KLEM: A Method for Predicting User Interaction Time and System Energy Consumption during Application Design

Lu Luo and Daniel P. Siewiorek†
School of Computer Science and †Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213 USA
{luluo, dps}@cs.cmu.edu

Abstract

The impact of user interactions on the electric energy consumption of a portable computer system and on user efficiency is often not obtainable until after the software application is implemented and deployed on a specific hardware platform. In this paper, we present the Keystroke-Level Energy Model (KLEM), a method that can predict the user time and system energy consumption it will take to perform an interactive task at run time during the phase of application design. KLEM is based on the Keystroke-Level Model (KLM), a psychological theory of human cognitive and motor capabilities that can predict execution time for a skilled user. We first create a design storyboard and define a set of tasks whose KLMs are to be constructed. We then construct KLEM of each task by correlating system activities to the user actions modeled in the corresponding KLM. We obtain the energy profiles of system activities from running a set of user interaction benchmarks on the target hardware platform. To verify KLEM, we conducted a user study of 10 participants on executing an information query task using eight different methods. The user time and system energy of the participants were measured on two popular handheld platforms: a Windows Mobile iPaq and a Palm OS Tungsten. Our experimental results show that KLEM has an average prediction error of 5.6% and 8.8% on user time, and 4.4% and 8.4% on energy consumption on the two platforms, respectively.

1. Introduction

Human users today have a wide variety of methods to interact with handheld and wearable systems. These systems, enabling the user to access information ubiquitously, are characterized by their small form factors with novel, heterogeneous user interaction modalities. While the current state-of-the-art in interface devices for portable systems include pen-based touch screen and hardware buttons, wearable computers add head-mounted displays, touch pads, trackballs, and Gyroscopic or Twiddler mice. Speech recognition, position sensing, and eye tracking are also becoming common input modalities. In the future, stereographic audio and visual output will be coupled with 3D virtual reality information [17]. This heterogeneity of interaction modalities makes it difficult to predict and make decisions on important run time performance attributes at earlier design stages before the application is actually implemented, deployed, and user-tested.

Energy consumption has been one of the most important design considerations for handheld and wearable systems. Most tasks performed on handheld and wearable systems are highly interactive. The unavailability of software application during design time and the large amount of time and effort needed to conduct user studies are substantial barriers to achieving accurate prediction of how the user would actually interact with the computer and of the resulting user time and system energy consumption.

In this paper, we present the Keystroke-Level Energy Model (KLEM), a quantitative analysis method to address the problems stated above. Our approach is based on the Keystroke-Level Model (KLM), a cognitive modeling technique that predicts the time it takes a user to perform a task with a given method on an interactive computer system [4]. In KLM, task execution time is estimated by listing the sequence of elementary perceptual, motor or cognitive actions, i.e., operators, and then summing the times of the individual operators. KLM aggregates all perceptual and cognitive function into a single value for an entire task, using a heuristic. KLM has been used for almost

three decades by Human-Computer Interaction (HCI) practitioners as a reliable and valid means of modeling human performance. In a previous work [15], we have shown that KLM can be used to produce accurate task execution time prediction on handheld systems. We now extend KLM into KLEM, which integrates the contextual system energy consumption information with the user interaction model of KLM, and predicts both system energy consumption and user interaction time.

The rest of the paper is organized as follows. In Section 2, we discuss the related work on user interface and software energy consumption. In Section 3, we introduce background information on KLM and explain why it is a suitable approach to energy estimation at design time. In Section 4, we explain the process of constructing KLEM. In Section 5, we describe the experimental setup for the task time and energy measurements. In Section 6, we present and discuss the results of a user study, and verify the KLEM predictions against the user study results. Finally we conclude the paper and propose future work in Section 7.

This paper makes two major contributions: *first,* a methodology based on well-established HCI theory and practices is introduced to make design-time prediction on interactive task energy consumption; *second,* the energy efficiency of different user interaction methods on the same task is compared and analyzed. We show when using different interaction modalities, the energy consumption can vary by a factor of three in achieving the same user goal.

