Modeling Task Performance for a Crowd of Users from Interaction Histories

Steven R. Gomez
Computer Science Department
Brown University
steveg@cs.brown.edu

David H. Laidlaw
Computer Science Department
Brown University
dhl@cs.brown.edu

ABSTRACT
We present TOME, a novel framework that helps developers quantitatively evaluate user interfaces and design iterations by using histories from crowds of end users. TOME collects user-interaction histories via an interface instrumentation library as end users complete tasks; these histories are compiled using the Keystroke-Level Model (KLM) into task completion-time predictions using CogTool. With many histories, TOME can model prevailing strategies for tasks without needing an HCI specialist to describe users' interaction steps. An unimplemented design change can be evaluated by perturbing a TOME task model in CogTool to reflect the change, giving a new performance prediction. We found that predictions for quick (5–60s) query tasks in an instrumented brain-map interface averaged within 10% of measured expert times. Finally, we modified a TOME model to predict closely the speed-up yielded by a proposed interaction before implementing it.

Author Keywords
Performance modeling; KLM; user interfaces; histories.

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation (e.g., HCI): User Interfaces

General Terms
Human factors; Performance; Measurement.

INTRODUCTION
Quantitative user studies can help interface developers evaluate new tool designs, but are often difficult to plan and carry out. Analyzing usage data in each design iteration is often prohibitively expensive. An alternate approach is to construct a predictive model of the tool's utility for a task (e.g., user speed or accuracy) and evaluate interface changes by running the model.

This paper describes a framework called TOME for building and extending such models with minimal human involvement. The framework prepares graphical storyboards of task executions that yield task completion-time predictions using the KLM [2] when loaded by the open-source application CogTool [7]. TOME is aimed at modeling quick (5–60s), interactive tasks in graphical user interfaces, e.g., querying and marking places of interest with Google Maps. Unlike previous approaches, TOME uses the crowd wisdom of end users expressed in interaction histories to model task executions. With this, we explore performance modeling to guide development of a prototype interactive brain-circuit map.

The contributions of this work include an early implementation of TOME and a case study with a brain-circuit visualization that demonstrates the framework's prediction accuracy for task completion times and usefulness for evaluating new interaction designs. We show that performance predictions for two circuit query tasks average within 10% of expert performance, and we extend one TOME-generated model to evaluate a proposed feature that speeds up one task by 16%.

DESIGN EVALUATION BY PERFORMANCE MODELING
Models of human performance with a tool can be used to guide design choices. Our work uses the KLM, which predicts the time an expert user takes to execute necessary keyboard and mouse input and also cognitive operations (e.g., "mental preparation"). Here, an 'expert user' is an application end user who knows the steps necessary to complete a task and can do them as quickly as possible. A prediction should be close to a lower bound on how long it takes to execute the critical interaction path for completing a task.

The TOME framework (shown in Fig. 1) gives a performance prediction for a task by: 1) collecting histories of that task as end users execute and label them; 2) determining a *canonical* interaction sequence – or, what end users might reasonably do – that completes the task; 3) compiling a canonical history into a project for CogTool, which computes time predictions from storyboards of interactions [7].

Collecting Histories
TOME provides an interface instrumentation library based on Java's Swing toolkit that automatically produces interaction histories as end users of applications complete tasks. Library widgets like buttons are meant to be instantiated in place of respective Swing components. A basic logging API can be used to capture other events and build logging widgets. There is a one-time cost of instrumenting an interface, and these

