Model-based Evaluation of Expert Cell Phone Menu Interaction

ROBERT ST. AMANT and THOMAS E. HORTON
North Carolina State University
FRANK E. RITTER
The Pennsylvania State University

We describe concepts to support the analysis of cell phone menu hierarchies, based on cognitive models of users and easy-to-use optimization techniques. We present an empirical study of user performance on five simple tasks of menu traversal on a cell phone. Two of the models applied to these tasks, based on GOMS and ACT-R, give good predictions of behavior. We use the empirically supported models to create an effective evaluation process for menu hierarchies. Our work makes three main contributions: a novel and timely study of a new, very common HCI task; new versions of existing models for accurately predicting performance; and a search procedure to generate menu hierarchies that reduce traversal time, in simulation studies, by about a third.

Categories and Subject Descriptors: H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—Evaluation/methodology

General Terms: Mobile telephones, menu traversal, cognitive modeling, evaluation

1. INTRODUCTION

There are 2 billion cellular telephones in use today, and this number is expected to reach 3 billion in 2008 [DiPrima 2006]. Cell phones are used for more than making calls; they now include tools for managing contact information, voice mail, and hardware settings, and often software for playing games, browsing the Web, and connecting to specialized information services. The market penetration of cell phones is much higher than that of conventional computers, which raises significant opportunities and challenges for HCI.

This research was supported by the National Science Foundation (IIS-0083281 and ITR-046852), the Space and Naval Warfare Systems Center, San Diego (N66001-1047-411F), and the Office of Naval Research (N00014-02-1-0021, N00014-03-1-0248, and N00014-06-1-0164). The information in this article does not necessarily reflect the position or policies of the U.S. government, and no official endorsement should be inferred.
The authors would like to thank three anonymous reviewers for their incisive comments, which significantly improved the analysis and discussion in this article.
Authors' addresses: Robert St. Amant, Department of Computer Science, North Carolina State University Raleigh, NC 27695; Thomas E. Horton, Department of Computer Science, North Carolina State University Raleigh, NC 27695; Frank E. Ritter, College of Information Sciences and Technology, The Pennsylvania State University, University Park, PA 16802.

The focus of this article is on techniques for evaluating cell phone usability, in particular the usability of the hierarchical menus that provide access to most functionality aside from dialing and data entry. While cell phone menu interfaces may appear simple at first glance, they pose a non-trivial design problem. Consider the menu hierarchy for the Kyocera 2325 cell phone, the first 25 items of which are shown in Table I. If we count as terminals those selections that open an application (e.g., a game), a list of data (e.g., recent calls), or a set of choices in the cell phone equivalent of a dialog box (e.g., for setting the ringer volume), then this hierarchy contains 98 terminals, reachable through 22 intermediate selections. The longest menu contains 12 items—all associated with the selection of different sounds. The shortest menu contains a single item, for entering a voice memo. Terminals in the hierarchy are up to four levels deep, and the mean number of actions to reach an item (scrolling plus selection), over all 98 terminals, is 13.3, taking on the order of 7 seconds for an experienced user.

This menu hierarchy is as large as that of a moderately-sized desktop application (e.g., Eudora 5.2 with 103 items). This is not unusual for cell phones; the menu hierarchy for the Samsung MM-A800, which includes a digital camera, contains a remarkable 583 items [Pogue 2005]. Designing menu systems for any platform, including desktop systems, can be challenging, but for cell phones the problem is made more difficult by several factors:

—Discrete selection actions in the form of button presses[1] are usually needed to move between menu items, because most cell phones lack more direct selection capabilities (e.g., a mouse or touch screen).

—Cell phone displays are small, allowing only a few menu items to be displayed at a single time. Many cell phones lack functionality for paging up or down, making display limitations even more significant.

—There is less standardization in hardware supporting menu traversal for cell phones than for desktop machines. Some phones have two-way directional buttons, others four-way; some have a labeled “Menu” button, while others rely on a button with overloaded functionality. Button placement can vary significantly, with “Cancel” and “OK” buttons reversed from one phone to another. If interfaces are developed for the lowest common denominator, independently of specific hardware (which is common practice at the mobile application level), then even cell phones with elaborate interaction support become less efficient.

These factors suggest that cell phone menu interfaces deserve close analysis, and that they need specialized techniques for their development and evaluation, which this article takes up in two parts.

In Section 2 we describe an empirical study of the traversal of cell phone menus, along with three models for predicting user performance: a Fitts’ law model [Fitts 1954], a GOMS model [John and Kieras 1996a; 1996b; Kieras 1999a], and an ACT-R model [Anderson et al. 2004]. All the models give good accounts of qualitative patterns in user behavior, and the latter two models give good to very good quantitative predictions of behavior, at both aggregate and detailed levels of analysis. In Section 3 we use our empirical results to define a novel evaluation metric for the efficiency of cell phone menu traver-

[1] We use the terms button presses and key presses interchangeably.

Menu
- Contacts
  - View All <list>
  - Add New
    - Phone Number <entry>
    - Email Address <entry>
    - Street Address <entry>
    - URL <entry>
  - Find Name <entry>
  - Add Voice Dial <entry>
  - Speed Dial List <list>
  - Voice Dial List <list>
  - Business List <list>
  - Personal List <list>
  - Information <info>
- Messages
  - Voicemail <list>
  - Send New
    - Recent List <list>
    - Enter from Scratch <entry>
    - Contacts List <list>
  - Text InBox <list>
  - Net Alerts <app>
  - Text OutBox <list>
  - Filed <list>
  - Erase Msgs <choices>
  - ...

Table I. The first 25 of 120 elements in the Kyocera 2325 menu hierarchy

sal. We define a search procedure that generates improvements to a menu hierarchy with respect to a given set of characteristic user profiles.

This article makes several contributions to the HCI literature: a novel and timely study of a very common new HCI task (menu use on cell phones), new models for accurately predicting performance on this task, and a simple, theoretically motivated search procedure that generates menu hierarchies that reduce traversal time in simulation studies by a third, which should be generally applicable to all menu-based systems.

2. A PERFORMANCE STUDY

Our interest is in expert (i.e., practiced and error-free) use of cell phone menu systems. For control purposes it was not feasible to collect data from experienced users on their own cell phones, with all the potential differences in hardware and software. As a compromise, we had users practice a small number of tasks, so that all tasks could be remembered easily, and then carry them out on a single cell phone. Though restrictive, these conditions give a reasonable starting point for an empirical study and model validation.

We used a Kyocera 2325, as shown in Figure 1. At the top level of its internal menu, the Kyocera display shows a single selectable icon. The OK button selects the current item; on the four-way scrolling button, RIGHT and LEFT move through the item list horizontally. For lower-level menus, three items are displayed at a time, oriented vertically. Each new menu list is displayed with the top item highlighted. The OK button, on the left, is used to select the currently highlighted item in these menus, while the CLR button, on the right,

Fig. 1. Kyocera 2325

returns to the previous level in the hierarchy. The UP and DOWN regions of the four-way button move through the menu. Downward scrolling is incremental, with items appearing one at a time at the bottom of the screen.

2.1 Procedures

We recruited fourteen experienced cell phone users for our study, students who took part for course credit. The first two users acted as participants in a pilot phase of the experiment, in which software for data collection and analysis was tested and procedures were refined; their data were also used in developing (but not validating) the models described in later sections. The remaining twelve users, male undergraduates in computer science, provided the main body of data for the study. All were right handed. All but one used their right hand to hold the cell phone, and all used the thumb of the right hand to press keys.

