
Towards a New Method for the Evaluation of Reality Based Interaction

Georgios Christou
Cyprus College
6 Diogenes st.
Engomi, Nicosia, 1516
Cyprus
christoug@acm.org

Abstract

In this paper we present work toward a new method of evaluation for Reality-Based Interaction Styles that we call Cognitive Description and Evaluation of Interaction (CoDeIn). This framework is similar to GOMS, but also shares similarities with cognitive architectures, such as ACT-R and Soar. We apply this new approach to a Tangible User Interface, built as a Wizard of Oz interface, to illustrate the new method, and compare CoDeIn's results to a GOMS model. We show that CoDeIn provides some promise of representing the parallel processing and new modes of physical interaction inherent in reality-based interfaces, and CoDeIn does a better job in predicting the completion time of an example task than GOMS.

Keywords

GOMS, RBI, Reality-Based Interaction, Interface Evaluation, Evaluation Method, cognitive architecture

ACM Classification Keywords

H.5.2 [Information Interfaces and presentation]: User Interfaces—evaluation/methodology, theory and methods, GOMS; H.1.2 [Models and Principles]: User/Machine Systems — human factors, human information processing.

Introduction

Today there are many methods for the evaluation of the Direct Manipulation (DM) interfaces [5, 11], with one of the most prominent being the GOMS (Goals, Operators, Methods, Selection Rules) family [1, 9]. But as new interaction styles are being created to take advantage of quickly evolving technology, we wonder whether existing evaluation methods can be used to evaluate tasks in these new interaction styles.

Here we propose a new method that particularly supports the evaluation of Reality-Based Interfaces (RBIs) [6, 7], and may extend our ability to evaluate existing interfaces. The method evaluates a task in terms of the users' required knowledge to perform that task. This is done through a diagramming notation that allows the designer to model the interactions allowed in a RBI interface, given the interface actions and the required task knowledge.

To evaluate the proposed method, we present a preliminary experiment that compares the findings of the proposed method to GOMS. The result is that the proposed method performs better than GOMS in the RBI context.

Background

GOMS is one of the most widely known model based evaluation methods in HCI. Based on the goals of the user to perform a task, it analyzes the task using operators, methods and selection rules. GOMS though, presents some well known problems in the evaluation process. First, it only applies to expert, error-free performance [1], which excludes evaluation for occasional users who are among the most frequent users of RBIs. Second, only one of its varieties, CPM-

GOMS [3], allows the evaluation of parallel tasks, something that seems common in RBIs [8], however, CPM-GOMS is very complex for most evaluation analyses [10].

The proposed evaluation method introduced here is based on an idea presented by Christou and Jacob [2], which stated that a task can be evaluated on the amount of knowledge required for the performance of that task. The method uses the Keystroke Level Model operators [1] for evaluation of other portions of the task. The notation of the proposed method is based on Augmented Transition Networks (ATNs) and is inspired by Statecharts [4].

CoDeIn: the Proposed method

The proposed method evaluates a task based on the required task knowledge. We name this method Cognitive Description and Evaluation of Interaction (CoDeIn). The knowledge that is modeled in CoDeIn is of two types: declarative knowledge and procedural knowledge. We assume that domain knowledge is constant across interfaces, and that the user already possesses the required domain knowledge for the task. Therefore, we do not take domain knowledge into account for the proposed method for comparing interfaces.

Because the task is modeled based on the knowledge required to perform the task, we move away from the concept of goals, and goal-based evaluation methods; the method uses the designer's model of the task, and not the user's. This is important, because when actions that may be performed in parallel are modeled, the model reflects this, even though the user may not choose to perform the actions in parallel. This may

seem to undermine the method's validity, but we argue that it does not because we expect that expert users (the ones modeled in the case shown in this paper), will perform the actions trying to be as efficient as possible, thus utilizing any "shortcut", like performing actions in parallel.

We also have some preliminary promising results in modeling non-expert users, by using the same model and fuzzy logic rules, on the required task knowledge. We do not however, present those in this paper.

Notation

The diagrammatic notation to describe and evaluate interaction tasks is shown in Figure 1. It uses an ATN to represent the flow of the task from one action to the other, and allows zooming into task states to see how they are carried out, like Statecharts [4].

Figure 1 includes four components: the rectangle that represents a knowledge state, a declarative knowledge chunk, modeled as a circle, a procedural knowledge chunk modeled using a triangle, and an action that moves the user from one knowledge state to another, shown as an arrow. Because we propose modeling the task, and not modeling the interface, the knowledge state does not represent a specific state of the interface. Rather, it represents a state in the process of the task, a subtask. Each knowledge state may either hold other knowledge states, or required knowledge for the performance of the action coming out of the state. The arrow may also have a condition, which defines when the transition will be followed, and not when the action will be performed. In any knowledge state, when an arrow exists, the action may be performed as long as the knowledge requirements of the knowledge state

are satisfied. Action conditions are optional, and if not included, then the action may be performed when the corresponding state is entered.

