Georgios Christou, Effie Lai-Chong Law, William Green, & Kasper Hornbaek (eds.) Challenges in the evaluation of usability and user experience in reality-based interaction (workshop proceedings). IRIT Press, Toulouse, France. At CHI 2009 Conference on Human Factors in Computing Systems, Boston, MA, 2009.

Knowledge-Based Usability Evaluation For Reality-Based Interaction

Georgios Christou
European University Cyprus
6 Diogenes St., Nicosia, Cyprus
gchristou@acm.org

Frank E. Ritter
College of IST
Penn State University
University Park, PA 16802, USA
frank.ritter@psu.edu

Robert J. K. Jacob
Tufts University
161 College Ave., Medford, MA 02155, USA
jacob@cs.tufts.edu

ABSTRACT
The great variety of new (Post-WIMP) interaction styles make them difficult to evaluate and compare. We propose a new evaluation method for them, Knowledge-Based Usability Evaluation (KBUE), that is based on similar ideas to those that drive cognitive architectures, such as ACT-R and Soar. We present KBUE as a way to formally specify the knowledge in the environment and in the user's head, and how this specification can be used to examine whether the aforementioned set of knowledge covers the required knowledge for the performance of a task in a user interface. We believe that by using this specification, it becomes easier to evaluate and compare Reality-Based interfaces.

Author 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 exist many different methods for the evaluation of Direct Manipulation Interfaces. However, it seems that these evaluation methods are not well suited for the evaluation of Reality Based Interaction (RBI) interfaces [4]. This is particularly true in cases where the RBI interfaces allow actions that cannot be modeled by the existing evaluation methods, or in cases where the nature of the interaction cannot be captured by these evaluation methods. An example of the first case is the reaching-and-grasping motion that is used in Virtual and Augmented Reality (VR and AR) interfaces, and in Tangible User Interfaces (TUI). This motion is not modeled in any of the existing evaluation methods, simply because it was not used in any of the existing interaction styles. An example of the second case is parallel performance of actions, although this example is being addressed currently by at least two versions of GOMS, GOMSL [15] and CPM-GOMS [10, 11], with different limitations in each version.

Because of problems such as the ones already mentioned, it is imperative that we create a new method that allows the evaluation of RBIs. This should not only predict the completion time of tasks in RBIs, but should also allow the direct comparison between designs of an interface in different interaction styles. In this paper, we propose a step towards the solution of this problem in the guise of a new evaluation method we call Knowledge-Based Usability Evaluation (KBUE).

BACKGROUND
The usability evaluation process is many times coupled to GOMS, because GOMS is one of the most widely known evaluation methods in HCI. Based on the goal of the user to perform a task, the GOMS method suggests the analysis of the task using operators, methods and selection rules. Kieras [14] has written extensively on how GOMS can be used to evaluate how interfaces support tasks. GOMS though, presents some well known problems in the evaluation process. First, it only applies to expert, error-free performance [12], which excludes evaluation for occasional users who are the most frequent users of RBIs. Second, only one of its varieties, CPM-GOMS [7], allows the evaluation of parallel tasks, something that seems common in RBIs [9], but CPM-GOMS is very complex for most evaluation analyses [11]. While other evaluation methods exist, few provide prediction capabilities, and those that do are usually specifically created for direct manipulation interfaces, such as, for example, the User Action Framework [2, 8].

KBUE has been used in different contexts than the one we propose here, such as by providing a tool to designers that provides easy access to knowledge about the design of user interfaces [17]. However, even in the way that we propose it here, KBUE is not a new concept per se. As an idea, it has been used in the literature [6, 18-21], although examples of this practice have been performed without any cohesion between them. For example, Ritter et al. [20] have used a cognitive architecture to evaluate a user interface, and Schoelles et al. [21] have used a cognitive architecture to predict human performance in a complex task environment.

Also, tools have been built that facilitate the description of tasks through task knowledge, such as [22], where St. Amand, Freed and Ritter describe a tool that takes an ACT-R model and transforms it into its GOMS equivalent.

Our view of KBUE however, is to use the knowledge that exists in the environment and the knowledge that is in the user's head, and compare whether that set includes all the required knowledge for the performance of a task in a user interface. This presentation of KBUE is based on an idea presented by Christou and Jacob [3], which stated that a task can be evaluated on the amount of knowledge required for the performance of that task, although it was not named KBUE at the time. We consider that it is possible to distill the knowledge required for the performance of a task, using any knowledge elicitation method (for a review see [5]). This is routinely done when creating task models in various cognitive architectures, such as ACT-R [1], Epic [13] and Soar [16].

