Reducing the Variability between Novice Modelers: Results of a Tool for Human Performance Modeling Produced through Human-Centered Design

Bonnie E. John
Human-Computer Interaction institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
412-268-7182
bej@cs.cmu.edu

Keywords:
Human performance modeling, Cognitive modeling, Keystroke-Level Model, KLM, HCI, HCD

ABSTRACT: *The variation between novice modelers has not been extensively studied, but it is important to organizations wishing to employ predictive human performance models in their system design process. This paper reports on the statistically-significant reduction in variation between novice modelers achieved by CogTool over the previously-established by-hand method of predicting the task execution time of skilled users (Keystroke-Level Model). CogTool was developed using human-centered design techniques specifically to understand and prevent novice errors by transforming the modeling process into an integral part of the system design process and these techniques seem to have worked.*

1. Introduction

The variability between modelers as they create human performance models has not been studied extensively. There have been comparisons between models in both AI and cognitive modeling, e.g., Sisyphus (Gaines, 1994), Project Halo (Chaudhri, et. al., 2009), the Ambr Project (Gluck & Pew, 2005) and the Predicting Cognitive Performance in Open-ended Dynamic Tasks Modeling Challenge (Lebiere, et. al., 2009), but each model in these comparisons is created by one person or team using their own modeling approach, and it is unknown whether a different person or team using the same approach would create a similarly-performing model.

The only instance of a comparison between modelers known to this author was a "by product" of a paper comparing different approaches to predicting skilled performance time on different user interfaces (UIs). Nielsen and Phillips (1993) were comparing heuristic estimation techniques to a predictive human performance modeling approach called the Keystroke-Level Model (KLM, Card, Moran & Newell, 1980) and provided data on 19 novice modelers building KLMs for two tasks on two UIs. This author followed up by publishing data from 8 additional novice modelers (John, 1994). In both instances, the coefficient of variance in these data hovered around 20%. This phenomena, called the "evaluator effect" in Human-Computer Interaction (HCI) has been shown for several different HCI techniques (e.g., heuristic evaluation (Nielsen & Molich, 1990) think-aloud usability studies (Jacobsen, Hertzum & John, 1998), and Cognitive Walkthrough (Hertzum & Jacobsen, 2001)).

The evaluator effect, about 20% for all the techniques yet studied, is particularly troublesome with a predictive human performance modeling technique like KLM, since it claims to have a prediction accuracy of about 20%, Thus, the variation between modelers is on the order of the expected accuracy of the technique itself and should therefore be of special concern to the behavior representation community.

This paper reports on an attempt to reduce the variation between novice modelers by providing tool-support for KLM analyses. Specifically, human-centered design (HCD) techniques were used to create a tool for constructing valid KLMs, called CogTool (http://cogtool.hcii.cs.cmu.edu/).

The next section reviews the original by-hand procedure to produce KLMs, what errors novice modelers tended to make using that procedure and how CogTool was built to obviate these errors. Section 3 describes the data assembled to establish the variability in KLMs created by both procedures. Section 4 analyzes the difference in variability between these two sets of models. Section 5 discusses the source of variation that remains in CogTool models and the final section maps future work stemming from these analyses.

2. Background

The KLM was introduced by Card, Moran and Newell (1980) as a method for predicting the task execution time of skilled users on UI design ideas before any code had been written to implement those ideas. The procedure for doing a KLM was to list all the overt actions that a user would have to take to accomplish the task: keystrokes on a keyboard or mouse clicks (K), pointing with a mouse (P), moving the hand between the mouse and keyboard (homing, H), and drawing (D, on a very constrained grid in a particular CAD system). The modeler then placed a single type of mental operator (M), to represent all the unobservable operations a user would perform, e.g., eye movements, memory retrievals, decisions, using a set of five heuristics defining where the Ms should appear in the model. These heuristics made distinctions between commands and arguments and depended on ill-defined terms like a "cognitive unit". Finally, if the system required its user to wait for it to respond, an R operator was included in the model.

Quantitative estimates for the KLM operators were established empirically (except for R, which must be estimated for each system), e.g., K=0.2s for an average skilled typist, P=1.10s for the average display size in 1980 (but could be calculated using Fitts's Law), H=0.4s, and M=1.35s. The modeler then added up these estimates to predict skilled execution time on the entire task. Doing a KLM "by-hand" means following this procedure using a spreadsheet to list the operators and do the addition.

