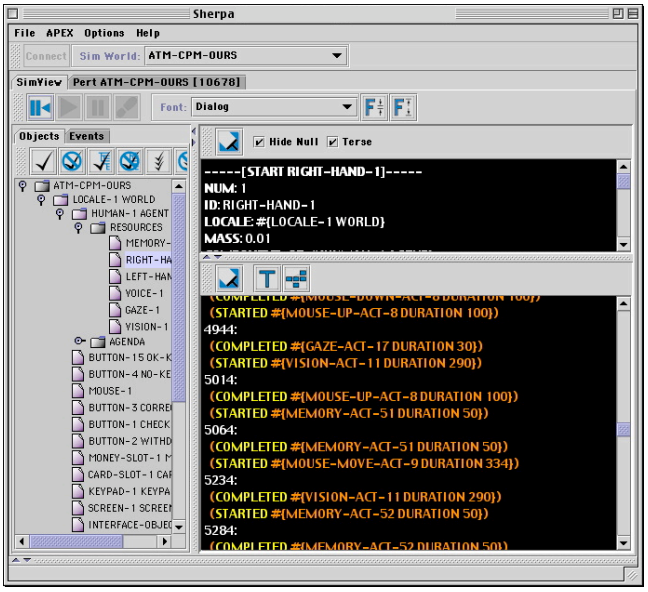
In addition, Sherpa has some helpful features tailored to the needs of CPM-GOMS modeling. For instance, the PERT chart can be shrunk horizontally to see patterns within the model or stretched to zoom in on the information in particular operators. It can be toggled to either display in standard PERT chart view where the width of each box is determined only by how much text must fit into it, or to a view where the width of the box is proportional to the time it occupies a resource (for short duration operators, a click will reveal the information that cannot be fully presented). Sherpa charts can be arbitrarily long to accommodate

```
(procedure
 (index (do banking))
 (step s1 (initiate session) (priority 300))
 (step s2 (do transaction) (priority 200))
 (step s3 (end session) (priority 100))
 (step t (terminate) (waitfor ?s3  ?s2 ?s1)))

(procedure
 (index (do transaction))
 (step s1 (choose withdraw) (priority 240))
 (step s2 (choose account) (priority 230))
 (step s3 (enter amount) (priority 220))
 (step s4 (retrieve money) (priority 210))
 (step s5 (terminate) (waitfor ?s4 ?s3  ?s2
?s1)))

(procedure
 (index (enter amount))
 (step s1 (enter-number 8-key) (priority 223))
 (step s2 (enter-number 0-key) (priority 222))
 (step s3 (enter-CORRECT) (priority 221))
 (step s4 (terminate)
(waitfor ?s2 ?s1 ?s3)))

(procedure
 (index (enter-number ?number))
 (step s1 (fast-move-click ?number))
 (step s2 (terminate) (waitfor ?s1)))
```