KBUE USER SPECIFICATION

We consider that there are sets of knowledge that represent knowledge the in-the-head (Kn(h)) and knowledge in-the-world (Kn(w)). We can further consider that there is a set that contains all the required task-performance knowledge (Kn(t)).

As can be seen from table 1, we can define virtually any user, by considering the amount and nature of knowledge that the user possesses. This allows the creation of scenarios that can be analyzed to examine where the specified user will run into problems. Thus, the design of a task can be addressed in such a way as to provide the required knowledge in the environment, so that the user will be able to complete the task by retrieving this knowledge, even if they do not posses it.

For example, we can consider the case of the MS-DOS or the UNIX command-line interface vs. a Graphical User interface (GUI). The prompt presents no information about the execution of any task, therefore Kn(w) is effectively empty for every task in this environment. In the case of a novice user, then Kn(h) is also effectively empty. On the other hand, in the case of GUI, even if Kn(h) is empty, there are enough chunks of knowledge in the Kn(w), such as the pointer, the icons and their names, and in the case of MS Windows, the "Start" button, which suggests that the user should start from there. Thus, by a simple analysis of the amount of knowledge in the sets that define which knowledge about the interface is available to the user, it is easy to conclude that GUI would be easier to use by a novice user than a command prompt.

Knowledge Case	Explanation
$Kn(t) \subseteq Kn(h)$	One of several best case scenarios, where the user knows everything about the performance of the task.
$Kn(t) \subseteq Kn(w)$	One of several best case scenarios, where everything needed for the performance of the task is shown in the environment
$Kn(t) \subseteq (Kn(h) \cup Kn(w))$	One of several best case scenarios, where everything needed for the performance of the task is either known by the user, or is in shown in the world.
$(Kn(h) \cup Kn(w)) - Kn(t) \neq \varnothing$	Common case scenario, where the knowledge for task performance is not all known by the user, even in combination with the knowledge embedded in the environment. Thus, the need for user manuals and tutorials arises.
$(Kn(h) \cup Kn(w)) \cap Kn(t) = \varnothing$	Worst case scenario. The combination of the user's knowledge about the task combined with the knowledge embedded in the environment say nothing about the performance of the task.

Table 1 Example user descriptions

ADVANTAGES OF USING KBUE

KBUE is an attempt to provide solutions to the various problems that were mentioned in the previous sections. Using knowledge to create models of task performance provides various advantages over just using the actions that are perform to complete the task.

The first advantage of using knowledge is that we may vary the degree to which someone knows something, to model performance that is not expert. By varying the degree of knowledge of the various aspects of the task, we are able to model a range of users, from novices to experts [4], which is an advantage, especially over existing predictive evaluation methods, because by integrating this method into a predictive framework, one may develop ways to predict the behavior and performance of any type of user.

A second advantage is that modeling the knowledge required for a task vs. the knowledge the users have (or can readily find) provides a basis for comparison between designs of the same interaction in different interaction styles. For example, moving a file in augmented environment is not performed in the same way as moving it in a direct manipulation interface. However, only using task completion time to compare the two designs does not provide the whole picture of which interface is better in terms of usability and user experience. The reason is that task completion time only accounts for the efficiency of the task design, whereas other factors, such as learnability, and cognitive workload may need to be taken into account. Modeling knowledge on the other hand, may provide interesting insights on how the task is structured, and provide pointers to the redesign of the interface to perform the task.

CONCLUSIONS

In this paper, we have proposed a new method of evaluation called KBUE. This evaluation method is based on the supposition that we are able to gather three sets of knowledge about the task to be evaluated. First, the set that contains the required task-completion knowledge, second the knowledge-in-the-head and third the knowledge-in-the-world. By using these three sets, we can provide scenarios about what knowledge modeled users will need, and at what times, so that we can augment their knowledge with knowledge in the world that will aid them to complete the specified task.

We also discussed the advantages of this method over existing methods that were not designed to handle post-WIMP interfaces, and we have shown how the scenarios may become formalized, through a framework such as Codein, to allow predictive evaluation as well.

However, KBUE requires not only large-scale testing, but real-world testing as well. Real-world situations will provide indications as to how the theory may be applied, and in which cases it may be applied. Also, real world testing will provide clues as to the feasibility of this style of analysis in the context of large, complex projects. Finally, it will allow the evaluation of the results of such a process, and show whether the results that are provided are practical and usable.