To collect data, we recorded the tone produced by each key press as transmitted through the earphone jack of the cell phone. Collection was initiated by the first key pressed by the participant and ended with the last key pressed. The onset of each key press is detectable by a threshold test on the audio output waveform from the earphone jack, using software we wrote for this purpose. Each tone lasts approximately 0.095 s, during which time the display changes, before the key is released. System responses are much faster than key presses and are treated as occurring within elementary key press actions and not contributing to the duration of user actions.

Participants started with a practice stage, in which they familiarized themselves with the cell phone and its menu system. We gave each participant a paper form describing how five terminal menu items were to be reached, as shown in the first column of Table II. Each "$>$" represents a scrolling action, with commas separating consecutive selection actions. Reaching each of the terminal items (those at the end of each sequence) constituted a task in the study. Participants practiced each task until they could carry it out three times in a row without error.

Each trial in the study required reaching one of the five target terminal items without access to the paper form. Tasks were presented to participants in a randomized order. We obtained five correct trials per participant (i.e., without errors or extraneous actions), discarding fewer than 10 trials across all participants, less than 3% of the data. This means that our cleaned dataset contains only OK and RIGHT/DOWN key press actions, 2,280 observations in total (2,280 = 12 users $\times$ 5 repetitions $\times$ (10 + 9 + 3 + 8 + 8 actions per

Table II. Task duration in s, with mean (and standard deviation) shown

Task	N actions	Duration
Menu > > > Settings > > Sounds > Ringer Volume	10	4.954 (1.077)
Menu > > > > Tools & Games > > Tip Calculator	9	4.027 (0.921)
Menu, Contacts, View All	3	1.271 (0.412)
Menu > > > > Tools & Games, Scheduler, View Day	8	4.393 (0.971)
Menu > > > > > > Web Browser	8	3.391 (0.827)

task)).

Table II shows the mean duration per task, over all participants in the study. User performance is much slower than for single-level menu selection with a mouse on a standard desktop platform [Byrne 2001], which highlights the importance of specialized models for this task, as we discuss below.

2.2 Models of user behavior

We predicted user performance with three models, each supported by a considerable background literature. A Fitts' law model, a GOMS model, and an ACT-R model were developed independently of each other, based on data from one task carried out by one of the users in the pilot stage of the experiment.[2] The three models run in the same software framework that evolved over the course of this research. The framework provides a common specification of the Kyocera cell phone, including the sizes and positions of keys and the distances between them, as measured on the physical device. The framework also supports a common representation of the menu hierarchy shown in Table I. The models use the same software environment that includes a simulation of the cell phone's interface and produces output in a consistent form.

2.2.1 *A Fitts' law model.* Our model is based on MacKenzie's [2003] version of Fitts' Law for one finger typing for text entry on mobile phones. Movement time in seconds for thumb input is

$$MT = 0.176 + 0.064 \log_2 (D/W + 1), \tag{1}$$

where D represents the distance (amplitude) of the movement and W the width of the target. The value for D in our study was 14.5 mm, which separates the OK button and the DOWN button area, with widths W of 6 mm and 10 mm, as provided by the cell phone specification. This model, as with the other models described below, makes the simplifying assumption that all scrolling actions can be represented by DOWN key presses, even though the first action is a RIGHT key press, with a slightly different size and distance from the OK button.

To execute the Fitts' law model for each of the five tasks, a path is generated from the root of the menu hierarchy to the terminal item for the task. Each step on the path is associated with a movement action or a key press action. Durations for all the steps are accumulated to produce an overall task duration.

2.2.2 *A GOMS model.* The second model is a GOMS model [Kieras 1999a; John 2003]. GOMS methods for task analysis produce hierarchical descriptions of methods

[2]Preliminary versions of the GOMS and ACT-R models described in an earlier conference paper [St. Amant et al. 2004b] contained minor inconsistencies; these inconsistencies were removed in revision. Performance was altered by no more than a few percentage points. The qualitative behavior of the models and comparisons between them remain unchanged from their earlier description.

Method for TraversalTo RingerVolume

Step.	Look for object whose label is "Menu".	
Step.	Accomplish goal: Select Menu.	
Step.	Accomplish goal: Select Settings.	
Step.	Accomplish goal: Select Sounds.	
Step.	Accomplish goal: Select RingerVolume.	
Step.	Return with goal accomplished.	

Method for Select Settings

Step Scroll-Test-1.	If (*Settings* not found and not on-scroll-button)	
	Move-to scroll-button.	0.050 s or 0.133 s
Step Scroll.	Press scroll-button; Goto Scroll-Test-1.	0.330 s
Step Scroll-Test-2.	If (*Settings* not found and on-scroll-button)	
	Press scroll-button; Goto Scroll-Test-1.	0.050 s or 0.330 s
Step OK-Test-1.	If (*Settings* found and on-OK-button)	
	Press OK; goto End.	0.050 s or 0.330 s
Step OK-Test-2.	If (*Settings* found and not on-OK-button)	
	Move-to OK-button.	0.050 s or 0.163 s
Step OK.	Press OK; Goto End.	0.330 s
Step End.	Return with goal accomplished.	0.050 s

Fig. 2. The GOMS method for selecting the Ringer Volume item, and a sample single-level menu selection for the *Settings* item

and operators needed to accomplish goals; some GOMS models have been strikingly successful in critical HCI domains [Gray et al. 1993]. In our model a method is defined for each task in the study. All of the methods are automatically generated from the menu hierarchy specification, based on the same path traversals used for the Fitts' law model. Within a method, each step corresponds to the selection of a menu item. The GOMS method for selecting the terminal item Ringer Volume is shown at the top of Figure 2.

Each of the steps in this method in turn decomposes into a selection method, such as Select Menu or Select Sound, which involves scrolling until a specific item in the sequence is reached—selection in a menu at a single level. There is one selection method for each menu item, from Select Settings to Select Ringer Volume. All of the selection methods have the same form, as shown in the example at the bottom of Figure 2. Specifications of these lower-level methods are created automatically from a generic template.

Processing in a selection method involves iterating through a sequence of four exhaustive tests of whether or not the target intermediate or terminal item is currently highlighted and whether the finger is on the appropriate key for selection or scrolling. The durations of the steps follow the guidelines established by Kieras [1999a] in his work on GOMSL and GLEAN3. Each test in a decision step requires 0.050 s, plus the time to execute any actions in the body of the decision if the test succeeds. Steps that involve key presses last 0.280 seconds plus the duration of tests or auxiliary operations (0.330 seconds in total). Moving to the DOWN key lasts 0.083 seconds (0.133 seconds in total); moving to the OK key lasts 0.113 seconds (0.163 seconds in total). Movement times are based on the movement component of the Fitts' law model in the previous section. The model assumes negligible system response time and that there are no verification steps. Further, the initial visual action to acquire the first menu item occurs before the first key press (timing begins at the first key press), and as the highlighted menu item changes no visual re-acquisition is needed during selection or scrolling activity. Processing is entirely sequential, with no overlapping of steps.

Modeling results, based on the description above, are generated by a GOMS interpreter that we implemented specifically for this project. While there would have been some benefit to using existing GOMS modeling tools and environments (e.g., GLEAN3 [Kieras 1999a]), we judged that the value of a single simulation and modeling framework (despite its limitations), for the Fitts' law, GOMS, and ACT-R models, would provide a worthwhile degree of consistency across our evaluation.