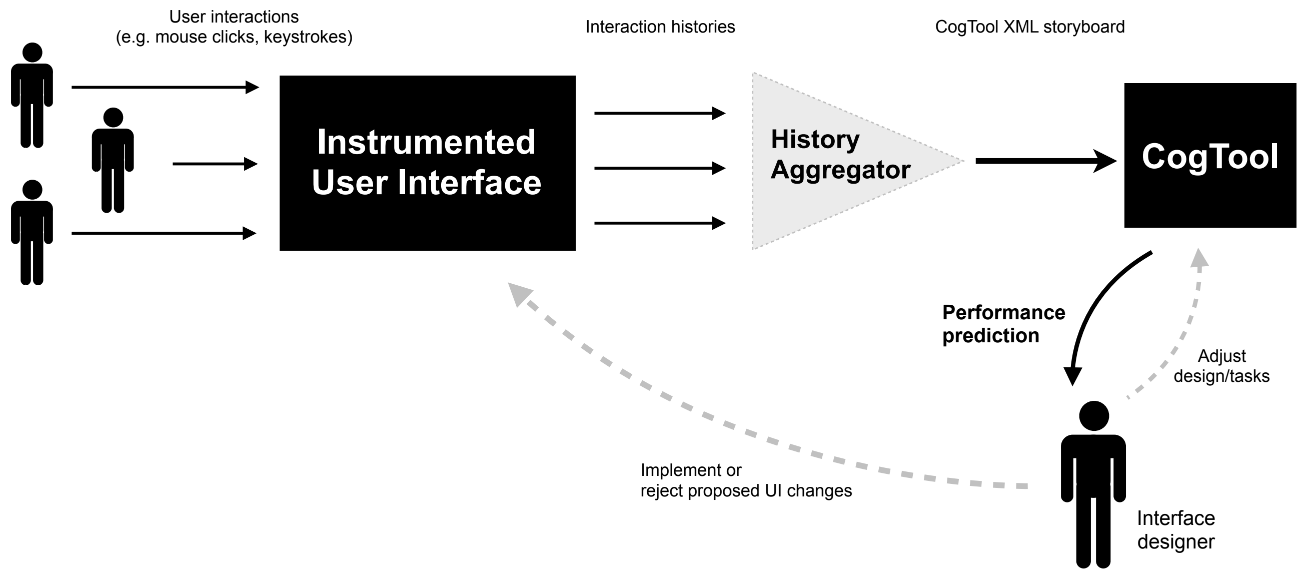


Figure 1. The TOME pipeline. Interaction histories are generated when end users complete tasks with the instrumented UI. Histories are aggregated by a program into canonical interaction storyboards for each task; CogTool then produces time predictions from these storyboards. The dotted arrows show actions a UI designer might take having retrieved the performance prediction from CogTool.

applications can be deployed 'as is'. End users can toggle logging on or off by editing a configuration file. Toggling the configuration does not affect regular application functionality, allowing end users to opt out of data collection easily.

Histories are encoded as sequences of widget-triggered interaction events and corresponding screenshots and keyboard or mouse input. In essence, each history gathers the information needed to build a graphical storyboard of the input events that cause GUI state changes throughout the task. Other subtle data is collected; for instance, the on-screen spatial bounds of widgets used are reported to model mouse-targeting times by Fitts' Law [9].

Finding Canonical Interactions
A unique aspect of this work is using many histories to produce a single time prediction for a task. The idea is that for certain types of tasks, the crowd wisdom for how to complete the task can be extracted from a set of real end-user histories.

In our implementation, when a history aggregation program is run, histories are grouped by labels that end users provide after finishing tasks. Within a group, histories that share the same interaction sequence are counted, and the most frequent sequence is treated as the canonical one for the task. This approach filters out noisy task executions (e.g., including accidental mouse or keystroke events) or unpopular strategies without having to interpret the semantics of histories. Furthermore, unlike applying the KLM manually, no modeler must know and express how to complete tasks *a priori*.

Creating a CogTool Project
The program then compiles a single history with the canonical sequence into an XML encoding of a CogTool project that describes a storyboard of the task execution (see Fig. 2). Finally, CogTool can open the project and run the model to predict completion time.

The ability to edit these storyboards in CogTool makes our approach more powerful than simply gathering average times from history timestamps; we can compare current UI designs against proposed changes by copying TOME storyboards and perturbing them in the WYSIWYG editor to reflect incremental design changes. This utilizes both CogTool's rapid prototyping ability and TOME's ability to gather baseline models for how end users currently complete tasks. We describe an example design revision in the section titled "Evaluating New UI Features".