REFERENCES

1. Anderson, J.R. and C. Lebiere. Atomic Components of Thought. Mahwah, NJ: Erlbaum; 1998.
2. Andre, T.S., et al., The User Action Framework: A Reliable Foundation for Usability Engineering Support Tools. International Journal on Human-Computer Studies, 2001; 54(1): 107-136.
3. Christou, G. and R.J.K. Jacob. Evaluating and Comparing Interaction Styles. In: DSV-IS 2003: 10th Workshop on the Design, Specification and Verification of Interactive Systems; 2003; Funchal, Portugal; p. 406-409.
4. Christou, G., CoDeIn: A Framework for the Description and Evaluation of Reality-Based Interaction. 2007, Tufts University: Medford, MA.
5. Cook, N.J., Varieties of Knowledge Elicitation Techniques. International Journal on Human-Computer Studies, 1994; 41(6): 801-849.
6. Das, A. and W. Stuerzlinger. A cognitive simulation model for novice text entry on cell phone keypads. In: Proceedings of the European Conference on Cognitive Ergonomics: ECCE 2007; 2007; London, UK; p. 141-147.
7. Gray, W.D., B.E. John, and M.E. Atwood, Project Ernestine: Validating a GOMS Analysis for Predicting and Explaining Real-World Task Performance. Human Computer Interaction, 1993; 8(3): 237-309.
8. Hartson, H.R., et al. The User Action Framework: A Theory-Based Foundation for Inspection and Classification of Usability Problems. In: HCI International '99 8th International Conference on Human Computer Interaction; 1999; p. 1058-1062.
9. Jacob, R.J.K., L. Deligiannidis, and S. Morrison, A Software Model and Specification Language for Non-WIMP User Interfaces. ACM Transactions on Computer-Human Interaction, 1999; 6(1): 1-46.
10. John, B.E. and D. Kieras, The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast. ACM Transactions on Computer-Human Interaction, 1996; 3(4): 320-351.
11. John, B.E. and D. Kieras, Using GOMS for User Interface Design and Evaluation: Which Technique? ACM Transactions on Computer-Human Interaction, 1996; 3(4): 287-319.
12. John, B.E., Information Processing and Skilled Behaviour, in HCI Models, Theories, and Frameworks, J.M. Carroll, Editor. 2003, Morgan Kaufmann: San Francisco, CA. p. 55-101.
13. Kieras, D. and D.E. Meyer, An Overview of the EPIC Architecture for Cognition and Performance with Application to Human-Computer Interaction. Human Computer Interaction, 1997; 12: 391-438.
14. Kieras, D. A Guide to GOMS Model Usability Evaluation using GOMSL and GLEAN3. 1999 [cited September 20th, 2006]; Available from: http://citeseer.ist.psu.edu/kieras99guide.html.
15. Kieras, D. A Guide to GOMS Model Usability Evaluation using GOMSL and GLEAN4. 2006 [cited

September 20th, 2006]; Available from: http://citeseer.ist.psu.edu/kieras99guide.html.

16. Laird, J.E., A. Newell, and P.S. Rosenbloom, SOAR: An Architecture for General Intelligence. Artificial Intelligence, 1987; 33(1): 1-64.

17. Lowgren, J. and T. Nordquist. Knowledge-based evaluation as design support for graphical user interfaces. In: CHI 1992 Conference on Human Factors in Computing Systems; 1992; Monterey, CA; p. 181-188.

18. Peebles, D.J. and A.L. Cox, Modelling interactive behaviour with a rational cognitive architecture, in Human Computer Interaction Research in Web Design and Evaluation, P. Zaphiris and S. Kurniawan, Editors. 2006, Idea Group Inc.: London, UK. p. 290 - 309.

19. Ritter, F.E. and R.M. Young, Embodied models as simulated users: Introduction to this special issue on using cognitive models to improve interface design. international Journal of Human-Computer Studies, 2001(55): 1-14.

20. Ritter, F.E., et al., Providing User Models Direct Access to Interfaces: An Exploratory Study of Simple Interface with Implications for HCI and HRI. IEEE Transactions on System, Man and Cybernetics, Part A: Systems and Humans, 2006; 36(3): 592-601.

21. Schoelles, M.J., et al. Steps towards integrated models of cognitive systems: A levels-of-analysis approach to comparing human performance to model predictions in a complex task environment. In: Proceedings of the 28th Annual Conference of the Cognitive Science Society; 2006; Vancouver, British Columbia; p. 756-761.

22. St. Amant, R., A.R. Freed, and F.E. Ritter, Specifying ACT-R Models of User Interaction with a GOMS Language. Cognitive Systems Research, 2005; 6: 71-88.

The columns on the last page should be of approximately equal length.