2.2.3 *An ACT-R model.* The third model is based on the ACT-R 5.0 cognitive architecture [Anderson et al. 2004]. We picked ACT-R as a representative cognitive modeling architecture and as a common choice in HCI work. ACT-R integrates theories of cognition, visual attention, and motor movement and has been the basis for a number of models in HCI (e.g., [Ritter and Young 2001]). ACT-R models simulate the time course and information processing of cognitive mechanisms, such as changes of attention and memory retrievals, as well as external actions, such as movement of the fingers. Roughly speaking, ACT-R models provide details that can explain behavior in cognitive terms at a level not addressed by the coarser GOMS representation.

In our ACT-R model, a virtual (simulated) display maintains a representation of the items in the cell phone's menu interface hierarchy. Menu items are presented in a vertical list, and one of the menu items is always highlighted. All items are presented for each list, independent of the physical display size. When an item is selected in the virtual display, the list is refreshed with the appropriate submenu.

The model's memory is initialized with a set of declarative memory chunks that represent the parent-child relationships between the intermediate menu items needed to reach terminal items. For example, for the Ringer Volume task, pairs of X's for the Menu/ Settings, Settings/Sounds, and Sounds/Ringer Volume relationships are included. Chunks representing the parent-child relationships are generated automatically via traversal of the menu hierarchy specification. ACT-R's model of the hand is initialized with the thumb on the OK button.

Procedural knowledge in the ACT-R model consists of eleven productions:

—*Find-top-item* searches the visual field for a highlighted menu item immediately after a selection action.

—*Find-next-item* searches for the next item below the one currently attended, immediately after a scrolling item.

—*Attend-item* causes the location of the highlighted item to be visually attended.

—*Encode-item* encodes the text for the attended menu item, so that its content (i.e., the name of the item in text form) becomes accessible to the model.

—*Respond-select-target* fires when the currently highlighted item is recognized as the terminal item.

—*Recall-item-association* retrieves an association, if it exists, between the currently highlighted menu item and its subordinate item along the path to the terminal item.

—*Respond-select-ancestor* recognizes an intermediate menu item along the path to the terminal item (i.e., one of its ancestors).

—*Respond-continue-down* fires when the highlighted item is neither the next item to be selected nor along the path to the terminal item.

—*Move-down* causes the motor module to press the Down key.

—*Select-target* causes the motor module to press the OK key on the terminal menu item, ending model execution.

—*Select-ancestor* causes the motor module to press the OK key on an intermediate menu item.

The model starts with the goal of selecting a specific terminal menu item. The simulation environment shows a single highlighted item. The model first retrieves a target intermediate item to be selected. It then searches for the currently highlighted menu item in its field of view. Once found, the visible item is attended and then encoded, so that its text representation becomes accessible. If the text matches the target item and this is the same as the terminal item, then the model initiates motor actions to move the thumb to the OK button (if necessary) and press it. Model execution completes at this point. If the text matches the target item but it is not the terminal item, then this means that the currently highlighted item is on the path to the terminal. The OK button is pressed and another target item is retrieved from memory. Visual processing repeats as before. If the text of the highlighted item does not match the target item, then motor actions are initiated to move the thumb to the DOWN button and press it. Control is transferred to the visual search action, as before. In the model, manual scrolling actions can trail behind visual processing by an unspecified amount (determined by processing in the model such as memory retrievals); the visual and manual modules become synchronized when a new menu is presented. User errors, such as pressing an incorrect key, are not modeled. Model execution is deterministic, with no noise parameters used.

Our model is defined in the ACT-R modeling language, but its execution depends on an extension to perceptual-motor processing in the architecture. The perceptual and motor components of ACT-R 5.0 have some bias toward desktop activities, such as selecting menu items with the mouse and navigating through windows and dialog boxes [Anderson et al. 2004; Byrne 2001]. In ACT-R, the keyboard is represented as an array of locations. Neighboring keys are a unit distance apart in a rectilinear arrangement, and each key has unit width and height. Standard key presses are modeled as finger movements from locations on the home row to the location of a target key. To handle interaction with a cell phone keypad, more flexibility is needed in models of finger movements and the keyboard. We extended the ACT-R environment representation to support a layout-based keypad in which the size and placement of keys can be specified individually. The new representation allows us to build specifications of different cell phone keypads that can be integrated with ACT-R motor processing in a straightforward way. Fingers are modeled as moving between locations, which in the case of this experiment are key locations, but may be arbitrary if needed.

2.3 Model performance

We can describe the performance of the models at two levels: the accuracy with which the models predict the overall duration of tasks, and the accuracy of their predictions of the duration of individual actions. These two levels are discussed in the sections below. Other factors commonly explored by modeling, such as learning behavior and the occurrence of errors, are excluded by the design of the experiment.

2.3.1 *Task-level performance.* Table III shows summary model performance and user data broken down by task. Figures 3 through 7 show more detailed views of the same data in graphical form. Both the GOMS and ACT-R models give good approximations

Table III. Model performance across tasks, in s. Upper and lower 99% confidence intervals are shown in parentheses for user data.

Task	User Performance	GOMS	ACT-R	Fitts' law
Ringer Volume	4.954 (4.584 5.325)	5.059	5.267	2.173
Tip Calculator	4.027 (3.710 4.343)	4.233	4.482	1.801
View All	1.271 (1.129 1.413)	1.160	1.280	0.352
View Day	4.393 (4.059 4.727)	3.707	3.883	1.428
Web Browser	3.391 (3.107 3.676)	3.407	3.883	1.428

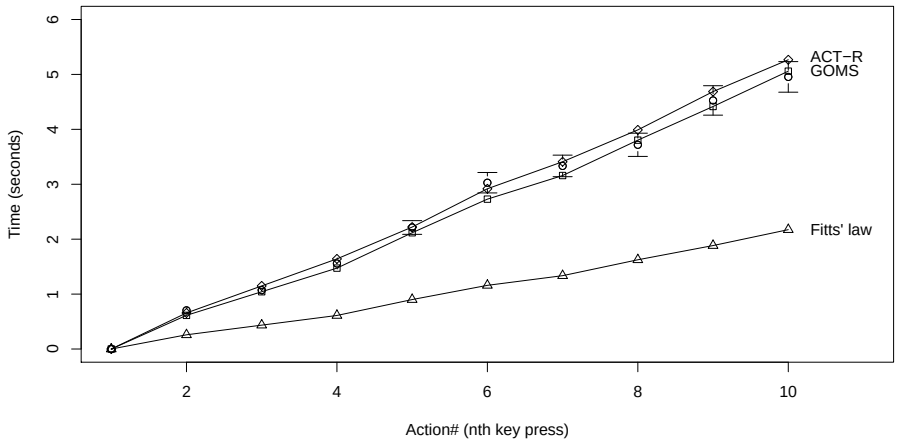

Fig. 3. Model predictions and user data by task: Ringer Volume. Each user data point (shown with a circle), plus 99% confidence interval, represents the mean of 60 sample points. Confidence intervals smaller than 200 ms are not shown.

of user performance. GOMS predictions are within the 99% confidence interval for mean overall task duration for all target items except View Day. ACT-R predictions are within this interval for two of the target items.

