Predicting Task Execution Time on Handheld Devices Using the Keystroke-Level Model

Lu Luo
School of Computer Science
Carnegie Mellon University, Pittsburgh, PA 15213
luluo@cs.cmu.edu

ABSTRACT
The Keystroke-Level Model (KLM) has been studied in many areas of user interaction with computers because of its validity and predictive value, but it has not been applied to estimate user performance time in mobile environment such as handheld platforms. This paper investigates and verifies the applicability of KLM on handheld tasks. The KLMs and prediction time are created using a suite of cognitive tools called CogTool. A user study on 10 participants has shown that KLM can accurately predict task execution time on handheld devices with less than 8% prediction error.

KEYWORDS
Keystroke-Level Model, cognitive model, user performance time, handheld devices, mobile environment.

INTRODUCTION
The growing popularity and unique attributes of handheld devices, such as the Palm™ organizer and personal digital assistant (PDA), have brought new challenges to user interface design. To achieve higher portability, handheld devices are often equipped with a small touch screen display in combination with a stylus pen and several hardware buttons as the fundamental input/output devices. These devices have different attributes compared to the traditional monitor, mouse, and keyboard in the desktop settings. While a large body of research on desktop user interfaces exists, there is the need for exploring the efficiency of these non-traditional interface modalities on handheld devices.

The Keystroke-Level Model (KLM), proposed by Card, Moran, and Newell in 1980[2], has been used and studied as a means to produce *a priori* quantitative predictions of human performance at an early stage in the development process. In particular, KLM predicts error-free task execution time for expert users on a given interface design. The basic idea of KLM is to list the sequence of keystroke-level actions, or operators, that the user must perform to accomplish a task, and then add up the time required by each action. KLM asserts that the execution part of a task can be described in terms of four different physical-motor operators: K (key-stroking), P (pointing), H (homing), and D (drawing), and one mental operator, M, by the user, plus a response operator, R(t), by the system [2]. In KLM, most operators are assumed to take a constant time for each occurrence except for R. KLM does not embody a theory of system response time. The response times must be input to the model by giving specific values for the parameter *t*. The R times are only counted when they require the user to wait for the system as well as when there is no M operation following them. KLM provides a set of heuristic rules for placing M's in the method encodings.

Though KLM has been applied to many areas such as text editing, spreadsheets, learning, telephone operator call handling, and highly interactive tasks [2, 3, 6, 7, 8] since its introduction, little has been done on handheld tasks. This paper investigates the applicability of KLM to tasks on handheld devices. The following sections first describe the cognitive tools used to construct the models. Then a set of tasks on the PalmVx handheld organizer is chosen and keystroke-level models are built for those tasks. In order to verify the KLM predicted performance time, actual data of task time is collected from 10 expert PDA users. The experiments show that KLM can be successfully used on handheld tasks with less than 8% of prediction error.

CREATING KLM USING COGTOOL
Building predictive human performance models has been made easy by the recent work in [4]. In view of the high cost it requires to learn and construct correct cognitive models such as KLM, especially for those modelers who are not cognitive scientists, John and colleagues have built a suite of tools called CogTool to facilitate quickly producing accurate KLMs. CogTool allows the modeler to mock up an interface as an HTML storyboard and demonstrate a task on the storyboard, and automatically produces a consistent, correct KLM of that task that runs in the ACT-R cognitive architecture [1] to produce predictions of skilled performance time. This section will briefly discuss the

composition and usage of this tool suite, more details can be found in [4].

CogTool is a collection of several tools that act in concert to quickly produce accurate KLMs. The essential components of CogTool and their corresponding roles are:

Macromedia Dreamweaver is used to create HTML mock-ups in a WYSIWYG manner. In CogTool, several extensions are added to Dreamweaver so that it can be customized to serve the need of modeling.

ACT-R is a computational cognitive architecture for simulating human behavior and performance [1]. The current publicly available version, ACT-R 5.0, can be found at http://act-r.psy.cmu.edu). ACT-R allows cognitive models to interact with external simulation environment. The **ACT-Simple** compiler [9], in combination with ACT-R, provides a framework of straightforward commands that compile into ACT-R production rules.

Netscape web browser is used to demonstrate the HTML mock-ups. The web pages use HTML event handlers to send messages to the Behavior Recorder via the LiveConnect feature of Netscape.