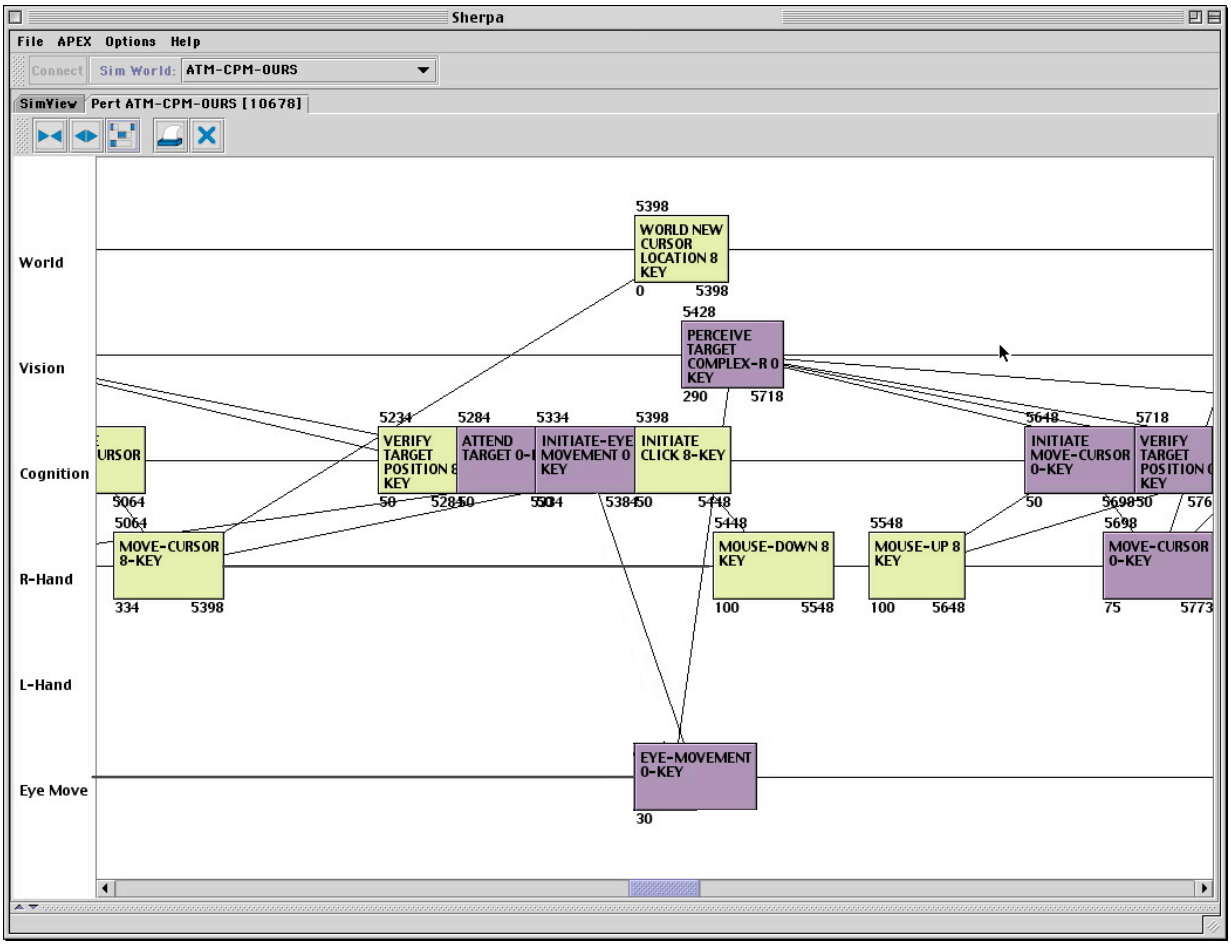


Figure 3: PDL code expressing the CPM-GOMS model can be run in Sherpa and automatically rendered as a PERT chart.

lengthy models. Finally, Sherpa automatically colors the operators from each template a distinct hue so interleaved operators visually pop out at the modeler. (These hues were specifically selected to also print distinctively in black & white, so some pop-out is visible in Figure 3.)

EXPERIENCE USING APEX FOR CPM-GOMS MODELING

We have used Apex to create a CPM-GOMS model for a simple HCI task, withdrawing money from an ATM. We ran two users through the same task, practicing them extensively because CPM-GOMS models are expected to predict the performance of highly-skilled users [1].

The Task

The task was to withdraw $80 from the checking account on a Visual Basic mock-up of an Automated Teller Machine (ATM). This ATM task allowed us to collect keystroke-level data with which to evaluate the automatically-generated models. Data at this level was never collected for most previously-published CPM-GOMS tasks [7, 19, 20], or was archived in an inaccessible form [1]. Moreover, using a mouse-based point-and-click task allowed us to borrow CPM-GOMS templates constructed by Gray and Boehm-Davis [14] with little modification.

To collect keystroke-level data, users performed actions on the ATM by using a mouse to click on simulated keys or slots. They were instructed to perform the following steps to satisfy the goal of withdrawing $80:

Insert card (click on the picture of the card slot)
Enter PIN (click on the 4, 9, 0, and 1 buttons in turn)
Press OK (click on OK button)
Select transaction type (click on withdraw button)
Select account (click on checking button)
Enter amount (click on 8 and 0 buttons)
Press if correct/not correct? (click on correct button)
Take cash (click on the picture of the cash slot)
Answer question about wanting another transaction (click on No button)
Take card (click on the picture of the card slot)
Take receipt (click on the picture of the cash slot)

The users performed this task without deviation 100 times. This level of practice mimics that used by both Card Moran and Newell [6] in a text-editing task and Baskin and John [1] in a CAD drawing task when they explored the effects of extensive practice on match to various GOMS models.