The Fitts' law model does less well, for reasons that are worth discussing. Many models of cell phone interaction, such as keypad dialing and one-finger text entry [MacKenzie 2003], have been based on Fitts' law, which motivated this aspect of our evaluation. Our Fitts' law model performs relatively poorly, despite the success of such models elsewhere. The Fitts' law model produces times that are about half of the observed times. This is not surprising—much of the activity of this menu selection task is outside the scope of the model. Silfverberg et al. [2000] describe a comparable example of where Fitts' law models break down, in a discussion of text entry on mobile phones. For some cell phones, text entry is aided by lexicon-based word disambiguation. While typing, the user ordinarily refers to the display in order to decide whether the system has correctly disambiguated the word being typed. In text entry, such cognitive processing may not be needed by expert users familiar with the disambiguation system. In this menu selection task, however, we assume that users confirm their actions. In other words, significant visual and cognitive processing is necessary at each step in the process, but this is not captured by the Fitts' law model (though it is represented in the GOMS and ACT-R models).

The consistent linearity of user performance across tasks in Figures 3 through 7 suggests

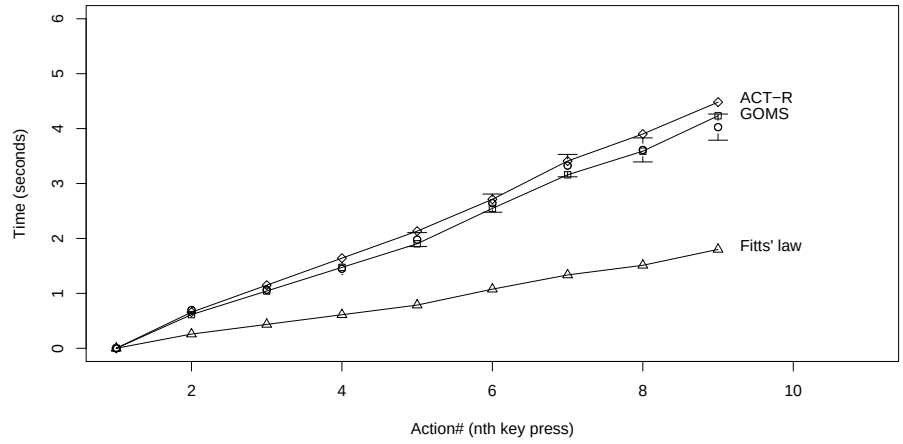

Fig. 4. Model predictions and user data by task: Tip Calculator

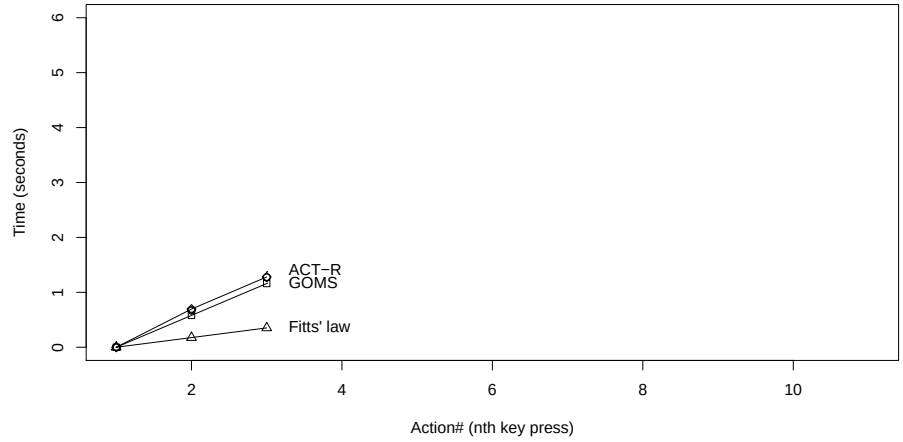

Fig. 5. Model predictions and user data by task: View All

a straightforward way to measure the predictive power of the GOMS and ACT-R models through comparison with a least squares linear model.

Because the models' predictions apply to each task, the harshest way to test them is to compare them to a linear regression model fit to data from each task. Table IV shows the coefficient of determination, R^2, for these linear models for each button press. The linear model predicts time T by the number of key presses k plus a constant. The remaining columns show analogous values for the ACT-R and GOMS models.

A linear model can also be fit to the aggregation of performance data for all tasks and all users:

$$T = -0.500 + 0.531k. \tag{2}$$

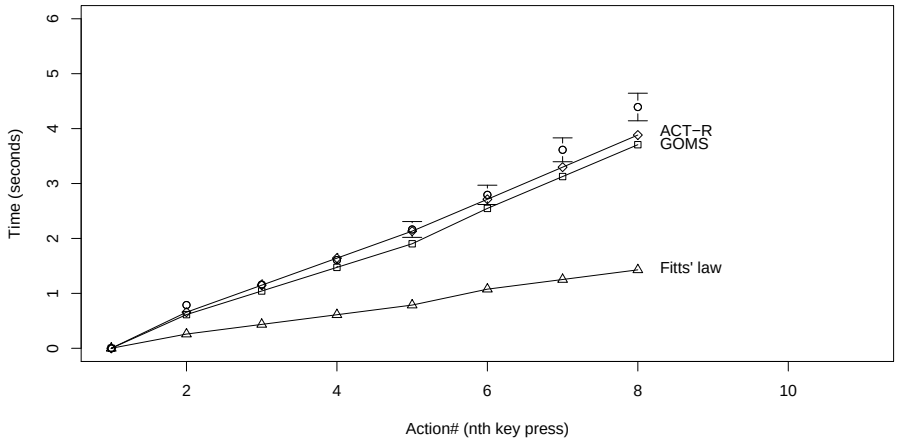

Fig. 6. Model predictions and user data by task: View Day

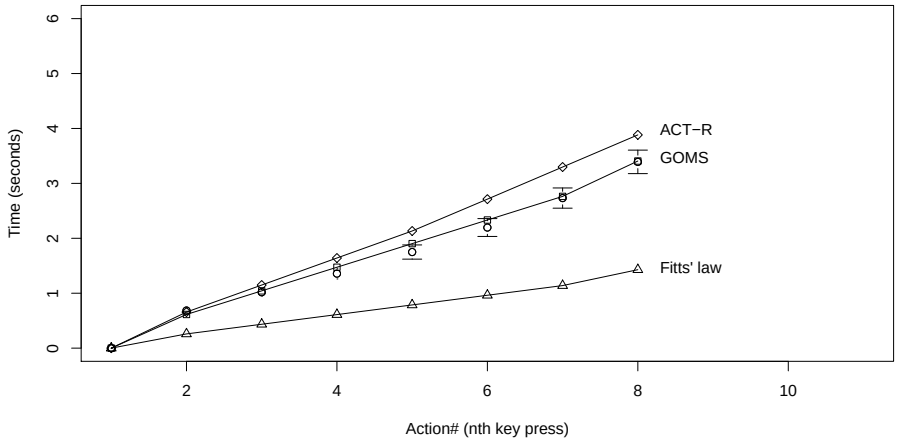

Fig. 7. Model predictions and user data by task: Web Browser

This aggregate linear model has an R^2 of 0.834. Although neither the GOMS nor the ACT-R model accounts for as much variance as a general linear equation, both are close; the comparable values are 0.809 for the GOMS model and 0.796 for the ACT-R model.