Related Work
Automating evaluation methods is an important challenge for interactive tools [6]. We build on previous projects aimed at making KLM more accessible as an evaluation method. Hudson et al.'s CRITIQUE [5] generates KLM predictions by having an individual demonstrate typical tasks on an instrumented prototype UI. Unlike CRITIQUE, TOME considers histories and task strategies from real end users and tolerates noisy interactions. John et al.'s CogTool [7] lets analysts build mock UIs for modeling-by-demonstration in a WYSIWYG editor. TOME uses CogTool for editing the storyboards it builds; it minimizes the work needed to evaluate design changes with CogTool alone.

Other previous work explored interaction histories for usability evaluation [4], but to the best of our knowledge multiple end-user histories have not been leveraged for easier KLM modeling. Though TOME histories are not used by the instrumented applications, history-based features have been developed in tools like Tableau [3] and VisTrails [1] to support end users in creating data visualizations or rendering dataflows.

CASE STUDY: INTERACTIVE BRAIN DIAGRAMS
We incorporated TOME into the development of a sample interface, which was an interactive visualization prototype of

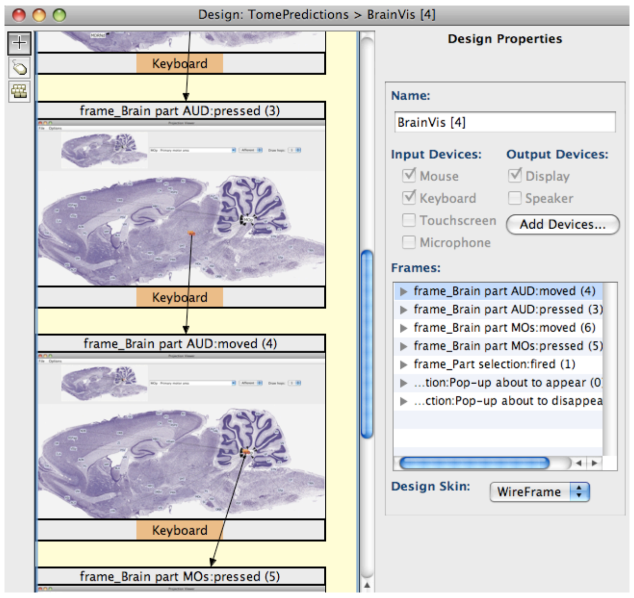

Design: TomePredictions > BrainVis [4]
Keyboard
frame_Brain part AUD:pressed (3)
Keyboard
frame_Brain part AUD:moved (4)
Keyboard
frame_Brain part MOs:pressed (5)
Design Properties
Name:
BrainVis [4]
Input Devices:
Output Devices:
Mouse
Display
Keyboard
Speaker
Touchscreen
Add Devices...
Microphone
Frames:
frame_Brain part AUD:moved (4)
frame_Brain part AUD:pressed (3)
frame_Brain part MOs:moved (6)
frame_Brain part MOs:pressed (5)
frame_Part selection:fired (1)
...tion:Pop-up about to appear (0
...ction:Pop-up about to disappea
Design Skin:
WireFrame

Figure 2. The CogTool interface showing a storyboard constructed by TOME during the T2 task. The arrows between frames indicate GUI state transitions caused by user interactions with widgets (located at the base of each arrow).

the rat brain circuit (Fig. 2 thumbnails). To establish the accuracy of TOME's predictions, we instrumented this interactive diagram to collect task histories, and then compared the KLM predictions with measured task completion times. Furthermore, we modified one model in CogTool to predict the performance improvement given by a proposed feature.

Experimental Design

To collect test histories, eight participants were recruited as application end users and completed two types of tasks with the brain-diagram tool, as described below. All were undergraduate or graduate students in computer science. The participants were split into two groups (A and B) that completed the task types with different brain part queries. Using two groups with different instances of data gives more model predictions to compare, and therefore more confidence in generalizing that these task types can be predicted with the KLM.

With the informed consent of each participant, we recorded participant videos and screen capture for posterior analysis. Participants were trained with the brain node-link diagram for 10-15 minutes and asked to complete the following tasks as quickly and accurately as possible:

T1: *'Nearest neighbor' neural projections.* Given the name of a specific brain part p, *select* the two nearest parts on the map that share a projection (edge) with p.