I examined eight novice modelers' KLMs in detail to discover if systematic errors could be identified (John 1994). Comparing to the 87 operators that comprised a KLM that I created for these four tasks, that examination revealed several common errors.

1. Novice modelers leave out overt steps necessary to do the task. If you were to follow the exact Ks, Ps, and Hs listed in their KLMs, you would not complete the tasks successfully. Of all the overt operators left out by novices 31% were Hs, 31% were Ks, and 22% were Ps. Seven of eight modelers exhibited this error.
2. Conversely, three of eight novice modelers included extra overt operators, Ks and Ps that were not necessary to do the task.
3. Finally, all novice modelers seemed to find it very difficult to apply Card, Moran and Newell's heuristics for placing M operators. Some novices put in extra Ms in one place and omitted Ms from other places in the models, but all novice KLMs included more Ms than my KLMs for the same tasks.

This last problem has been exacerbated by the arrival of modern UIs. KLM was created in the era of command-line interfaces and command-based text editors, where it was relatively clear when something was a command or an argument. With direct-manipulation UIs, this distinction blurs. For example, when a user double-clicks on a word in a text-editor, is that operating on an argument or issuing a command to highlight the word? Card, Moran and Newell's heuristics are still applicable, but it takes interpretation and increasingly more experience to apply them to UIs as they evolve further from command-line operations.

In the early 2000s, under the support of ONR's Affordable Human Behavior Modeling Program, the above error analysis was one of several human-centered design (HCD) techniques used to design CogTool. The aim of the CogTool Project is to create a tool that allows UI designers to use predictive human performance modeling to evaluate their design ideas quantitatively before investing resources in programming those ideas. We used the aforementioned error analysis to guide the design of CogTool so that it would eliminate the identified errors as much as possible. We used Contextual Inquiry (Beyer and Holzblatt, 1998) to understand the pain points of cognitive modelers and how such a tool would fit into the workflow and culture of UI designers. We used competitive analysis to understand what had already been tried in this regard (Baumeister et. al, 2000), and a series of usability analyses (Cognitive Walkthrough (Polson., et. al. (1992), think-aloud usability studies, and, yes, KLM with an early version of CogTool itself). All results from these analyses were fed into the design of CogTool, and continue to be, so that CogTool is now being used in real-world design and evaluaton processes and taught to hundreds of HCI, UI design, and Human Factors students and professionals each year.

To do a KLM with CogTool, a modeler follows a very different procedure from doing a KLM by hand. Instead of listing overt operators in a spreadsheet divorced from a UI design, the modeler expresses the UI design in a graphical storyboard by placing pre-established widgets (e.g., buttons, check boxes, text fields) in frames that represent what users would see as they progress through a task. The modeler then connects those frames by drawing a transition from a widget to another frame, which represents the user's action that would cause the screen display to change (e.g., clicking on a button, typing on the keyboard). Finally, the modeler demonstrates a particular task on the storyboard, which creates a KLM by demonstration. CogTool creates ACT-R code (Anderson, et. al., 2004) from this demonstration and runs it to get the prediction of skilled execution time.

CogTool automatically places Ms consistently and in the correct position as suggested by Card, Moran and Newell's heuristics applied to modern UI widgets. Thus, CogTool has transformed the modeling process to a design process, where modelers decide what type of widget to use in their design rather than decide where a user might have to stop and think, addressing error (3) mentioned before. Errors (1) and (2) were addressed by the "modeling by demonstration" on the storyboard, as we surmised that modelers would be less likely to leave out or insert Ks and Ps if they were looking at a picture of the actual interface. Likewise, "bookkeeping errors" like forgetting to home the hand between devices should be eliminated because CogTool keeps track of where the simulated hand must be and automatically places H operators.

CogTool is now at a point where we can examine if it has met any of its aims. John et. al. (2004) demonstrated that a novice modeler could produce model estimates as well as an expert modeler. This paper examines whether CogTool has reduced the variability in novice modelers' models.

3. Data

I assembled data from previously-published papers that reported the results of groups of novice modelers creating KLMs on the same interfaces and tasks (Groups1&2), and from unpublished exercises in university classes (Groups3&4), to establish the variability of predicting skilled execution time with the original formulation of the KLM. I then acquired new data on 100 novice modelers using CogTool to investigate the variability of prediction with that modern tool.