The aggregate linear model has appealing conceptual and computational simplicity; we use Equation 2 in Section 3 for just this reason. As a general model of performance, though, it has several shortcomings in comparison with the GOMS model, the ACT-R model, and even the Fitts' law model. First, the latter are *a priori* models—they were not tuned specifically to the data. Second, as we discuss later in this section, the models give predictions at a more detailed level than the linear model can provide. Third, and most important, the GOMS, ACT-R, and Fitts' law models have theoretical underpinnings that give them ex-

Table IV. Variance accounted for by a linear least squares model, the GOMS model, and the ACT-R model

Task	Linear	GOMS	ACT-R
Ringer Volume	0.846	0.845	0.843
Tip Calculator	0.831	0.832	0.817
View All	0.787	0.768	0.787
View Day	0.836	0.797	0.826
Web Browser	0.800	0.803	0.709

planatory power. In the case of GOMS, performance is explained by the specific tasks that are represented, the hierarchical structure in which they are combined, and dependence on a cognitive processing framework that provides specific timing predictions (e.g., for Fitts' law movements). The ACT-R model extends the level of detail in its explanations, in accounting for the interval between actions by explicit visual processing and memory retrievals, and in modeling visual processing and motor actions as proceeding in parallel for scrolling actions but synchronizing with selection actions. This allows performance on multiple tasks to be based on models of single tasks (e.g., [Salvucci 2001]).

Like all model-generated explanations, these are provisional and subject to further testing. In particular, the predictiveness of a linear model raises a warning flag: because task-level performance accumulates the durations of actions in sequence, almost any reasonable cumulative function is likely to have the same qualitative shape. The accuracy of the models over tasks of different durations suggests that the models have some generality, but this is far from conclusive. If, as with computing, the purpose of modeling is insight rather than numbers [Hamming 1962], we should look more closely at our results.

2.3.2 *Action-level performance.* Table V shows the predictions of the mean time between user key presses that each model makes over all the menu selection tasks. There are three different categories: all actions aggregated over all tasks, only selection actions, and only scrolling actions. The ACT-R and GOMS models both provide good predictions at this level, with differences of at most 0.045 s, about 8% error. Although the Fitts' law model is qualitatively correct in predicting that selection actions take longer than scrolling actions, it underpredicts user reaction time in all categories—our discussion in this section will therefore mainly focus on the ACT-R and GOMS models.

The results in Table V are limited in two ways. First, they distinguish only between classes of actions in the abstract, independent of the task context in which they are executed. Second, the results neglect the inherent variance in user performance. Even under identical task conditions, the actions of different users (or a single user in different trials) may have different durations. We address these two limitations in the remainder of this section.

One way to describe the execution of a menu navigation task is as a simple repeated pattern: each task is carried out through a number of scrolling actions followed by a selection action, repeated until a terminal item is reached. We define a *selection run* as a sequence of scrolling actions leading up to a selection action. As in our per-task analysis, user performance is basically linear for the overall duration of selection runs and is well-predicted by both the GOMS and ACT-R models with respect to overall duration. What is more interesting is differences in duration for actions of the same type. In selection runs, the first scrolling action after a selection action lasts much longer than subsequent scrolling

Table V. Mean key press duration across tasks, in s. Upper and lower 99% confidence intervals are shown in parentheses for user data.

	Mean User Performance	GOMS	ACT-R	Fitts' law
All actions	0.547 (0.532 0.561)	0.532	0.570	0.218
Scrolling	0.515 (0.497 0.533)	0.488	0.558	0.202
Selection	0.610 (0.584 0.634)	0.620	0.592	0.248

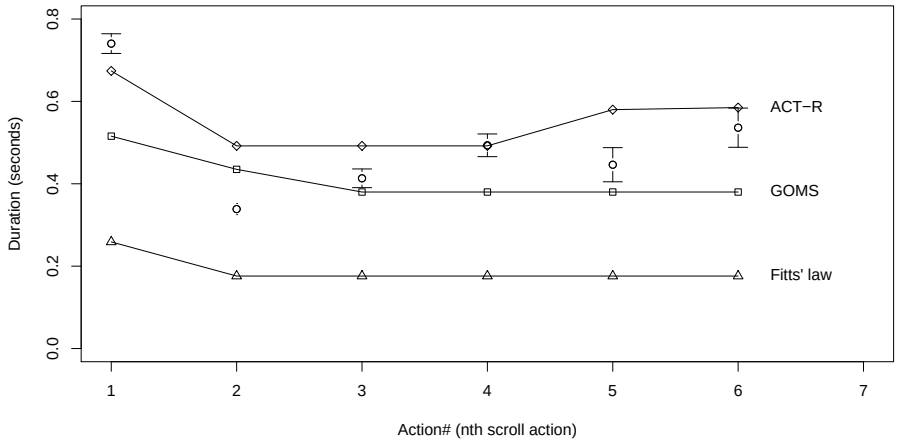

Fig. 8. Differences in the duration of scrolling actions during selection runs of different length, with model predictions. 99% confidence intervals are shown for user data points. The confidence interval for Action 2, not shown, is less than 30 ms.

actions, as shown in Figure 8. A general linear model (see Equation 2) would be a straight line at 531 ms per action.

Two factors appear to contribute to the longer duration of the first scrolling action. One factor is movement time, in that for the first scrolling action, the thumb must move from the OK key to the DOWN button; movement between keys is unnecessary for further scrolling. The other factor is visual processing: each time that a selection action takes place, a new set of menu items appears on the display and must be read. All of the models include a movement component and thus reflect the general qualitative pattern, but they vary in how they handle visual processing. The ACT-R model carries out an explicit visual search, which occurs in parallel with the motor movement. The GOMS model does not include an explicit visual processing step, but each selection entails decision-making tests as well as a call to a new method, adding to the duration of motor movement. Both models produce slight underpredictions of the duration of the first scrolling action.

A less obvious pattern is also present in Figure 8. For users, the first scrolling action lasts the longest, the second the shortest, and all succeeding actions in between. We believe that three factors explain the increase in duration between the second and remaining actions. The first factor is an environmental constraint on users' visual processing. Because only three menu items are presented on the display at a time, we can expect the duration of the fourth and succeeding scrolling actions to be slower than the second and third, because the items to be traversed are not immediately available for visual processing. The second factor is a possible strategy that users took in dealing with menus that are known, via

practice, to be long: users may quickly execute several scrolling actions with less attention to the display, until the approximate region of the target item is reached. The third factor is parallelism in visual processing and motor action. Related experiments on menu selection with a mouse [Byrne 2001] suggest that eye movement is not strictly synchronized with motor activity in the discrete menu traversal actions of our domain; this means that the eyes may scan ahead of the items being highlighted by button presses. The first factor is not reproduced by our simulation environment and the second is not yet included in the model. The third factor, parallelism in motor and visual behavior, is represented in the ACT-R model. This parallelism accounts for the increase in duration after the fourth scrolling action, as the duration of motor actions dominates that of visual processing. The exact point in the user data at which the increase occurs is not captured by any of these models; neither is the remaining variability in the duration of scrolling actions.

Finally, we note that the cognitive and visual processing component of actions in selection runs is much higher than the movement component. The Fitt's law model provides a baseline for movement-only duration, but underpredicts the duration of user actions by a factor of two to three. The GOMS and ACT-R models, for reasons discussed above, come much closer to user performance. Overall, the mean error for mean action duration during all selection runs is about 14% for the GOMS model, 19% for the ACT-R model, and 60% for the Fitts' law model.