2. Related Work

Research that recognizes the significance of human interaction and user interface design on user performance and system energy consumption is relatively new. Most existing approaches focus on run-time energy management techniques rather than design-time energy prediction. The user interface events for dynamic voltage scaling were discussed in [14]. Techniques were proposed to aggressively power-manage the system during user idle periods based on user-delay predictions [21]. Low power techniques for interface devices such as displays were reported in [7], [8], [11], and [16].

An emulation-based energy model for the Palm OS was given in [3] to characterize system energy consumption, a. The limitation of this approach is that an emulator running on a PC can only accept desktop mouse events, which do not reflect the real pen and key user interface behaviors on the Palm device, especially when applications are used interactively. There has been a large body of research focusing on characterizing and modeling energy consumption of individual hardware components, embedded operating system and software [2, 9, 18, 19]. Those approaches either failed to provide higher level views of system energy behavior, or lacked the generality to be applied on consumer platforms and applications.

The only existing work that is closely related is the characterization of energy consumption of different graphical user interface (GUI) features on handheld computers presented by Zhong et al [20, 21, 22]. They pointed out the importance of idle time and user productivity in system energy efficiency, and suggested several energy efficient GUI design techniques in [13]. Another work of theirs characterized the energy requirements and overheads imposed by various human sensory and speed limits [22], but did not provide a comprehensive way to understand how these human aspects could be integrated during the actual user interactions.

3. Why Keystroke-Level Model

The form factors of handheld/wearable computers have generated diverse and novel user interaction modalities. The most common are pen-based touch screen interface inherited from the desktop graphical user interface (GUI), handwriting recognition such as Graffiti™, and hardware navigation buttons. In addition, speech recognition and synthesis are gaining popularity. These user interaction modalities have brought variety to the design of handheld/wearable applications, and we are interested in predicting their impact on the human performance and system energy consumption of interactive user tasks. The goal is to create quantitative, accurate prediction during the application design phase. However, how a human user actually perform the task, i.e., the steps the user takes and the cognitive delays between consecutive steps often remain unknown without the expense of observing and recording the target application on real hardware executed by real users.

Cognitive engineering models view the human mind as another information-processing system – the Model Human Processor [5] that interacts with the computer system. The KLM is one of the cognitive modeling techniques that predicts the time it takes an *expert user* to perform an *error-free task* using a *given method* on an interactive computer system [4]. In KLM, task execution time is estimated by listing the sequence of elementary perceptual, motor or cognitive

actions, i.e., operators, and then summing the times of the individual operators.

The original KLM had four physical-motor operators: *K* represents pressing a key or a button, *P* represents pointing with the mouse to a target on the display, *H* represents moving hands to the home position on the keyboard or mouse, and *D* represents drawing lines using the mouse; one mental operator: *M* is a heuristic to incorporate mentally preparing for a task; and one system response operator: *R* for system response where the user waits for the system. For each operator, there is an estimation of constant execution time set by previous psychology and HCI studies. An exception is that the R operator must be estimated by the designer, and only include the time that the user must wait for the system after any M operator has completed. Additionally, there is a set of heuristic rules to account for mental preparation time.

KLM has been used and validated repeatedly by academics and practitioners since it was introduced, been applied to a significant amount of interactive tasks in areas such as word processing, spreadsheets, graphic, and video games [4, 10], and shown to provide good predictive accuracy. Our previous work [15] showed that KLM can provide accurate human performance predictions for pen-based user interfaces on handheld devices.

4. Keystroke-Level Energy Modeling

In this section, we first present the overall process of predicting the user time and system energy consumption of a given task using KLEM. We then describe how KLM operators are integrated to energy-consuming system activities based on the contextual user interaction information of the task. The energy profiles of system activities are obtained by running a set of user interaction benchmarks on the target platform.

4.1. Overview of KLEM

Given: A task, the methods used to execute the task, the proposed interface design, and a target platform. Here we define a *task* as a series of interactive operations a user performs on a computer system to achieve a certain goal, such as schedule a meeting, inquire about store hours, or changing the system settings.

Predict: The user time and system energy an expert user will take to execute the task using the system, providing the user uses the method without error.