Behavior Recorder is a software application developed by the authors of [5]. The Behavior Recorder can observe the demonstration of the mock-up in Netscape, generate the corresponding ACT-Simple commands, and automatically produce the resulting ACT-R code. The KLMs are created when demonstrating the mock-up, and the corresponding performance time is predicted in the ACT-R environment when running the models.

In this paper, all the KLMs and execution time prediction of tasks are constructed using CogTool.

TASKS ON HANDHELD DEVICES

To achieve the goal of investigating the applicability and prediction ability of KLM on tasks on handheld devices, the handheld platform should be popular enough to be representative. According to the consumer reports from Palm Computing, roughly 85 percent of handheld PDAs sold in 2001 use the Palm operating system (Palm OS). Therefore a Palm Vx PDA (Palm OS 3.3, 8MB RAM) is chosen as the handheld platform understudy. In addition, tasks should be carefully chosen so that the major user interaction modalities on Palm are included. To input data to a Palm platform, two methods are often used: tapping on an on-screen keyboard (referred as a "virtual" or "soft" keyboard), and writing in shorthand, known as Graffiti. To give operations, methods like tapping on icons, menus, lists and so forth on the touch screen and hardware buttons are often used. The tasks chosen should not only include most of the modalities, but also support multiple methods to achieve the same goal for comparison.

Based on the principles, an off-the-shelf software application named ChoiceWay Guides (CWG) for New York City is chosen. More information about CWG software can be found at http://www.choiceway.com/. This commercially available software provides complete guides of several large cities for both Palm OS and Windows CE. Basically, CWG is loaded with information about planning, city & country facts, and detailed information of hundreds of locations in the particular city. Figure 1 (a) shows the main interface of CWG New York City (CWG-NYC). The user can tap on different icons to perform corresponding query operation.

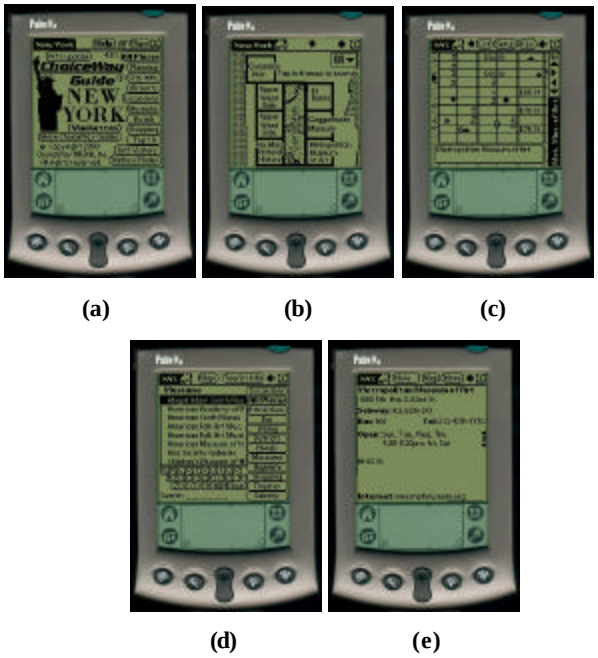

Figure 1. Snapshots of the CWG New York City application. (a) start page (b) region map (c) street map (d) museum list (e) query result

The goal of all tasks in this study is "*to find the information of Metropolitan Museum of Art* (MET)". There are four different methods to achieve this goal:

M1: Map Navigation. From the start page of CWG-NYC in Figure 1(a), tapping on the "map" icon at the top of the screen will lead to Figure 1(b), the interactive region map. Tapping on the corresponding area of MET in the region map, the street map in Figure 1(c) will be displayed. Further taps lead to zoomed-in street maps. Tapping on the hot spot of MET in the street map displays the query result – the text information of MET in Figure 1 (e).

M2: Soft Keyboard. From the same start page, tapping on the "museum" icon on the right side of the screen leads to Figure 1(d), a list of all museum names in alphabetic order. The item of MET is not shown when the list is open. In order to show the item of MET, M2 is to input text by tapping on the soft keyboard, displayed at the bottom of the screen in Figure 1(d), and enter the letters "M" followed by "E" to list all the museums starting with "ME". Now that the item of MET is shown, the corresponding text information in Figure 1(e) can be retrieved by tapping on

the item. M3 and M4 will also operate on the museum list, with different operations respectively.

M3: Graffiti. The only difference between M3 and M2 is that M3 uses Graffiti shorthand to input text. Again, when the item of MET is shown in the list, the corresponding text information of Figure 1(e) can be retrieved.