The evaluation process is best shown by applying it to an example task. We next describe an experiment that gathered data to compare the performance of CoDeIn to GOMS.

Experiment

PARTICIPANTS

10 participants were recruited from the Computer Science department of Cyprus College, ranging in age from 19 to 25 years old. All were taking an introductory HCI course at the time of the experiment, and participated in the study for course credit.

PROCEDURE

Participants were asked to perform the following task: "Find a specific mp3 file in a list of folders, where the target file, the source and target folders are given, and move the requested file from the source folder to the destination folder". For example, the participants would be asked to find the mp3 file called "Cleaning my closet" by Eminem, which would be in the "Sting" folder (source folder), and move it to the correct (destination) folder, which in this case would be the "Eminem" folder.

Figure 2 shows the layout that was used for the experiment. The experiment was performed by asking the participants to reach into the source folder, grab the file objects and search through them. When they would find the target mp3 file they would proceed to place it into the destination folder. This signaled the end of the trial. The whole task is shown as an NGOMSL model in Table 1.


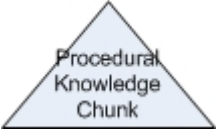


Description	Symbol
Knowledge State	
Procedural Knowledge Chunk	
Declarative Knowledge Chunk	
Action	

Figure 1 The diagrammatic components that make up the syntax of the proposed evaluation method

<p>Top Level Method (8.95s): Method for Goal: Move file from source folder to destination folder Step 1: Locate source folder (M = 1.2s) Step 2: Reach into source folder and grab files (R = 0.85s) Step 3: Examine the name of the file object on top of the file object pile (negligible) Step 4: Decide: If the filename matches the required filename Then Accomplish Goal: Place File in Destination Folder (3.1s) Else Accomplish Goal: Find required file (3.5s) Step 5: Return with goal accomplished</p> <p>Method for Goal: Find Required File (3.5s) Step 1: Move file object from top of file object file to bottom. Step 2: Examine the name of the file object on top of the file object pile Step 3: Decide: If the filename matches the required filename Then Return with goal accomplished Else Accomplish Goal: Find Required File</p> <p>Method for Goal: Place File in Destination Folder (3.1s) Step 1: Locate destination folder (M = 1.2s) Step 2: Place file in the destination folder (R = 0.85s) Step 3: Place remaining files in source folder (R = 0.85s) Step 4: Return with goal accomplished</p>

Table 1 The NGOMSL methods for the evaluation of the experimental task. [move lower if you can, always after intro



Figure 2 The layout of the experimental task.

MATERIALS AND DESIGN

The task was designed as a Tangible User Interface (TUI), as shown in Figure 2, using a mockup of a physically realized interface. Because the interaction with the interface is what is of interest here, it was deemed unnecessary to actually build the computationally realized TUI to support the task. Thus, it was built as a wizard of Oz interface. Folders were represented by 16 x 14 cm cardboard boxes. The name of each folder was written on the front side of each box. Inside each box there were paper cutouts of file objects, measuring 7.2 cm wide and 5.4 cm long. Each folder had a minimum of 3 and a maximum of 11 file objects. The file objects represented the files that were inside each folder, with the name of the file they represented printed in a box in the middle of the paper cutout. The song/album pairs were the same for all participants. Each participant performed ten trials per condition.

During the experimental task, two things could happen in parallel. The first was that the participants would find the positions of the source and target folders first,

because the folder view was constantly in front of them. This parallel action is shown in the first knowledge state (upper left), by using only one M operator to show that the participants searched for the position of the two folders.

The second pair of possible parallel actions happens after the last knowledge state, when the participants place both the target file object and the rest of the file objects back to their respective positions in one motion. This is reflected by the two arrows coming out of the last knowledge state, and by only marking the completion time on one of the arrows.

Discussion

The estimated completion time results could not be found just by using the primitive operators offered by GOMS: reaching to grasp, or prehension, and the search through the file objects that allowed the participants to find the target object. This shows the incompleteness of GOMS as is, to evaluate RBIs. Thus, we performed another experiment that showed the average time for prehension, in this experimental setting, to be 0.85 s. This was used for both analyses, shown in the NGOMSL model as the R operator, and shown in the CoDeIn model of Figure 3 as the grabbing action.

The second action that GOMS could not handle was the search through the file objects. For this reason, we performed yet another experiment, where we isolated the search task, and found the average search time to be 3.5 s, in the same experimental setting.

Table 2 shows the results of the two analyses. Here, CODE'IN provides a more accurate result than GOMS.

	TUI	Deviation
Experiment Average	4.12	--
GOMS	8.95	4.74
CODE'IN	6.4	2.28

Table 2 Analysis results of task performance predictions and times (in seconds) for the interface along with deviations from the experimental time. Dashes are used where no available value exists.