Our results so far show that the GOMS and ACT-R models give accurate predictions of the mean duration of user actions, when actions are grouped into classes or, as in the analysis of selection runs, associated with a specific task context. These predictions, however, give no information about the accuracy of the models for specific instances of actions. In evaluating model predictions at the action level, the issue of inherent variability in user performance is perhaps the most important.

Two common measures of error that give insight into this issue are root mean squared error (RMSE) and mean absolute error (MAE). Table VI shows RMSE and MAE for the three models, for all actions and for the categories of scrolling and selection actions. The prediction error for the GOMS and ACT-R models, per action duration, is much higher than for mean duration, about 42% of the mean action duration (as given in Table V) for the RMSE measure and 32% for MAE. The values are similar for the subcategories of scrolling and selection actions. By these measures, the models are very close in performance, with neither GOMS or ACT-R having an obvious advantage.

While these values of RMSE and MAE are disappointingly high for the GOMS and ACT-R models, it is worth asking how much they might be improved. As in our task-level analysis, we can define a post hoc model, based on the user data, for comparison with the models we have built. We begin by observing that the predictions of the models abstract away performance differences between individual users and across trials. For example, a model will give the same predicted duration for the sixth action in the Ringer Volume task, regardless of which of the twelve users or which of their five trials is involved. The variance in the 60 data points per unique action, in context, gives rise to the error measured in Table VI. How well would an optimal model perform? For a sample of data points, the best estimator with respect to mean squared error is simply the mean of the sample. The best post hoc model (with respect to least squares error) thus returns the mean duration for each unique action, over users and trials. The error for this model is shown in the MD (Mean Duration) columns of Table VI. These errors are lower than but still relatively

Table VI. Model prediction errors for action duration, in seconds, for the three models and for a mean duration model

	Root Mean Squared Error				Mean Absolute Error			
	GOMS	**ACT-R**	**Fitts' law**	**MD**	**GOMS**	**ACT-R**	**Fitts' law**	**MD**
All actions	0.230	0.228	0.411	0.192	0.175	0.177	0.330	0.142
Scrolling	0.218	0.215	0.390	0.188	0.159	0.170	0.313	0.138
Selection	0.248	0.261	0.452	0.208	0.205	0.190	0.362	0.151

close to those produced by the GOMS and ACT-R models. As percentages of mean action duration, the models might improve from 42% to 35% with respect to RMSE and from 32% to 26% for MAE. This comparison suggests that the GOMS and ACT-R models are performing almost as well as is possible in predicting user behavior at the action level, given the variance in the user data.

2.4 Discussion

All of the models we have presented have proved robust in our analysis, though at a sufficiently detailed level they break down (as all models do). Our results indicate that detailed, rigorous models of low-level interaction with cell phones is possible, and that such models make good predictions. Aside from the use of this work as a possible exemplar of the application of cognitive modeling techniques to HCI evaluation we can note a few observations.

Modelers need to consider the tradeoff between modeling effort and the value of increasingly veridical results. The GOMS model developed here is as good as or better than the ACT-R model, and was much cheaper to build. For modeling efficiency, a reasonable heuristic is to apply simple formalisms to model simple procedures. This is especially relevant if the simple formalism can predict all the observable information or all the needed behavior. All the data we have in this study (keystroke times by expert users) can be predicted by both ACT-R and GOMS, though in other situations (e.g., if we had eye-tracking data and wanted to predict eye movements), GOMS would be at a distinct disadvantage.

Further, GOMS offers considerable flexibility in modeling. A coarser formalism does not necessarily imply stricter constraints on modeling, which is perhaps an unintuitive observation; rather, the reverse can be the case. In our GOMS model, for example, the specific ordering of decision steps, as shown in Figure 2, is not governed by cognitive constraints. A different ordering (e.g., one that tested whether an OK key press was appropriate before rather than after scrolling) would have produced different predictions. It turns out, in our case, that user behavior is sufficiently regular that the GOMS model we developed for a single user's behavior generalized very well to a larger sample; if this had not been the case, the modeling flexibility we describe would not have been helpful. Our ACT-R model, for the same task, does not allow such direct fine-tuning to be carried out in the same way, because of tighter architectural constraints on the interactions between visual and motor actions.

The remaining differences between the models' predictions and the data suggest further improvements to the models are possible. Most importantly, the comparison in Figure 8 shows that only the ACT-R model starts to account for the faster second keystroke, and none of the models predict this (or the later changes) very well.

There are limitations to this work so far, aside from model performance. For example, many cell phones have additional interaction features, such as shortcut menus and non-linear graphical icon displays, that are not captured by the models we have built. Further

studies, perhaps extending to include novice users, could take error types and error distributions into account, to help extend the range of application of these models. We believe, nevertheless, that our work lays out clear directions for future research. One issue we have begun to explore is the performance differences between the GOMS and ACT-R models. As can be seen in the evolution of cognitive modeling architectures such as ACT-R and EPIC [Kieras and Meyer 1997], there is considerable overlap in basic assumptions about the way that perceptual-motor constraints should be modeled [Byrne 2001; Kieras 2002], and so it is not unreasonable that the models produce similar predictions.

Nevertheless, because ACT-R represents behavior at a greater level of detail than GOMS, the ACT-R model is capable of more detailed performance predictions than the GOMS model. That GOMS outperforms ACT-R in some areas of our study is disappointing from a cognitive modeling standpoint, but not entirely unexpected, for the reasons described above. Further, the models were developed independently of each other, and different modeling paradigms and modelers can lead to different opportunities for modeling error [Ritter 1991]. There has been recent work toward automatically translating between models at different levels of abstraction, but this research is in its early stages [John et al. 2004; St. Amant et al. 2004a; St. Amant and Ritter 2004; Salvucci and Lee 2003].

3. USER PROFILES AND SEARCH

Once models of menu traversal have been built, the models can be applied toward improving menu hierarchies so that end users can traverse menus more quickly. This is a key concern for developers who may be less interested in modeling theory or model development than in the pragmatic issues of increasing usability.

An evaluation of a menu hierarchy independent of usage patterns would be uninformative: different users choose different items, and items are chosen with varying frequency. In other words, different usage patterns favor different designs. We define a *user profile* to be a probability distribution over the set of terminal items in a menu hierarchy that specifies the probability of each terminal being chosen, relative to the entire set. Each user profile is also associated with the *frequency* that the menu system is accessed. For the entire population of users of the menu hierarchy, there may be many different user profiles, some more common than others, a distribution captured by the *coverage* of individual profiles. As an example, imagine that 20% of the users of a given cell phone access only two items, Recent Calls and View All Contacts, each on average twice a day. In the probability distribution of the profile for this set of users, these two items have probability 0.5 and all others have 0.0; the coverage of the profile is 0.20; and its frequency is 4 (a value that becomes meaningful in the context of the per-day usage values of other profiles).

In formal terms, the design problem involves the construction of a mapping (in the form of a hierarchical ordering h) between T, the set of terminal menu items, and U, the set of all user profiles defined on T. A reasonable evaluation measure for a given menu hierarchy h is its efficiency: the expected cost of reaching a terminal item. This turns out to be straightforward to represent. Expected cost is given by

$$EC(h) = \sum_{t \in T} p(t) c_h(t), \tag{3}$$

where $p(t)$ is the probability of the occurrence of a specific terminal t, and $c_h(t)$ is the cost of reaching terminal t in hierarchy h.