A flowchart of the modeling process is shown in Figure 1. The resulting user-energy performance prediction consists of task execution time (User Time Prediction) and task energy (Energy Prediction). The task execution time is obtained by running the *Modeling Process* to produce the energy prediction, which produces the KLM of the task. The *Energy Characterizing Process* obtains the necessary Energy Profiles by running the UI Benchmarks on a Target Platform. The KLM contains the information (i.e. ACT-R Trace and Visualization, to be discussed in Section 4.2) that is needed in the *Mapping Process*.

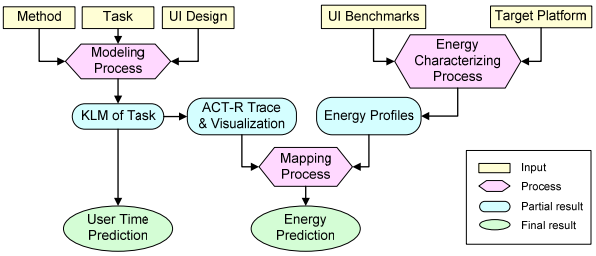


Figure 1. The flowchart of constructing a KLEM

4.2. Modeling Process

In the *Modeling Process*, the KLM of a given task is constructed. Although the KLM enjoyed a rich history in HCI because of its validity and predictive value, the cost of learning and the complexity of constructing correct models remained too high to justify the benefits of predicting task execution time. Fortunately, designers without much expertise in KLM can use CogTool [6,12] to generate predictive cognitive models of skilled user performance simply by demonstrating their task on storyboards of the new design. The predictions made by CogTool are based on KLM, and are generated with a computational cognitive engine called ACT-R [1]. For mobile devices, CogTool currently provides functionality to simulate interaction using a fingertip or stylus, therefore can predict performance of platforms with hardware buttons or pen-based input. The latest version of CogTool will support Hear and Say behaviors, which facilitates our future work of modeling speech based interactions.

To make a prediction, the designer first needs to create a S*toryboard* for the interface design upon which tasks will be performed. A storyboard contains a series of *Frames* that represent the changes of the display between user operations. A frame can be a sketch of the proposed user interface design. Once the design storyboard has been created, the set of *Tasks*

needs to be defined. Then, the designer needs to demonstrate on the storyboard the steps a user would perform to accomplish a task. The demonstration is recorded to a script for CogTool to generate predictions.

A human user manipulates an application using GUI widgets. In CogTool, a *Widget* is represented as a "hot-spot" in a frame to indicate an interactive area on the actual physical device. Interactive widgets that are currently supported by CogTool include Button, Check box, Radio button, Textbox, Pull-down list, List box, Menu (header, submenu, and menu item), and handwriting input area (e.g. Graffiti™ in Palm OS and Soft Input Panel (SIP) in Windows Mobile). The KLM P operation is relevant to layout of Widgets defined in the storyboard, based on Fitts' Law [5].

As an example, a piece of the model trace created in CogTool is shown in Figure 2, showing the activities ACT-R goes through to make the time prediction. The ACT-R trace contains the contextual information necessary for the construction of KLEM. For instance, the model user executes a MOTOR activity at 0.683 second, which taps the device display at the coordinate (278.0 177.0).

```
...
 0.400  MOTOR            INITIATION-COMPLETE
 0.400  PROCEDURAL       CONFLICT-RESOLUTION
 0.683  MOTOR            MOVE-CURSOR-ABSOLUTE #(278.0 177.0)
 0.683  Storyboard transitioning to frame "List1"
 0.683  PROCEDURAL       CONFLICT-RESOLUTION
 0.733  MOTOR            FINISH-MOVEMENT
 0.733  PROCEDURAL       CONFLICT-RESOLUTION
 0.768  VISION           Encoding-complete LOC1-0 NIL
 0.768  PROCEDURAL       PRODUCTION-SELECTED WAIT-FOR-SYSTEM-5
 0.768  PROCEDURAL       BUFFER-READ-ACTION GOAL
...
 0.768  PROCEDURAL                BUFFER-READ-ACTION GOAL
 0.768  PROCEDURAL       QUERY-BUFFER-ACTION MANUAL
 0.818  VISION           CHANGE-STATE LAST NONE PREP FREE
 1.324  COGTOOL          Restoring display at end of system wait (0.556)
...
```