The Apex CPM-GOMS Model

The CPM-GOMS model was constructed by expressing the goal decomposition in PDL (see Figure 3). The decomposition continued until the steps were the names of two CPM-GOMS templates: Slow-Move-Click, and Fast-Move-Click. The underlying cognitive, perceptual, and motor operators for these two templates were taken directly from Gray & Boehm-Davis [14] where those researchers matched these CPM-GOMS models to data from several variations of a target selection task. The Slow-Move-Click template is shown in PERT chart form in Figure 1 and in PDL in Figure 2. Portions of two Fast-Move-Click templates are shown interleaved in the generated PERT chart in Figure 3. For Gray and Boehm-Davis, Slow-Move-Click represents a careful selection of a visible target because there was uncertainty about where the target would appear in each trial. Fast-Move-Click represents a more confident selection of a stationary target when the user knows where that target lies. In our model we chose to use Fast-Move-Click for clicking on the buttons because they were stationary and of reasonable size. We chose to use Slow-Move-Click for clicking on the card and cash slots because these slots were represented in Visual Basic as only being a few pixels wide and were difficult to hit unless the user was exerting special care.

The 10-1/2 seconds of behavior required to withdraw cash was comprised of 15 templates, about 180 cognitive, perceptual, and motor operators. Fifty-three of these operators interleave, that is, they begin before all the operators in the template that precedes them are completed. All parameters in the models were set *a priori* from prior research, without reference to the collected data.

Comparing the Model to Data

Figure 4 shows a comparison of the CPM-GOMS predictions and the mean times of the error-free trials from 91 to 100 for each user. As is apparent from Figure 4, the fits to the data are quite good. The average absolute difference between model predictions and observed data was only 80 ms and the absolute average percent error was 13%. This fit required no parameters to be set with reference to the data. This adds support to the finding that CPM-GOMS models predict behavior well at around the 100th trial of a practiced procedure [1].

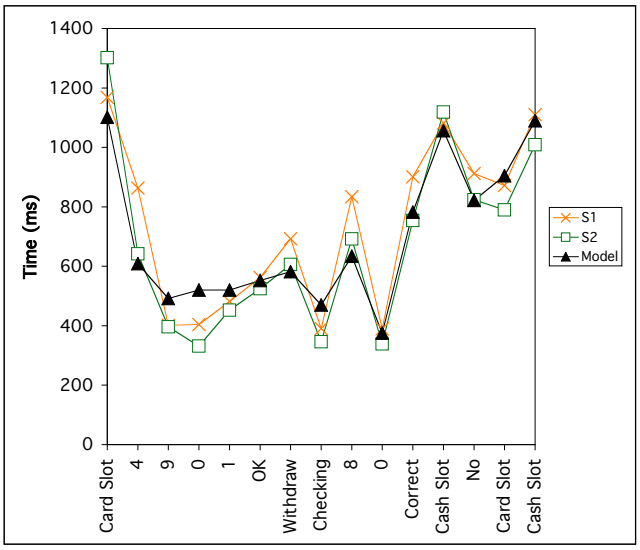


Figure 4: Model predictions and user results

CONCLUSIONS AND FUTURE WORK

Using Apex has allowed us to uncover regularities in CPM-GOMS models that were previously unknown. We leveraged these regularities by expressing them in the Apex architecture such that CPM-GOMS models can be

automatically generated from a higher-level goal decomposition and low-level templates of primitive HCI behavior like moving a mouse and typing. The resulting zero-parameter model we created for an ATM task fit the data very well.

In order for computational cognitive modeling to come into wider use in the design process, it is necessary to make the production of models easier and more valid than it has been in the past. We believe that packaging the abundance of data on human perceptual, cognitive, and motor phenomena into a set of behavioral primitives (templates) that can be directly incorporated into predictive, computational models is a promising way to proceed. Templates reduce the amount of psychological and methodological knowledge required to build models, which allows the modeler to focus on the task analysis instead of low-level psychological theories.