In some situations it may be possible to estimate $p(t)$ directly through usage statistics across user profiles. This would mean maintaining a local log of menu selections on individual cell phones, to be uploaded opportunistically to a central repository, or making these local actions visible remotely as they are carried out. If this is not practical due to storage or bandwidth constraints, an alternative is possible. We can express the probability $p(t)$ as follows:

$$p(t) = \sum_{u \in U} p(t|u)p(u). \tag{4}$$

That is, the probability of the occurrence of t is the conditional probability of its occurrence in a specific user profile u, scaled by the probability of u and summed over all user profiles. The conditional probability $p(t|u)$ is given by the distribution associated with each user profile as described above. Values for $p(u)$ can be estimated from the coverage and frequency of a profile at the time the profile is assigned to a user. In practice, we can imagine individual users being asked questions about their cell phone usage when they are assigned to a specific user profile: how often they will access their cell phone's menu system and the types of functions they expect to use. The tradeoff, compared with direct sampling of $p(t)$, is between accuracy and resource demands.

All that remains is to define a specific cost function c_h, which we can do with our study results. For pragmatic reasons, we use the easiest metric available to compute cost, the linear regression given in Equation 2 (the GOMS or ACT-R model could have been used, with comparable accuracy but with a significant increase in processing time). The factors that make the linear regression less appropriate for modeling do not apply here. Our choice for c_h means that $EC(h)$ produces the expected duration of choosing an arbitrary terminal menu item in hierarchy h.

This measure can be used by an automated search algorithm to identify alternative designs of the menu hierarchy that improve user performance. A complication is that the automated modification of a menu hierarchy cannot arbitrarily rearrange structure purely for efficiency. Changes must respect the semantic relationships between the items. That is, the item Ringer Volume is under the Settings category rather than vice versa for good reason. To avoid the difficulties of representing and reasoning about menu item semantics (we leave this for future work), we rely on two search operators that produce only small changes. For a terminal item with non-zero probability, these operators can be applied:

—*Promote item* moves an item to the beginning of its menu list, to reduce scrolling time.

—*Promote subtree* moves an ancestor of the item up one level in the hierarchy, to reduce the number of intermediate items that must be selected to reach the terminal.

An item or subtree rooted at an ancestor may only be promoted once. Even with these constraints, the search space size is exponential in the number of target items with non-zero probability in any profile (e.g., if all non-zero items in a user profile are in one menu list, then all permutations of these items will be considered). Exhaustive search is thus impractical for the phone hierarchy shown in Table I; for just the menu containing 12 items mentioned in Section 1, half a billion permutations are possible. A best-first search algorithm, however, gives good results after as few as 100 steps.

Table VII. Menu traversal times, in seconds, showing the effect of the optimization of menu structure

Profile size	Initial cost	Final cost	Savings
20	7.325	4.530	37.5%
30	6.962	4.762	31.5%
40	7.009	4.940	29.4%

3.1 Results

Ideally, we would be able to validate the search procedure based on real user profiles found in the most commonly used cell phones. We have been unable to acquire such data, unfortunately. Lacking real cell phone user profiles, we can only illustrate the search procedure in practice, but our results are promising. Based on the Kyocera menu hierarchy, we defined random profiles of different sizes, where size refers to the number of non-zero probability menu items contained in the profile. The probabilities for each profile were drawn from a uniform random distribution and normalized. Because these profiles were randomly generated, we used only a single profile for the search, rather than composing arbitrary probabilities from different random profiles. These profiles approximate profiles for spreadsheet usage [Napier et al. 1992] and modeling languages [Nichols and Ritter 1995].

Table VII shows the results for user profiles of size 20, 30, and 40 terminals. In each case, 10 different random profiles were generated for each size, and a best-first search, bounded at 500 steps, was applied to produce improvements. The cost values are means of the time estimates produced by the linear model. The last column gives the time savings in traversing the reordered menus, as a percentage of the duration of the traversals in the original menu hierarchy.

Because these results are based on random probabilities of accessing menu items, rather than actual user experiences, they can only be viewed as suggestive. Anecdotal evidence from industry contacts indicates that performing usability studies on menu hierarchies is not common practice. We expect that with improvements in data collection, however, this approach may help to make cell phones more efficient in the future. Targets for future research include examining the plausibility of a uniform distribution for selectable menu items in user profiles, more efficient search to optimize menu layouts, application to other types of menu layouts, and the inclusion of other factors (e.g., profile size) in cost computations.

3.2 Discussion

We have presented formulas and a search algorithm to show how menu efficiency can be improved by about a third. The modifications to the menu hierarchy produced by the search have the effect of reducing the depth of the hierarchy and increasing the length of individual menus. This was a simple change, but clearly one that could be applied to at least one commercially available phone. It could plausibly be applied to other systems with similar menu structures.

The general approach laid out in this section is related to two areas of HCI other than cognitive modeling, both of which provide opportunities for further research. The first area is adaptive user interfaces. The issue of finding the best menu hierarchy for a given user profile is separate from that of deciding when the menu structure should be put in place. Our discussion in this section assumes that a static hierarchy is associated with each user

profile, even if new usage data were to become available over time. If such data were recorded over time, if possible for individual users, then a new search could be carried out incrementally to find improved menu hierarchies. This function should not be performed lightly, but one now quite real possibility is to treat the automated adaptation of the menu hierarchy as a customization option that users can select at their own discretion, whenever they choose. It should also be possible to incorporate a theory of learning that could predict when to do this and the costs involved in learning the new menu structure.

The second related area is support for navigation. A menu hierarchy is a small, restricted information space in comparison with other spaces such as the World Wide Web. The modifications explored by the search procedure are only a small subset of possible transformations that might be applied to an interface. Nevertheless, some of the same conceptual issues apply to the analysis of navigation in general. For example, imagine a set of recommendations for improving navigation on a web site that promote links upward toward the site entry page and move specific links closer to the top of their pages [Ritter et al. 2005].

In practice, the most effective approach to navigation redesign addresses the semantics of the information space rather than focusing only on its surface organization and presentation [Young 1998]. For menu hierarchy modification, this implies that greater potential benefits can be gained from examining the semantic relationships between menu categories and menu items than their ordering.

The most relevant research along these lines is Pirolli's work on optimal-foraging theory and information scent [Pirolli 1997; 2003]. Optimal-foraging theory explains behavior adaptations in terms of resource availability and constraints. In its application to menu navigation, information scent is a metaphor for visible semantic cues that lead users to information they seek. Pirolli has developed an extension of ACT-R, called ACT-IF, to evaluate a foraging model of information navigation. ACT-IF relies on a spreading activation network to capture associations in memory processing. The models described in this article are based on the assumption that associations between menu items such as Sounds and Ringer Volume can be directly retrieved from memory by an expert user. A more general model, based on ACT-IF, might be able to explain the strength of these associations, based on measures of semantic distance. With such flexibility in representation, it would be possible to explore additional modeling issues, such as how novice users might traverse an unfamiliar menu hierarchy, which paths through the hierarchy are more likely to result in errors, and how renaming or recategorizing menu items could influence navigation performance more than just reordering.

4. CONCLUSION

In this article we have described a set of evaluation concepts and tools to support cell phone menu design. The GOMS model is able to predict user performance very well. The ACT-R model performs almost as well. It took more effort to create, but also provided more detailed predictions and could be used for a wider range of analyses. Although our work has relied on a simpler performance model, both of these models could be used by a simple, efficient algorithm to optimize the redesign of cell phone menus. The redesign could let users on average perform their tasks about 30% faster, based on plausible assumptions about usage.