Figure 2. An example ACT-R model trace created by CogTool

The execution of a MOTOR in the model (note that the model simulates a human user) is a trigger to a GUI event that the computer system needs to respond to (therefore consume energy). In the example trace above, at 0.683 second, the MOTOR causes the storyboard to transition to the next frame, which represents a display update in the computer system. Similarly, at line `0.768 PROCEDURAL PRODUCTION-SELECTED WAIT-FOR-SYSTEM-5`, the model user starts waiting for the system until the system restores at 1.324 sec. This duration of waiting is considered as a system response time operator in KLEM. The PROCEDUAL activities during the system response time represent the thoughts the model has when performing the task and the VISION activities represent the eyes seeing objects on the frame. The system response time can be caused by any sort of computation or communication job that the system must finish before the user can continue operation.

4.3. Energy Characterizing Process

We use a measurement based approach to obtain the energy profiles of KLEM operators by running a set of benchmarks on the target platform. This approach is suitable when the target platform is already available at the time of application design. If the target platform is not available for benchmarking, the energy profiles can be obtained from the manufacturer hardware datasheets or data from literature. In this paper, we focus on the measurement-based approach only.

In an ACT-R trace, some activities such as the MOTOR and WAIT-FOR-SYSTEM cause non-idle system activities thus extra energy consumption, while during other activities such as the VISION and PROCEDUAL the system is usually idle waiting for user operations. The energy consumption of non-idle system activities depend on the hardware platform, operating system, and application software. It also depends on the particular interaction method. For instance, a MOTOR activity can be tapping, dragging, key pressing or releasing, or a handwriting stroke. Besides, a WAIT-FOR-SYSTEM can be of any length, depending on the details of a design.

We create a set of benchmarks of interaction activities performed on common GUI widgets. Table 1 lists the benchmarks grouped by similar functions: *Selection*, *Navigation*, and *Text Input*. The operation(s) user can perform on each widget and the corresponding system activities are also listed.

Table 1. Interaction activity benchmarks

Widget	Operation	System Activity
Selection		
Button	Tap	Small, Medium, Large
Checkbox	Tap	Small
List box	Tap	Small
Dropdown list	Tap+ Tap	Small
Radio button	Tap	Small
Menu	Tap	Small, Medium, Large
Hardware button	Tap	Small, Medium, Large
Navigation		
Tab	Tap	Medium, Large
Scrollbar	Tap/Drag	Small, Medium, Large
Slider	Tap/Drag	Small
Text Input		
Soft keyboard	Tap	Small
Handwriting	Stroke	Medium, Large

In the *Selection* group, the widgets Checkbox, List box, Dropdown list, and Radio button are usually used to make a selection among several items that the user operates by tapping the widget. Note that the

Dropdown list widget requires two taps to perform a selection, and one can use tapping and dragging to operate the Scrollbar and Slider widgets. After the user selects an item, the application records the selection, updates a small area of the display to look responsive to user operation, and goes back to the waiting/idling state for the next user operation. We define this kind of system activity "Small". As for "Medium" and "Large" system activities, for example, if a "next" button is pressed to open a new window, the consequent system activity is defined "Medium", while an "open" button that reads a 1MB file is considered "Large" system activity. At this point we can only use rough estimation instead of accurate quantification.

We assume that during the task execution, the system runs in normal operation mode with no special power management techniques. For the tasks we studied in this paper, the hardware components involved are processor, memory, and display. We define a simple state machine of system activities with the corresponding power level of each state in **Figure 3**. The power levels P_i and P_b are obtained from the interaction activity benchmarks.

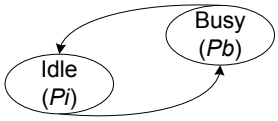


Figure 3. A simplified state machine of system activities. If other hardware components are involved during user operation, more states should be added.