M4: Scroll Bar. Unlike M2 and M3, M4 does not need to input any text to update the list. Instead, M4 is to simply tap on the scroll bar to the right of the list so that the list scrolls down one page per tapping. When the item of MET is shown it can be selected for text information.

In this study, the four different methods, M1 through M4, are used interchangeably with the term "task" referred in Card, Moran, and Newell's original KLM paper. In the next sections, "task" and "method" will be used without distinction. As is stated in the CogTool section, KLMs are constructed for each task using CogTool, and the corresponding performance time prediction is obtained by running the model in ACT-R environment.

EXPERIMENTAL RESULTS

To determine how well KLM actually predicts performance time for handheld tasks, an experiment is run on 10 actual expert PDA users. All the participants are recruited by posting advertisement on campus electronic bulletin boards, and they all own one of the several kinds of PDAs listed in Table 1. It is intentionally chosen that the participating users own different PDAs including the Palm series, pocket PC, and smart cell phones for the sake of comparison and analysis among user data. Table 1 shows the detailed information of the participants including their gender, the model of, and for how long they haven owned their PDAs. The user numbers are assigned by the order each user participate in the study.

User (gender)	Device owned	How long
1 (M)	Palm Vx	5+ years
2 (M)	Compaq iPAQ	3 years
3 (M)	Palm IIIe	4 years
4 (M)	Handspring Visor	3 years
5 (F)	Handspring visor Pro	2 years
6 (F)	Dell PDA	1 year
7 (M)	iPAQ 3630	4 years
8 (M)	Kyocera 7135	4+ years
9 (M)	Handspring Visor Prism	3 years
10 (F)	Palm VA	3 years

Table 1. Information of participants.

Event Logger

The actual user task execution time is measured using EventLogger, a Palm OS system extension which records system events to a log file. The log files are Palm database (PDB) files that are formatted as text files for extracting and analyzing user behaviors. After the top part of palm data base header, each line of a log file is a tab-delimited listing of one system event, in the form "*TickCount sysEventName OptionalInfo*", where the *TickCount* is the timestamp of the event, the *sysEventName* is the name of the event, and the *OptionalInfo* includes information such as the character entered in a keystroke event, the name of the form in a form open event, etc. From the log files, user execution time for each task can be obtained by subtracting the ending event timestamp from the starting event timestamp and dividing the result by the number of system ticks per second defined in the PDB header.

Each participant is asked to perform all the tasks for two rounds. In the first round, each participant is given a detailed instruction on how to operate the EventLogger, and a step-by-step guidance on how to perform the four tasks. The participant is asked to precisely follow each step in the written instruction, and to repeat each task for 10 times. In this session, the participants are told to focus on getting familiar with the tasks. The event log files collected in this session are saved as training data. In the second round, the participants are asked to run each task for 10 times again. But this time, the users are asked to not refer to the instruction, because they are supposed to become familiar with the given tasks during the training session. The event log files collected from this session are used for verifying the model data. Among the total 4x10x10=400 task executions by the users, 20 are erroneous and those data is thrown away. Table 2 lists the result of the user study: the average, maximum, and minimum user time in seconds to perform each task, as well as the difference between the fastest and the slowest users.

Task	User time (sec)			Variation
	Average	Max	Min	
M1	9.00	11.92	7.31	38.7%
M2	12.84	15.65	10.14	35.2%
M3	13.60	16.24	11.06	31.9%
M4	10.30	12.26	8.90	27.4%

Table 2. Task execution time from user test.

The predicted time automatically generated from the KLMs for each task are M1=9.213 sec, M2=8.054 sec, M3=7.935 sec, and M4=8.347 sec. Large variation of prediction error is found when comparing the model time with the user time. The smallest prediction error, 2.25%, comes from M1, while the biggest prediction error is 71.43% for M3. Even the prediction error for M2 (59.38%) and M4 (23.37%) is much higher than average 21% prediction error stated in [2]. Since this result is far beyond what is expected, checking the correctness of the automatically generated KLMs is necessary.

In order to figure out the reason for the high prediction error, the keystroke level operators in the generated KLMs are checked carefully and compared with the user operations extracted from the system events in the log files. While there is barely any difference between the model and user operations for task M1, which is reflected in its low prediction error, some operators are found missing in M2, M3 and M4. This observation makes good sense because all three tasks involve opening the list of museums for further query or selection. It takes the palm system a certain amount of time to respond when the CWG application wants to display or update the list. Therefore an **R(*t*)** operator should be placed in the models where the list is open or updated (the parameter ***t*** for opening the list is slightly longer than it for updating the list). Another observation is that there should also be system response time for Graffiti input recognition in task M3. This leads to the addition of an **R(*t*)** after each Graffiti stroke, the ***t*** here is set to be 500ms, based on previous studies on Graffiti text input.