3.1 Previously-collected data: Performing KLMs by hand

3.1.1 The interfaces and tasks

The groups who created KLMs by-hand were predicting the performance of skilled users of two telephone-number look-up systems described by Neilsen & Phillips (1993). The first interface, Design A Dialog Box, used menu selection, then a dialog box in which a telephone number was typed into a text field, and then a series of mouse clicks on on-screen buttons to submit a query. The second interface, Design B Pop-Up Menu, submitted the query through context menus accessed by clicking on displayed telephone numbers. Each modeler created four KLMs, looking up one telephone number and looking up two telephone numbers, on each of two interfaces. The predicted task execution times for these four tasks range from 5s (PopUp-1) to 22s (DialogBox-2).[1]

Ideally, designers of the system who know the screen layout and procedures for accomplishing tasks well are the people who create predictive human performance models for UI evaluation and design. To simulate this familiarity, the modelers were given step-by-step instructions showing what would be on the screen and what actions to take at each point in the tasks. We are looking for variability in the models they produce, not variability in how well they understand the interfaces, so this level of direction is appropriate and was used in all groups analyzed here.

3.1.2 ByHand-Group1

The data for ByHand-Group1 was published by Nielsen and Phillips (1993). The modelers were described as "19 upper-division undergraduate students in a human–computer interaction class as their second assignment using GOMS." Actually, the Keystroke-Level Model [1] was performed, not a full GOMS model (Erik Nilsen, private communication, 6 Sept 1993). Although no information was published about the instructional sessions or materials given to these students, it is likely that they were given one of the two publications about KLM by Card, Moran and Newell (1980 or 1983), as they were the readily available. Nielsen and Philips reported means and standard deviations for each of the four models for these 19 novice modelers. Because the magnitudes of the task execution times vary, the coefficients of variance (CV = standard deviation/mean) is calculated and appear on the first line of data in Table 1.

3.1.3 ByHand-Group2

The data for ByHand-Group2 was published by this author (John,1994). The modelers were "eight Carnegie Mellon undergraduate students at the end of their first HC1 class." The class was an elective offered in the computer science department, although students from other disciplines attended. These student had one lecture on KLM, one prior homework assignment on KLM, and Card, Moran and Newell 1980 was a required reading in the class. I "reproduced the Nielsen and Phillips interfaces from their descriptions" to create the materials given to the modelers. The means

[1] The purpose of KLMs is to predict skilled execution time and Nielsen and Phillips (1993) provided empirical data against which to compare those predictions. ByHand-Goup1 had an average absolute percent error of about 30%, whereas ByHand-Goup2, 3 & 4 had about 15%. No user data is available for the tasks and interfaces modeled by the CogTool-Group, regrettable, but not necessary to study variability.

and standard deviations for each of the four models for these 8 novice modelers were converted to CVs and appear on the second line of data in Table 1.

3.1.4 ByHand-Group3 & ByHand-Group4

The data for ByHand-Group3 was supplied by Wayne D. Gray (personal communication, November 28, 2009) from classes he taught in 1996 and 2002 using the same materials given to ByHand-Group2. The class, "Cognitive Task Analysis" was a core course in a masters program in Human Factors and Applied Cognition at George Mason University. These students had five weeks of other task analysis lectures but only one lecture on specifically how to do KLM and this was their first assignment using it. They were assigned Chapter 8 of Card, Moran and Newell (1983), which is essentially the same as Card, Moran and Newell, 1980. Twelve modelers were in the 1996 class and nine in the 2002 class. The means and standard deviations for each of the four models for these 21 novice modelers were converted to CVs and appear on the third and fourth line of data in Table 1.

3.2 New data: Performing KLMs with CogTool

The data labeled "CogTool" in Table 1 was recently generated in the "HCI Methods" class at Carnegie Mellon University (Fall 2009), which is a required class for the bachelors and masters programs in HCI and about ¾ of the students class are in those programs. All students in the class are in an undergraduate major other than HCI (the bachelors in HCI is a 2nd-major) or already hold a bachelors degree in another major, with about half from a technical background, ¼ from the behavioral sciences and ¼ from design in a school of fine arts. The class included 101 students, all of whom completed the assignment. One student had worked as a programmer on the CogTool Project the previous year and was removed from analysis because he had considerably more knowledge of the tool than the other modelers, resulting in an N of 100.