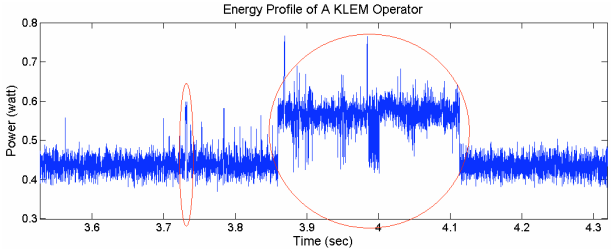


Figure 4. Energy profile of a "button press" followed by a display update

Figure 4 depicts the energy profile of a "button press" (Tap) operation followed by a full display update (e.g. open a new window, load a picture) measured on one of the handheld devices we studied. The values along the *y* axis are the instantaneous power level at time *t*, and the system energy consumption during any time interval (t_1, t_2) is the area under the power trace between t_1 and t_2. In Figure 4, the smaller oval circle indicates the energy profile of the *Tap* operation and the larger oval indicates the energy profile of the display update activity. The lower power level (e.g. 3.6 sec to 3.7 sec) is the P_i in the *Idle* state, and the higher power level (e.g. 3.9 sec to 4.1 sec) is the P_b in the *Busy* state.

We define the energy consumption of a KLEM operator the sum of the KLM operator energy and the system activity energy it invoked. Let S be the set of power states of the system activity following an operator, P_s be the power level of each state, T_s be the time the system stays in state c, then the system energy consumption triggered by this operator E_o can be predicted using the following formula:

$$E_o = \sum_{s \in S} P_s T_s \qquad (1)$$

To achieve accurate system energy prediction, quantitative characterizations of the system activities invoked by KLEM operators must be obtained. We already have the power profiles during system activities defined in **Figure 3**, and we still need the system activity time T_s. We create another set of benchmarks that measure the time it takes a target platform to perform common system activities such as opening a new window, writing to a file, or searching an item in database. These benchmarks are no different than other system performance benchmarks. Due to space limitation, we do not elaborate on this topic because there are enormous research and practices on performance analysis and benchmarking. We used two simple benchmarks to measure the display update time to construct the models in this paper.

4.4. Mapping Process

The *Mapping Process* takes the CogTool predicted total task time T_{task}, the storyboard and the ACT-R trace that contains the contextual system activity information, and the KLEM operator energy profiles defined in the *Energy Characterizing Process*, and produces the task energy prediction. Let O be the sequence of KLEM operators in the task, T_o be the time of operator o, the total system energy consumption during idle state E_{idle} is:

$$E_{idle} = P_i(T_{task} - \sum_{o \in O} T_o) \qquad (2)$$

The total task energy can be predicted using:

$$E_{task} = \sum_{o \in O} E_o + E_{idle} \qquad (3)$$

5. Experiment Setup

We chose two popular handheld platforms to validate the KLEM predictions on user time and

system energy. The specifications of devices are summarized in Table 2.

We removed the battery from both devices to eliminate the current draw of battery charging. The devices were connected directly to the external power supply and the input current I_i was obtained by measuring the voltage V_i across a 1 Ohm resistor R connected in series with the device. The voltage value is sampled at 10 KHz using a high speed Data Acquisition Card (DAQ) PCI-9820. For both devices, we ignored the minor fluctuation in the supply voltage V_s and assumed it to be constant. The instantaneous electric power P that a device consumes at a given time is:

$$P = V_s \times I_i = V_s \times V_i \div R = V_s \times V_i \quad (4)$$

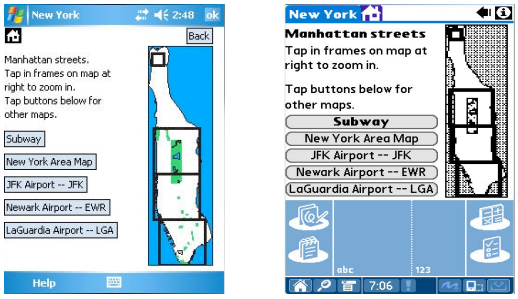
When choosing the tasks to validate the KLEM model, we kept in mind two principles. First, the operations required to accomplish the tasks should cover as many different interaction methods available in the target platforms as possible. Second, for comparison purposes, the same goal should be accomplishable by using different interaction methods. We selected a commercial off-the-shelf tour guide application named ChoiceWay™ Guides (CWG) for New York City because it has releases for both Windows Mobile and Palm OS.