T2: *Map adjustment.* Given the names of two specific brain parts p_1 and p_2 and a target part t, click and drag both p_1 and p_2 on top of t.

In both tasks, participants were required to interpret the diagram and complete several motor activities using the keyboard and mouse.

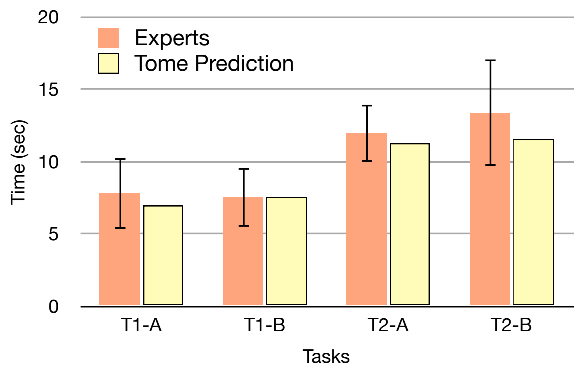


Figure 3. Average prediction error for these models is less than 10%. Error bars show ± 1 std. dev.

Each participant completed each task 25 times during a session of about an hour. The first five runs in each task tested the subject on all different brain parts so as to increase familiarity with the tool and task. The remaining 20 runs of each task were repeated with the same query in order to estimate the average *expert* completion time (mean from runs 11–20, using a timer) to compare to KLM. Of the 160 total expert runs collected, 5 times were discarded from this mean due to users stopping or encountering technical problems in these trials. Runs 1 through 10 for each task were training data (histories) for TOME to construct the canonical storyboard.

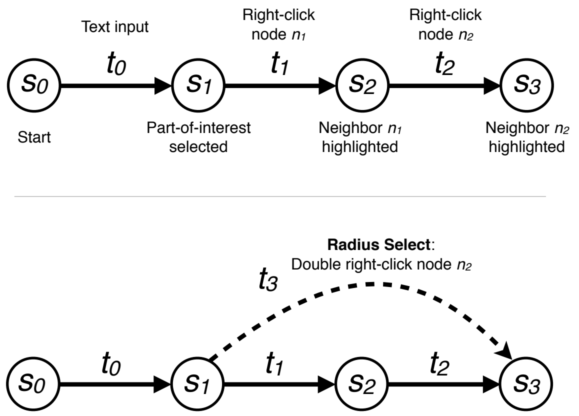
Evaluating New UI Features

After gathering histories and building TOME storyboards, we extended one of these models to evaluate a new feature before implementing it. We used a model created for the T1 task to evaluate an interaction called *radius select* that makes T1 faster. With radius select, a user can select all brain parts within a circular area of interest by choosing a central brain part and a radius on the map; this interaction can thus solve T1 quickly, without individually selecting nearby nodes. We used CogTool to edit the T1 model built by TOME after our experiment (see interface on Fig. 2). This amounted to adding one transition triggered by a new mouse action to the previously constructed storyboard. We simulated radius select in CogTool to produce a time prediction for experts.

Results

Figure 3 shows results for prediction accuracy for the tasks described previously. The worst error was just under 14%, on group B's T2 task. Reviewing the video for this instance showed that one participant repeatedly deviated from the most popular strategy that TOME automatically storyboarded; this participant's significantly slower task executions raised the mean expert time. Performance times over repeated trials became more consistent with experience. For both groups A and B, the standard deviation of all training set times (runs 1-10) was at least 50% higher than in expert trials (runs 11-20) for each task.

We extended the T1-A storyboard to include the *radius select* interaction. The prediction for the T1-A task using this feature was 5.7 seconds, 18% faster than the original prediction (6.9 sec). We implemented this feature and tested it with four



Text input
Right-click node n_1
Right-click node n_2
t_0
t_1
t_2
s_0
s_1
s_2
s_3
Start
Part-of-interest selected
Neighbor n_1 highlighted
Neighbor n_2 highlighted
Radius Select: Double right-click node n_2
t_3

Figure 4. Updating a storyboard. States $s_0 \ldots s_3$ and transitions $t_0 \ldots t_3$ express the storyboards for task T1 in the original (top) and modified (bottom) designs. The dotted arrow shows the transition we added in CogTool to predict the performance improvement given by this feature.

participants – two from group B and two new participants – using the previous protocol. One of the 40 expert runs collected was discarded due to a technical problem during the trial. Expert times using radius select are about 16% faster than previous expert times.