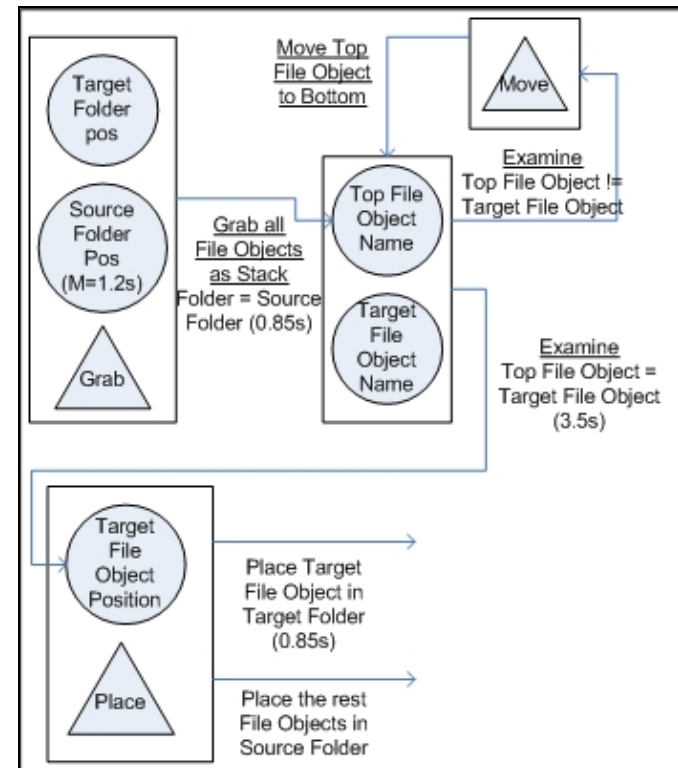


Figure 3 The CoDeIn model for the experimental task.

As mentioned earlier, we believe that CoDeIn can be used not only in the analysis of expert behavior for a task, but for any user, because of the explicit definitions of knowledge for each action.

Conclusions and Future Work

In this article we presented a preliminary experiment, carried out to compare the performance GOMS with CoDeIn, a new proposed evaluation method. The experiment included a task with a TUI condition,

representing the new trend in interaction styles based on computing embedded in the world, collectively called RBI.

While GOMS needed to be extended, CoDeIn managed to provide a good estimate for the completion time of the task. Also, because of the graphical notation, CoDeIn presents an easy way of modeling parallel actions. Another result that suggests further work with CoDeIn is that we believe it can model any type of user, unlike GOMS, which can only model expert performance. We are working towards this purpose now, by creating more complex experimental tasks, and using different types of users for our experiments, to gauge the performance of CoDeIn given the different user classes. We are also working on a set of operators, much like GOMS' primitive operators, that cover different actions that have not yet been modeled in GOMS, such as the prehension operator, mentioned earlier.

Acknowledgements

The author would like to thank Dr. Robert J. K. Jacob and Dr. Frank E. Ritter, who made this work possible through their advice and support. Portions of this research were supported by the National Science Foundation (Grant IIS-0414389).

Citations

- [1] S. K. Card, T. P. Moran and A. Newell, *The Psychology of Human Computer Interaction*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1983.
- [2] G. Christou and R. J. K. Jacob, *Evaluating and Comparing Interaction Styles*, in J. Jorge, N. Jardim Nunes and J. Falcao e Cunha, eds., *DSV-IS 2003: 10th Workshop on the Design, Specification and Verification*

of Interactive Systems, Springer Verlag, Funchal, Portugal, 2003, pp. 406-409.

- [3] W. D. Gray, B. E. John and M. E. Atwood, *Project Ernestine: Validating a GOMS Analysis for Predicting and Explaining Real-World Task Performance*, *Human Computer Interaction*, 8 (1993), pp. 237-309.
- [4] D. Harel, *Statecharts: A Visual Formalism for Complex Systems*, *Science of Computer Programming* (1987), pp. 231-274.
- [5] E. Hutchins, J. Hollan and D. Norman, *Direct Manipulation Interfaces*, in D. A. Norman and S. W. Draper, eds., *User Centered System Design: New Perspectives in Human-Computer Interaction*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1986, pp. 87-124.
- [6] J. K. R. Jacob, *CHI 2006 Workshop Proceedings: What is the Next Generation of Human Computer Interaction?*, Tufts University, Medford, MA, 2006.
- [7] R. J. K. Jacob, *Reality-Based Interaction: Understanding the Next Generation of Human-Computer Interfaces*, 2004.
- [8] R. J. K. Jacob, L. Deligiannidis and S. Morrison, *A Software Model and Specification Language for Non-WIMP User Interfaces*, *ACM Transactions on Computer-Human Interaction*, 6 (1999), pp. 1-46.
- [9] B. E. John and D. Kieras, *The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast*, *ACM Transactions on Computer-Human Interaction*, 3 (1996), pp. 320-351.
- [10] B. E. John and D. Kieras, *Using GOMS for User Interface Design and Evaluation: Which Technique?*, *ACM Transactions on Computer-Human Interaction*, 3 (1996), pp. 287-319.
- [11] B. Shneiderman, *Direct Manipulation: A Step Beyond Programming Languages*, *IEEE Computer*, 16 (1983).