Table 2. Specifications of target platforms

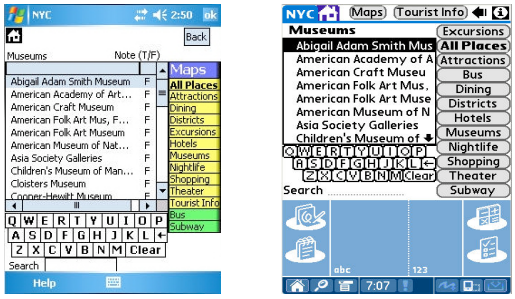
	iPaq	Tungsten
Vendor	HP	PalmOne
Model	Rx1955	T5
SoC	300MHz Samsung SC32442	416MHz Intel XScale
Storage	32 MB built-in RAM 64 MB Flash ROM	215MB storage capacity 160MB internal flash drive
Display	TFT color LCD 64K colors 240 x 320 (QVGA).	TFT color display 64K colors 320 x 480
OS	Windows Mobile 5.0	Palm OS v5.4

Figure 5 shows four screenshots of the CWG application interface on the two platforms we used. For simplicity, we built the storyboard for CogTool using the screenshots since we already have the software. In reality where KLEM is used during design time, modelers can use sketches of the proposed interface design to build the storyboard.

All tasks we modeled have the same goal of finding the opening hours of the Metropolitan Museum of Art (MET). There are two different ways in CWG to find this information and each can be considered as a different interface design. One way is to navigate the map of Manhattan area as depicted in Figure 5(a). The user can zoom in the map by tapping one of the several areas delimited by the four boxes. When the street map of the MET neighborhood is displayed, the user can view the open hour information by tapping the dot representing the location of MET on the street map. The other way is to display a list of all New York museums, and choose MET from this list to display the open hour information. There are various methods to select MET from the list: searching or browsing. For the iPaq, the user can tap the letter "M" on the software keyboard at the bottom of the museum list as shown in Figure 5(b); tap the trough of the list's scrollbar until MET can be viewed in the list; tap the down arrow; tap the down arrow and hold it until the MET item appears in the current list window; drag the scrollbar; and press the hardware navigation button at the bottom of the device to browse down the list.



(a) The map navigation interface of CWG for WM5 and PalmOS5



(b) The scroll list interface of CWG for WM5 and PalmOS5

Figure 5. Screenshots of CWG software

For the Tungsten, the user can input the letter "M" on the software keyboard; tap the down arrow: gesture "M" in the Graffiti area at the lower part of the device display, and press the hardware button to browse down the list.

The scrollbar in the Tungsten does not have the same design as the iPaq and cannot be manipulated using dragging or tapping the trough. Although one can invoke the soft input panel (SIP) at the bottom of the iPaq screen, the handwriting area will obstruct the lower part the list, which makes it unnatural and error-prone to use. Therefore handwriting recognition in iPaq that corresponds to the Palm Graffiti input is not used in these tasks.

6. User Study and Experimental Results

To verify the KLEM prediction of user time and system energy, we performed a user study on ten participants (six male, four female), all are engineering major undergrad or graduate students who are familiar and comfortable with using computers. Each participant was first asked to practice all the tasks under the author's instruction as a training session. The participants were given adequate time to practice the tasks until s/he became very familiar with the tasks without making errors or unnecessary pauses during the task execution. This conforms to the central ideas of KLM. The participants were then asked to perform all 12 tasks on the two devices as the testing session. The device power supply traces with corresponding time stamps of each task were measured during the testing session, as described in the previous session.