This menu redesign approach is simple; we believe it issimple enough to be taught to

and used by designers. This approach is based on knowing users (through the models) and knowing their tasks. In its simplest form, the approach is to reorder the menu items to put the most commonly used tasks earlier and higher in the hierarchy. Where users' task frequencies are not known or vary widely between users, it appears reasonable to allow the system to reorder itself upon a user's request after a sufficient break-in period. Of course, the semantics of the task and the semantics of the task titles will have a role to play as well, which we did not explore here. Others are working with ACT-R to create models that start to take account of this aspect of interaction [Pirolli 2003].

These models and the optimization algorithm bring together several interesting aspects of human behavior and show how a simple AI algorithm can help in HCI design. It also gives rise to both theoretical and practical implications.

Theoretically, novice user actions, learning, error recovery behavior, performance under stress, and generality across different devices are now areas ripe for further exploration. Having the models in hand also let us explore and explain new regularities in user behavior, such as the variations in key press time shown in Figure 8.

From a practical standpoint, developers have models that are ready for use—these models are general enough that they do not require cognitive modeling expertise or programming skill to apply them to different traversal tasks, in different menu hierarchies, or on different cell phones. Our longer-term goals for this research include the application of modeling techniques to provide insights into usability issues [Nichols and Ritter 1995] and the development of better cognitive modeling tools for evaluating and designing more general classes of user interfaces.

We believe that as modeling concepts and techniques become more accessible to HCI developers, they will become increasingly significant in their contribution to improving user interfaces. Wide application of the menu design approach in this article could, for example, save significant amounts of time. If 2 billion users were to use their cell phone menus every day for just three seconds, our improvements could save almost 30 years of user time per day.

REFERENCES

ANDERSON, J. R., BOTHELL, D., BYRNE, M. D., DOUGLASS, S., LEBIERE, C., AND QUIN, Y. 2004. An integrated theory of the mind. *Psychological Review 111,* 4, 1036–1060.

BYRNE, M. D. 2001. ACT-R/PM and menu selection: Applying a cognitive architecture to HCI. *International Journal of Human-Computer Studies 55,* 1, 41–84.

DIPRIMA, D. 2006. Cellular market rising above 2.14 billion subscribers. telecombeat.com (http://www.telecombeat.com/content/view/4689).

FITTS, P. M. 1954. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology 47*, 381–391.

GRAY, W. D., JOHN, B. E., AND ATWOOD, M. E. 1993. Project Ernestine: Validating a GOMS analysis for predicting and explaining real-world task performance. *Human-Computer Interaction 8,* 3, 237–309.

HAMMING, R. W. 1962. *Numerical methods for scientists and engineers*. Dover, New York, NY.

KIERAS, D. E. 2002. Model-Based Evaluation. In *The Human-Computer Interaction Handbook*, J. A. Jacko and A. Sears, Eds. LEA, Mahwah, NJ.

JOHN, B. E. 2003. Information processing and skilled behavior. In *HCI models, theories, and frameworks*, J. M. Carroll, Ed. Morgan Kaufman, San Francisco, CA.

JOHN, B. E. AND KIERAS, D. E. 1996a. Using GOMS for user interface design and evaluation: Which technique? *ACM Transactions on Computer-Human Interaction 3,* 4, 287–319.

JOHN, B. E. AND KIERAS, D. E. 1996b. The GOMS family of user interface analysis techniques: comparison and contrast. *ACM Transactions on Computer-Human Interaction 3,* 4, 320–351.

JOHN, B. E., PREVAS, K., SALVUCCI, D. D., AND KOEDINGER, K. 2004. Predictive human performance modeling made easy. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '04)*. ACM, Vienna, Austria. 455–462.

KIERAS, D. 1999a. *A Guide to GOMS Model Usability Evaluation using GOMSL and GLEAN3*. University of Michigan, Ann Arbor, MI.

KIERAS, D. AND MEYER, D. E. 1997. An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction 12,* 4, 391–438.

NAPIER, A., BATSELL, R., LANE, D. M., AND GUADAGNO, N. 1992. Knowledge of command usage in a spreadsheet program: Impact on user interface design and training. *Data BASE 23*, 13–21.

MACKENZIE, I. S. 2003. Motor behavior models for human-computer interaction. In *HCI models, theories, and frameworks*, J. M. Carroll, Ed. Morgan Kaufman, San Francisco, CA.

NEWELL, A. 1990. *Unified Theories of Cognition*. Harvard University Press, Cambridge, MA.

NICHOLS, S. AND RITTER, F. E. 1995. A theoretically motivated tool for automatically generating command aliases. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '95)*. ACM, NY, NY, 393–400.

PIROLLI, P. 1997. Computational models of information scent-following in a very large browsable text collection. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '97)*. ACM, NY, NY, 3–10.

PIROLLI, P. 2003. Exploring and finding information. In *HCI models, theories, and frameworks*, J. M. Carroll, Ed. Morgan Kaufman, San Francisco, CA.

POGUE, D. 2005. The cellphone that does everything imaginable, at least sort of. In *The New York Times*, May 12, 2005.

RITTER, F. E. 1991. Towards fair comparisons of connectionist algorithms through automatically generated parameter sets. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*. 877-881.

RITTER, F. E., FREED, A. R., AND HASKETT, O. L. 2005. User information needs: The case of university department web sites. *ACM interactions 12,* 5, 19–27.

RITTER, F. E., VAN ROOY, D., AND ST. AMANT, R. 2002. A user modeling design tool for comparing interfaces. In *Proceedings of the Fourth International Conference on Computer-Aided Design of User Interfaces (CADUI)*. 111–118.

RITTER, F. E. AND YOUNG, R. M. 2001. Embodied models as simulated users: Introduction to this special issue on using cognitive models to improve interface design. *International Journal of Human-Computer Studies 55,* 1, 1–14.

ST. AMANT, R., FREED, A., AND RITTER, F. E. 2004. Specifying ACT-R models of user interaction with a GOMS language. *Cognitive Systems Research 6,* 1, 71–88.

ST. AMANT, R., HORTON, T. E., AND RITTER, F. E. 2004. Model-based evaluation of cell phone menu interaction. In *Proceedings of the ACM Conference on Human Factors in Computing (CHI '04)*. ACM, New York, NY, 343–350.

ST. AMANT, R. AND RIEDL, M. O. 2001. A perception/action substrate for cognitive modeling in HCI. *International Journal of Human-Computer Studies 55,* 1, 15–39.

ST. AMANT, R. AND RITTER, F. E. 2004. Automated GOMS–to–ACT-R model generation. In *Proceedings of the Sixth International Conference on Cognitive Modeling (ICCM)*. LEA, Mahwah, NJ. 28–34.

SALVUCCI, D. 2001. Predicting the effects of in-car interface use on driver performance: An integrated model approach. *International Journal of Human-Computer Studies 55,* 1, 85–107.

SALVUCCI, D. D. AND LEE, F. J. 2003. Simple cognitive modeling in a complex cognitive architecture. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '03)*. ACM, Fort Lauderdale, FL, 265–272.

SILFVERBERG, M., MACKENZIE, I. S., AND KORHONEN, P. 2000. Predicting text entry speed on mobile phones. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '00)*. ACM, NY, NY, 9–16.

YOUNG, R. M. 1998. Rational analysis of exploratory choice. In *Rational models of cognition*, M. Oaksford & N. Chater, Eds. Oxford University Press, Oxford, UK, 469–500.