DISCUSSION

We were able to predict task times with reasonable accuracy for two diagram-query tasks. Simple design iterations can be evaluated by 1) using TOME to get a model of task performance with the current interface, then 2) using CogTool's editor to insert design changes into the storyboard, and rerunning the model to retrieve a prediction. For the *radius select* interaction we considered, the predicted speed-up was compelling enough to implement. For the tasks studied, we also confirmed that participants became more consistent in completion time over the course of trials. This validates the choice of tasks in the study. Tasks where completion times do not become more consistent or appear to converge over many trials are unlikely to be predicted well with the KLM.

Limitations

We evaluated only a small number of end users and tasks in a lab setting. An extensive, longitudinal study of end users completing tasks *in situ* would be more ecologically valid than having participants repeat trials in hour-long sessions.

The main limitation with TOME itself is that only certain kinds of tasks can be modeled with the KLM. Some tasks, like freely exploring a visualization, usually do not have predictable interaction steps that make sense to model with the KLM. Additionally, tasks that can be modeled must be executed in a TOME-instrumented interface. Instrumenting an interface and editing storyboards in CogTool requires time and learning. While our experience suggests that editing a TOME-built storyboard (as in Fig. 4) in CogTool is faster than building one from scratch, we did not evaluate the time and difficulty involved.

Automating TOME further could make it easier to use in live settings. An open problem is automatically classifying interaction histories with task labels. What processes are needed to differentiate noisy executions from divergent strategies or different tasks? Currently, end users manually label their task histories, but this bookkeeping may be tedious or error-prone.

CONCLUSION

We have described work toward a novel architecture for modeling human task performance from multiple interaction histories. Unlike previous methods, our system does not require an HCI expert to predict and model the steps taken by crowds of end users to complete tasks with an interface. Limitations of this approach include those of the KLM and that end users must label their task histories. Modeling higher-level cognitive processes with minimal human expertise remains an important challenge. Still, our results are encouraging: for quick diagram-query tasks, we demonstrated that TOME generates predictions within the 20% error claimed by KLM [8] and that these models can be used to evaluate iterative designs.

ACKNOWLEDGMENTS

This work was supported in part by NSF award IIS-10-16623 and NIH award R01-EB004155. All opinions, findings, conclusions, or recommendations expressed in this document are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

REFERENCES

1. Callahan, S. P., Freire, J., Santos, E., Scheidegger, C. E., Silva, C. T., and Vo, H. T. Vistrails: visualization meets data management. In *ACM SIGMOD* (2006), 745–747.
2. Card, S. K., Moran, T. P., and Newell, A. The keystroke-level model for user performance time with interactive systems. *Comm. of the ACM 23*, 7 (1980), 396–410.
3. Heer, J., Mackinlay, J., Stolte, C., and Agrawala, M. Graphical histories for visualization: supporting analysis, communication, and evaluation. *IEEE TVCG 14* (November 2008), 1189–1196.
4. Hilbert, D. M., and Redmiles, D. F. Extracting usability information from user interface events. *ACM Comput. Surv. 32* (December 2000), 384–421.
5. Hudson, S. E., John, B. E., Knudsen, K., and Byrne, M. D. A tool for creating predictive performance models from user interface demonstrations. In *ACM UIST* (1999), 93–102.
6. Ivory, M. Y., and Hearst, M. A. The state of the art in automating usability evaluation of user interfaces. *ACM Comput. Surv. 33* (December 2001), 470–516.
7. John, B. E., Prevas, K., Salvucci, D. D., and Koedinger, K. Predictive human performance modeling made easy. In *ACM CHI* (2004), 455–462.
8. John, B. E., and Suzuki, S. Toward cognitive modeling for predicting usability. In *HCI International* (2009), 267–276.
9. MacKenzie, I. S., and Buxton, W. Extending Fitts' law to two-dimensional tasks. In *ACM CHI* (1992), 219–226.