Promising as these results are, additional work needs to be done. For example, we need to demonstrate the power of building CPM-GOMS models in Apex by producing many more of them. In particular, the user actions in the ATM task are too sequential to show substantial parallelism and interleaving, which is the unique strength of CPM-GOMS models. To remedy this deficit, we plan first to reproduce existing successes of CPM-GOMS models [1, 7, 14, 15, 19] to demonstrate the expressive power of Apex. This effort will also allow us to compare previously-published, hand-constructed models with Apex-generated models to better understand the operation and implications of automatic generation. We then plan to extend beyond previously modeled tasks to more complex tasks like air traffic control.

In order to accomplish this first goal, we will have to add several more templates to the library already containing slow-move-click and fast-move-click. For instance, to reproduce the models of drawing in CAD, we will add a typing template and several other variations of moving and clicking a mouse (e.g., with a snapping and/or changing cursor and chorded mouse-button clicks). Every template added to the library increases the usefulness of Apex for CPM-GOMS modeling [12].

We will continue to develop Apex as a tool, making it robust, fast, and usable in the context of predictive modeling for system design. In addition, we will add libraries of templates, documented examples, and documentation.

ACKNOWLEDGMENTS

We would like to thank Michael Dalal and Robert Harris for their efforts on the Apex project. This research was supported in part by the NASA Intelligent Systems Program, the NASA Aviation Operations Systems Program and by NASA Award #NAG2-1472.

REFERENCES

1. Baskin, J. D., and John, B. E. (1998). Comparison of GOMS Analysis Methods. *Proceedings of ACM CHI'98 Conference on Human Factors in Computing System (Summary)* 1998 v.2 p.261-262.
2. Baumeister, L. K., John, B. E., Byrne, M. D. (2000). A Comparison of Tools for Building GOMS Models. *In Proceedings of ACM CHI2000 Conference on Human Factors in Computing Systems*, v.1 p.502-509.
3. Beard, David V., Smith, Dana K. & Denelsbeck, Kevin M. (1996). Quick and Dirty GOMS: A Case Study of Computed Tomography, *Human-Computer Interaction, 11* (2) p.157-180.
4. Byrne, M. D., Wood, S. D., Sukaviriya, P. N., Foley, J. D. & Kieras, D. (1994). Automating Interface Evaluation, *Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems*, B. Adelson, S. Dumais, & J. Olson (Eds.), v.1, pp. 232-237. New York: ACM Press.
5. Card, S.K., Moran, T. P., & Newell, A. (1980). The keystroke-level model for user performance with interactive systems. *Communications of the ACM*, *23*, 396-410.
6. Card, S. K., Moran, T.P. & Newell, A. (1983). *The Psychology of Human-Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.
7. Chuah, M. C., John, B. E., & Pane, J. (1994). Analyzing graphic and textual layouts with GOMS: Results of a preliminary analysis. *In Proceedings Companion of CHI, 1994*, (Boston, MA, USA, April 24-28, 1994). New York: ACM, pp. 323-324.
8. Firby, R. J. (1989). Adaptive Execution in Complex Dynamic Worlds. Ph.D. thesis, Yale University, Computer Science Department. Technical Report 672.
9. Freed, M. (1998a) Managing multiple tasks in complex, dynamic environments. In *Proceedings of 15th National Conference on Artificial Intelligence*, (Madison, Wisconsin,) Menlo Park, CA: AAAI Press/ MIT Press. pp. 921-927.
10. Freed, M. (1998b) Simulating Human Performance in Complex, Dynamic Environments. Doctoral Dissertation, Northwestern University.
11. Freed, M. & Remington, R. (2000a) GOMS, GOMS+ and PDL. *In Working Notes of the AAAI Fall Symposium on Simulating Human Agents*. Falmouth, Massachusetts.
12. Freed, M. & Remington, R. (2000b) Making human-machine system simulation a practical engineering tool: An Apex overview. In *Proceedings of the Third International Conference on Cognitive Modeling.* Veenendaal, The Netherlands: Universal Press. pp. 110-117.
13. Gong, R. & Kieras, D. (1994). A Validation of the GOMS Model Methodology in the Development of a Specialized, Commercial Software Application, *Proceedings of CHI, 1994*, New York: ACM Press, pp. 351-357.
14. Gray, W. D., & Boehm-Davis, D. A. (2000). Milliseconds matter: An introduction to microstrategies and to their use in describing and

predicting interactive behavior. *Journal of Experimental Psychology: Applied*, 6(4), 322-335.