These students had one 1.5-hour lecture on predictive human performance modeling, about 20 minutes of which was a demonstration of CogTool. John (1995) was required reading and the students were encouraged to download the CogTool User Guide (http://cogtool.hcii.cs.cmu.edu/use-today/documentation-and-other-support). There was a 3-hour session where this author, a graduate student, and a programmer were available to answer questions about the mechanics of using CogTool (e.g., "I closed my Project window, how do I get it back?" and "I still have Tiger on my Mac and CogTool's not working, what do I do?"), but not about decisions that would effect predictions. About 2/3 of the students attended this session.

3.2.1 The interfaces and tasks

The interfaces and tasks modeled by this group were considerably more modern than those modeled by the other groups. Pragmatically, a teacher cannot continue using the same assignment for 15 years; students can get the answers from previous classes and they become so dated that they are irrelevant to the students' lives. Therefore, these novice modelers compared two web-based interfaces on three tasks, for a total of six models apiece.

The interfaces were real-world web services for cataloging books and sharing collections on-line: Booktagger (http://www.booktagger.com/) (Figure 1) and LibraryThing (http://www.librarything.com/). The tasks were (1) sign-in and add a book to your collection, (2) tag the book you just added, and (3) rate that book and sign-out. The task execution times for five of these tasks were on the order of those for the telephone look-up tasks (ranging from 5s to 48s). This

Figure 1. Booktagger page showing information about a book. Interactive widgets on this page include a textbox, buttons, links, checkboxes, and a star rating widgets.

assignment mimics what a designer would do in the real world to benchmark competitors' services before designing a new book-sharing service or the next release of an existing one.

As with the interfaces in the ByHand groups, these modelers were given step-by-step instructions of how to do each task on each interface with pictures of the screens that a user would encounter while doing the tasks. Again, we are looking for variation in predictions due to the modeling process, not due to a modeler's misunderstanding of the interfaces or task procedures, so providing this detailed information is justified.

3.2.1 The data

Each modeler produced a quantitative prediction of task execution time and the bottom line of Table 1 shows the CVs for each of these six predictions. In addition, each modeler turned in a CogTool file, which contains all the information relevant to coming up with that prediction. These 100 files were analyzed to understand the source of variance, e.g., the decisions the modelers made that led to different numeric predictions, and will be discussed in Section 5.

4. Analysis of Difference in Variability

To determine whether the predictions produced by novice modelers creating a KLM by-hand are more variable than those produced by novice modelers using CogTool, we follow Dow (1976). Dow explored the statistical tests used by ornithologists to study geographical variation in birds from previous publications reporting only the N, mean and standard deviation (SD) of their observations. The N of these studies is often as small as 5 and, in Dow's exploration, not more than 65, their means often differ in magnitude, and SD is sometimes correlated with the mean and sometimes not. Thus, ornithologists face a situation similar to the data sets I have been able to assemble. Dow explains both a t-test procedure and an F-test procedure, finding each more conservative under different characteristics of the data and concludes that the t-test is marginally better for comparing variability when studies have both small (<22) and large N (>=22) as is the case for the studies compared here.

As the data were reported as individual models for each task on an interface, Table 1 shows 4 models (2 tasks x 2 interfaces) per study, for a total of 16 instances (N-mean-SD triples) in the ByHand condition. In the CogTool condition, Table 1 shows 6 instances (3 tasks x 2 interfaces) of N-mean-SD triples. I calculated the average CV, weighted by N for ByHand (CV=22%), and CogTool (CV=7%), and used Dow's equation to calculate the t value for the comparison:

$$t = (CV_1 - CV_2) / \text{sqrt}\,(SE_1^2 + SE_2^2)$$

where $SE = CV / \text{sqrt}(N)$

$N_{ByHand} = \Sigma N_{Group} = 48$

$N_{CogTool} = 100$

The resulting difference in CV is highly significant using a 2-tailed t-test as recommended by Dow ($t=6.3$, $df=146$ $p<0.0001$). Thus, we can conclude that the models produced using CogTool are less variable than those produced by-hand.

Table 1. Coefficients of variance (CVs) for each prediction of a task on an interface for four groups using KLM by hand and one group using CogTool.