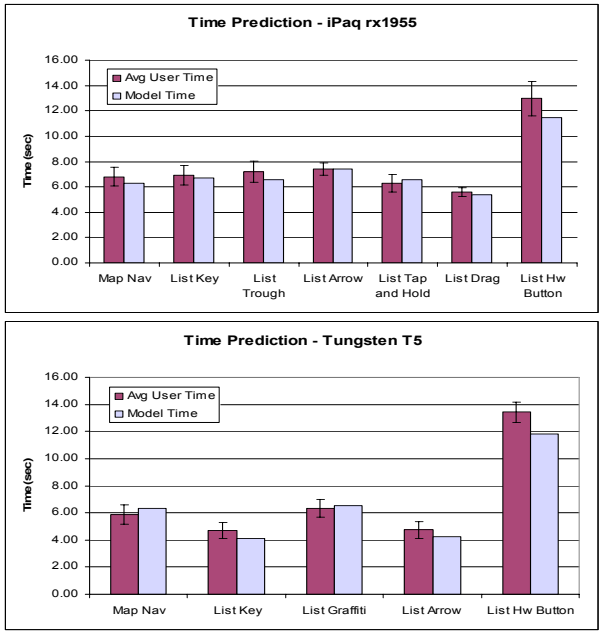


Figure 6. Comparison of measured user time and model predicted time. The bars represent observation means and the error bars on the user data indicate the 95% confidence intervals.

The average measured user times versus the predicted model times is shown in Figure 6. The time prediction errors against the average measured user time for the iPaq tasks are between 0.1% and 11.7% (average 5.6%). For the Tungsten, the error rates are 2.6% to 12.6% (average 8.8%) for KLEM time predictions. Note that the predictions for "List Hardware Button" tasks for both platforms have comparably higher error due to the fact that the number of hardware button presses used by different users to browse the list varies widely.

The average measured task energy versus the predicted model energy is shown in Figure 7. The energy prediction errors against the average measured task energy for the iPaq tasks are between 0.3% and 8.1% (average 4.4%). For the Tungsten, the error rates are between 1.2% and 12.5% (average 8.4%).

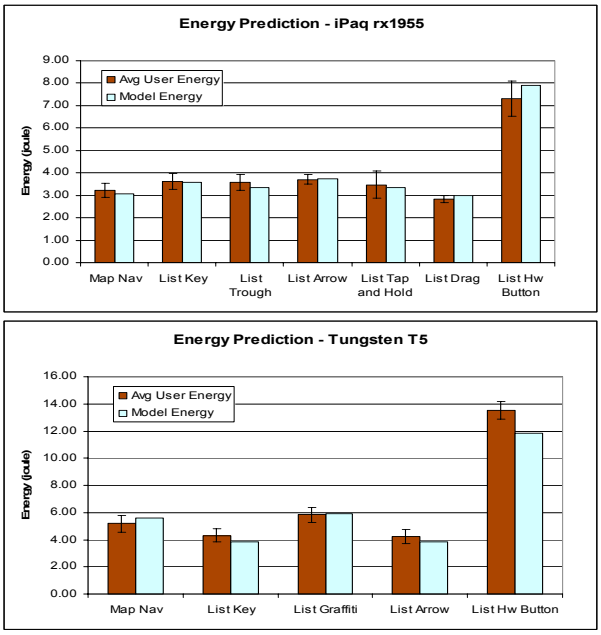


Figure 7. Comparison of measured task energy and model predicted energy. The bars represent observation means and the error bars on the user data indicate the 95% confidence intervals.

7. Conclusion and Future Work

Based on the KLM user performance prediction techniques, KLEM can predict user time and energy consumption from story boards of proposed user interactions with good accuracy. The resulting predictions are within 13% of measured user time and system energy for two different popular handheld platforms. The time and energy for doing the same task in different modalities on the same platform (List Hardware Button vs. List Drag or List Key) can vary by a factor of two to three. Up to a 30% variation in doing the same task with the same modality on different platforms was also observed.

Future work will include extending the model to other interaction modalities such as speech recognition for input and speech synthesis and speech playback for output. The performance benchmarks also need to be enriched for a larger scope of system activities.

8. Acknowledgements

This material is based upon work supported by the Defense Advanced Project Agency (DARPA) under Contract No. NBCHD030010, and the National Science Foundation under Grant Nos. EEEC-540865 and 0205266.

The authors would like to thank Bonnie John for providing insightful guides on cognitive models, and Don Morrison and Jason Cornwell from the CogTool team for their help on fixing bugs and answering questions.

9. References

[1] http://act-r.psy.cmu.edu/ The ACT-R website.

[2] K. Baynes, C. Collins, et. al., "The performance and energy consumption of three embedded real-time operating systems", In Proc. Int. Conf. Compilers, Architecture, and Synthesis for Embedded Systems, Nov. 2001, pp. 203-210.