15. 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* (3), pp. 237-309.
16. Hudson, S.E., John, B.E., Knudsen, K., & Byrne, M. D. (1999). A Tool for Creating Predictive Performance Models from User Interface Demonstrations. *Proceedings of the ACM Symposium on User Interface Software and Technology*, p.93-102
17. John, B. E. (1988) Contributions to Engineering Models of human-computer interaction. Ph.D. Thesis. Carnegie Mellon University.
18. John, B. E. (1990) Extensions of GOMS analyses to expert performance requiring perception of dynamic visual and auditory information. In proceedings of *CHI, 1990* (Seattle, Washington, April 30-May 4, 1990) ACM, New York, 107-115.
19. John, B. E. (1996) TYPIST: A Theory of Performance In Skilled Typing. *Human-Computer Interaction , 11* (4), pp.321-355.
20. John, B. E. & Gray, W. D. *GOMS Analyses for Parallel Activities*. Tutorial materials, presented at CHI, 1992 (Monterey, California, May 3- May 7, 1992), CHI, 1994 (Boston MA, April 24-28, 1994) and CHI, 1995 (Denver CO, May 7-11, 1995) ACM, New York.
21. John, B. E. & Kieras, D. E. (1996a). The GOMS family of user interface analysis techniques: Comparison and Contrast, *ACM Transactions on Computer-Human Interaction*, *3* (4), pp. 320-351.
22. John, B. E. & Kieras, D. E. (1996b) Using GOMS for user interface design and evaluation: Which technique?, *ACM Transactions on Computer-Human Interaction*, *3* (4), pp. 287-319.
23. Kieras, D. E. (1996) Guide to GOMS model usability evaluations using NGOMSL, *The Handbook of Human-Computer Interaction*, M. Helander and T.Landauer (Eds.), 2nd ed. North-Holland Amsterdam.
24. 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. *International Journal of Man-Machine Studies*, 22, 365-394.
25. Kitajima, M. & Polson, P. G. (1995) A comprehension-based model of correct performance and errors in skilled, display-based, human-computer interaction. *International Journal of Human-Computer Studies*, *43*(1):65-99.
26. Kosbie, D. S. & John, B. E. (1994). Hierarchical Event Histories and GOMS, *Poster at the Human-Computer Interaction Consortium Winter Workshop* (January, 1994, Frasier, Colorado.)
27. Lee, Adrienne Y., Polson, Peter G. & Bailey, Wayne A. (1989). Learning and Transfer of Measurement Tasks Performing Prediction: Predicting Performance, *Proceedings of ACM CHI'89 Conference on Human Factors in Computing Systems,* p.115-120.
28. Lerch, F. J., Mantei, M. M. & Olson, J. R. (1989). Translating Ideas into Action: Cognitive Analysis of Errors in Spreadsheet Formulas, *Proceedings of ACM CHI'89 Conference on Human Factors in Computing System*, pp. 121-126. New York: ACM.
29. Pirolli, P. and Card, S. K. (1999). Information foraging. *Psychological Review*, 106, 643-675.
30. Williams, K. E. (1993) Automating the cognitive task modeling process: An extension to GOMS for HCI. *In Proceedings of the Fifth International Conference on Human-Computer Interaction Poster Sessions: Abridged Proceedings* (vol 3. p. 182).
31. Young, R. M., Green, T. R. G., & Simon, T. (1989) Programmable user models for predictive evaluation of interface designs. In J. C. Chew & J. Whiteside (Eds) *Proceedings of ACM CHI'89 Conference on Human Factors in Computing Systems*, 15-19. ACM Press.
32. Young, R. M. & Wittington, J. E. (1990) Using a knowledge analysis to predict conceptual errors in text-editor usage. In J. C. Chew & J. Whiteside (Eds) *Proceedings of ACM CHI'90 Conference on Human Factors in Computing System*, 91-97. ACM Press.