Group	N	Interface/Task Coefficient of Variance (CV)					
		DialogBox 1 number	DialogBox 2 numbers	PopUp 1 number	PopUp 2 numbers		
ByHand-G1 (Neilsen & Phillips,1993)	19	0.22	0.24	0.14	0.17		
ByHand-G2 (John, 1994)	8	0.22	0.21	0.22	0.21		
ByHand-G3 (1996)[2]	12	0.19	0.13	0.35	0.33		
ByHand-G4 (2002)[2]	9	0.23	0.25	0.21	0.25		
		Booktagger Add book	Booktagger Tag book	Booktagger Rate book	LibraryThing Add book	LibraryThing Tag book	LibraryThing Rate book
CogTool (2009)	100	0.03	0.04	0.12	0.03	0.09	0.13

[2] Data supplied by Wayne D. Gray, personal communication, November 28, 2009

5. Discussion of Sources of Variability that Remains in CogTool Models

In addition to the numeric predictions, data exist on exactly what was in every CogTool model and can be analyzed to determine the source of the remaining variability. Unlike the analysis done of the eight by-hand KLMs (John 1994), it is intractable to visually inspect 600 CogTool models. As this had to be done to grade the students' assignments and give them appropriate feedback on their models, we devised a more automated way to focus our attention on deviations from an acceptable model.

CogTool files can be exported to several formats that help with this analysis. First, the demonstrations can be exported to a csv format appropriate for importing into Microsoft Excel. I created an acceptable CogTool file that contained models of all six tasks, exported their demonstrations to csv, and then imported them into Microsoft Excel. I inspected each line in these demonstrations and inserted one line for each possible deviations from the canonical solution. That is, if a line said "Left Click on the Sign-in Button", I inserted four "error lines" for (1) missing the step entirely, (2) using a transition other than a left-click. (3) using a widget other than a button, and (4) inserting an inappropriate system response time. If an interface object could be reasonably construed as more than one widget, e.g., it is often difficult to decide whether some object on a web page is a button or a link, and the decision between these two would not influence the numeric outcome of the models (see the CogTool User Guide, Appendix C, for a description of the equivalent widgets), then I annotated the error-line to allow multiple answers. For example, error-line (3), above, would be changed to "using a widget other than a button or a link." This resulted in 28 steps that could be influenced by modeler decisions, for a total of 164 error-lines, i.e., opportunities to differ from an acceptable solution.

Again, visually inspecting 100 files for 164 possible errors, is intractable. However, a CogTool file also can be exported to an XML representation that preserves all the components of all the models in the file (the frames, widgets, transitions, and demonstrations). I exported my CogTool file to XML and scripts were used to compare this XML to each novice modeler's XML, highlighting those sections that differed in ways important to the results of a model (e.g., when using a different type of widget, but not when giving a widget a different name). With this highlighted file, four teaching assistants then visually inspected the difference between the novice's XML and the canonical XML and entered a "1" in the appropriate error-line in the Excel file for that particular difference. This resulted in a Excel chart with 164 rows of possible errors, 100 columns of novice modelers and a matrix of 1s and blanks representing the correct decisions (blanks) errors (1s) each novice modelers made. This matrix was manipulated to find the following sources of variability in the CogTool models.

Recall that in the error analysis of by-hand KLM (John 1994), all eight novice modelers deviated from the canonical model. "The student with the least deviation left out only 1 operator and added only 1 extra operator to the instructor's 87 operators. The student with the most deviations left out 25 operators and added 8 operators to the instructor's 87. (John, 1994, p. 286). With the CogTool models, 3 of 100 students did not differ at all from an acceptable model despite 164 opportunities to do so; 26 differed 1-4 times; 17 differed 5-8 times. Therefore almost half the novice modelers (46) made only 5% of the errors that were possible in this exercise. About one quarter (27) made 5-10% of the possible errors and the remaining quarter (27) made 10-20% of the possible errors, with the average being 7% and the median being 6%.

Recall that forgetting an H operator (homing) was very common in by-hand KLMs. CogTool automatically keeps track of the hand and inserts Hs if the hand must move between the mouse and the keyboard to complete the steps, so these types of errors should not occur in CogTool models. There were 11 H operators across the 6 tasks, for a total 1100 H operators possible in the combined novice's models and 128 Hs were missing, the most common type of error in the models. This occurs because, although CogTool keeps track of the hand as it goes through the task, the modeler must tell CogTool where the hand starts at the beginning of each task. The current default is for the hand to start on the keyboard, but all 6 tasks had the hand starting on the mouse (which was told to the modelers in the written assignment). 14 modelers did not set the hand's starting position to the mouse in all 6 tasks; 32 modelers did not set the position at least once. The starting position is set with a pulldown menu in CogTool's interface and may have been accidentally overlooked. We will investigate changing that interaction to a more salient one in future releases of CogTool.