The analysis above leads to small modification of the CogTool: two instrumented widgets (system response time and graffiti recognition) are added to the DreamWeaver extension. The HTML mock-ups for M2 through M4 are therefore to be created again, adding the corresponding widgets where needed. The models are then reconstructed and the calibrated model time in comparison with the average user time is shown in Figure 2.

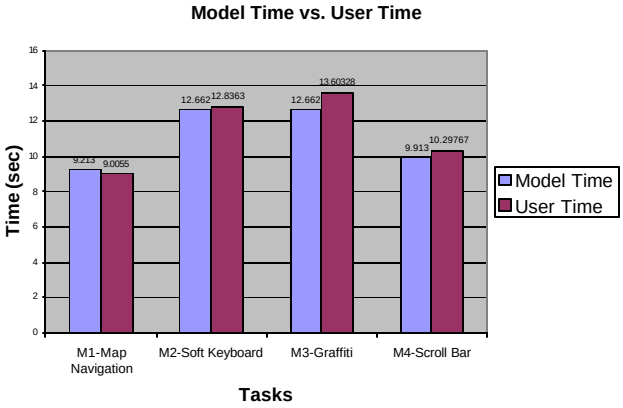

Figure 2. Model predicted time and actual user time.

Result Analysis

The result shows that the calibrated model successfully predicts the execution time for all four tasks with prediction error of 2.25%, 1.38%, 7.43%, and 3.88% for M1 through M4, correspondingly. This result complies with the declaration in [4] that CogTool generates more accurate results than previously published models. More importantly, the result verifies the assumption that KLM can be successfully used to model user behaviors and predicts task execution time in the environment of handheld devices.

It is worth mentioning that the slightly higher prediction error for M3 may be attributed to user difference. The longest execution time for M3 actually comes from users 2 and 7, and both of them are pocket PC (iPAQ) users. Since there is no Graffiti method in pocket PCs, it should take longer thinking time for those users to input the required Graffiti strokes. By removing their data from the average user time, we get a better prediction error of 3.8% for M3.

CONCLUSION AND FUTURE WORK

It is shown in this paper that KLM is effective for predicting task execution time on handheld devices. Aided by CogTool, it turns out to be straightforward to construct cognitive models even for models with little or no background in cognitive science. Future work will study a larger scope of tasks and users in various mobile platforms.

REFERENCES

1. Anderson, J.R. and Lebiere, C. *The Atomic Component of Thought*. Lawrence Erlbaum Associates, Hillsdale, NJ, USA, 1998.
2. Card, S.K., Moran, T.P., and Newell, A. The Keystroke-Level Model for User Performance Time with Interactive Systems. *Communications of the ACM archive*, 23(7), 1980, 396-410.
3. John, B.E. Extensions of GOMS Analysis to Expert Performance Requiring Perception of Dynamic Visual and Auditory Information. *Proceedings of ACM CHI'90 Conference on Human Factors in Computing Systems*, 1990, 107-115.
4. John, B.E., Prevas, K., Salvucci, D.D., and Koedinger, K. Predictive Human Performance Modeling Made Easy. *Proceedings of ACM CHI'04 Conference on Human Factors in Computing Systems*, 2004.
5. Koedinger, K.R., Aleven, V., and Heffernan, N. Toward a Rapid Development Environment for Cognitive Tutors. *Proceedings of AI-ED 2003*, 455-457
6. Olson, J.R. and Nilsen, E. Analysis of the Cognition Involved in Spreadsheet Software Interaction. *Human-Computer Interaction*, 3(4), 1988, 309-350.
7. Olsen, J.R. and Olson, G.M. The Growth of Cognitive Modeling in Human-Computer Interaction since GOMS. *Human-Computer Interaction*, 5(2&3), 1990, 221-265.
8. Singley, M.K. and Anderson, J.R. A Keystroke Analysis of Learning and Transfer in Text Editing. *Human-Computer Interaction*, 3(3), 1988, 223-274.
9. Salvucci, D.D. and Lee, F.J. Simple Cognitive Modeling in a Complex Cognitive Architecture. *CHI 2003, ACM Conference on Human Factors in Computing Systems*, CHI Letters 5(1) 265-272.