[3] T. L. Cignetti, K. Komarov, and C. S. Ellis, "Energy estimation tools for the Palm", In Proc. ACM MSWiM 2000: Modeling, Analysis and Simulation of Wireless and Mobile Systems, Aug. 2000.

[4] S. K. Card, T. P. Moron, and A. Newell, "The Keystroke-level model for user performance time with interactive systems", In *Communications of the ACM archive*, 23(7), 1980, pp. 396-410.

[5] S. K. Card, T. P. Moron, and A. Newell, *The psychology of human computer interaction*, Lawrence Erlbaum Associates Inc., Mahwah, NJ, USA, 1983.

[6] http://www.cogtool.org/download.html CogTool website.

[7] W. C. Cheng, and M. Pedram, "Power minimization in a backlit TFT-LCD display by concurrent brightness and contrast scaling", In IEEE Trans. Consumer Electronics 50, 1, Feb. 2004, pp. 25–32.

[8] I. Choi, H. Shim, and N. Chang. "Low-power color TFT LCD display for handheld embedded systems", In Proc. Int. Symp. Low Power Electronics & Design, Aug. 2002, pp. 112-117.

[9] K. I. Farkas, J. Flinn, et. al., "Quantifying the energy consumption of a pocket computer and a Java virtual machine", In Proc. ACM SIGMETRICS: Int. Conf. Measurement & Modeling of Computer Systems, June 2000, pp. 252-263.

[10] P. Haunold, and W. Kuhn, "A keystroke level analysis of a graphics application: Manual map digitizing", In *Proc. of CHI*, April, 1994, pp. 337-343.

[11] S. Iyer, L. Luo, R. Mayo, and P. Ranganathan, "Energy-adaptive display system designs for future mobile environments", In Proc. Int. Conf. Mobile Systems, Applications, & Services, May 2003, pp. 245–258.

[12] B. John, K. Prevas, D. Salvucci, and K. Koedinger, "Predictive Human Performance Modeling Made Easy", In *Proceedings of CHI*, April 2004.

[13] K. S. Vallerio, Lin Zhong and N. K. Jha, "Energy-efficient graphical user interface design", In *IEEE Trans. on Mobile Computing*, July 2006.

[14] J. Lorch, and A. Smith, "Using user interface event information in dynamic voltage scaling algorithms", In Proc. Int. Symp. Modeling, Analysis & Simulation of Computer Telecommunications Systems, Oct. 2003, pp. 46–55.

[15] L. Luo and B. John, "Predicting Task Execution Time on Handheld Devices Using the Keystroke-Level Model", In *Conference on Human Factors in Computing Systems (CHI '05) extended abstracts on Human factors in computing systems*, April 2005, pp. 1605-1608.

[16] S. Pasricha, S. Mohapatra, et. al., "Reducing backlight power consumption for streaming video applications on mobile handheld devices", In Proc. First Workshop Embedded Systems for Real-Time Multimedia, Oct. 2003.

[17] D. Siewiorek, "New frontiers of application design," Communications of ACM, vol. 45, no. 12, Dec. 2002.

[18] V. Tiwari, S. Malik, and A. Wolfe, "Power Analysis of Embedded Software: A First Step Towards Software Power Minimization", In IEEE Trans. on Very Large Scale Integration (VLSI) Systems, 2(4):437-445, Dec. 1994.

[19] Y. Xiong, X. Zhou, X. Li, and Y. Gong, "OOEM: object-oriented energy model for embedded software reuse", IEEE Int. Conf. on Information Reuse and Integration, 2003.

[20] L. Zhong, and N. K. Jha, "Graphical user interface energy characterization for handheld computers", In Proc. Int. Conf. Compilers, Architecture, & Synthesis for Embedded Systems, Nov. 2003, pp. 232–242.

[21] L. Zhong, and N. K. Jha, "Dynamic power optimization for interactive systems", In Proc. Int. Conf. VLSI Design, Jan. 2004, pp. 1041–1047.

[22] L. Zhong, and N. K. Jha, "Energy efficiency of handheld computer interfaces: Limits, characterization, and practice", In Proc. USENIX/ACM MobiSys, June 2005