Forgetting other operators (keystrokes, Ks, and pointing, Ps) was also prevalent in KLMs done by hand. However, of the 3500 decisions to insert such a step, only 1% (42) were forgotten by the novice modelers using CogTool. The vast majority of these, 30, were forgetting to click in a text box before typing into it. Both interfaces required this action at some point in the tasks (though not consistently), and (as with the by-hand KLMs) the modelers were told about these steps, so why they forgot them is as inexplicable

in the CogTool case as it was in the by-hand case. Perhaps novice modelers are still overwhelmed with the modeling activities, even with CogTool that if the tool does not enforce every step, novices will "just forget." If this were a running system rather than a storyboard mock-up, the system would prevent the task from progressing if it indeed worked required a click before typing. However, programming a running system defeats the purpose of predictive human performance modeling. I know of no way to solve this problem at this time, but at least it is reduced to less than 1% of the steps with CogTool. (The other 12 forgotten actions were evenly spread across other steps in the tasks with no apparent pattern.)

Recall that placing M operators was difficult for modelers doing KLM by-hand. CogTool modelers do not place Ms at all; the Ms are placed automatically by CogTool depending on the widget choices. There were 21 widgets necessary to do all 6 tasks. Two of the 21 were more difficult and will be discussed next, but of the 1900 relatively straightforward choices, only 5% (100) contained errors. Of these, 58 were choosing some widget other than a link in 4 different frames. In many modern websites, links don't follow old visual conventions (e.g., underlined text), so the distinction between links and buttons is murky. However, this choice does not influence the outcome of the CogTool predictions, so this common "error" may be considered more of style than substance.

The next most common error in widget choice was 25 choices of something other than a text box widget in three frames. CogTool distinguishes between text boxes and the text inside them. This distinction does make a difference to the predictions (see Appendix C of the CogTool User Guide) and is a known difficulty for novice modelers. There are several sections written in the CogTool User Guide about the difference between these widgets, when to use each one, and how to use them in concert to mock-up editing text, but this prevalent error indicates that either novice modelers do not read the User Guide or do not understand its information as written. Further investigation is necessary to understand how to eliminate this source of variance through redesign of CogTool itself or the documentation and training associated with it.

Two of the widgets in the models were quite difficult because they did not map directly to widgets supplied by CogTool. Both systems had a rating feature where a user clicks one of 5 stars to rate the book. Is each star a button widget? Is the set of stars equivalent to a set of radio button widgets? The novice modelers were asked to choose a widget and justify that choice. The scoring judged the justification, not the actual choice of widget. Thus, the 58 errors (29 modelers making the same error in both systems) were more for the modeler's ability to articulate their decision as opposed to actually making the right decision (which is to represent them as a set of radio button widgets). As new interaction styles are designed, modelers will encounter this problem of mapping CogTool's widgets to those interaction styles. How to best do so, or grow CogTool's widget set to accommodate innovative design, is an area for further research.

6. Conclusion

The evidence seems clear; CogTool has achieved its aim to reduce the variability in models created by novice modelers. In fact, with an average CV of 7%, it is the least variable of any usability evaluation technique studied to date. We attribute this success to using HCD methods (Contextual Inquiry, error analysis, usability evaluation, etc.) in the development of the CogTool. Modelers are simply another type of user and HCD methods (despite their variability), when used in concert and when they provide converging design advice, simply work.

7. Acknowledgements

The author thanks Joanna Bresee Brian Lim, Min Kyung Lee, Bryan Pendelton, (the teaching assistants in HCI Methods F09) for their careful identification of errors in the CogTool models, Wayne Gray for archiving and contributing data, and to Nicholas Yee for his help in the statistical analysis. This research was supported in part by funds from ONR (N00014-03-1-0086), NASA, NEC, PARC, and IBM. The views and conclusions in this paper are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of ONR, NASA, NEC, PARC, IBM, or the U.S. Government.

8. References

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review, 111*(4), 1036-1060.

Baumeister, L., John, B. E. & Byrne, M. (2000) A Comparison of Tools for Building GOMS Models. *Proceedings of CHI, 2000* (The Hague, The Netherlands, April 1-6, 2000) ACM, New York. pp. 502-509.

Beyer, H., & Holtzblatt, K. (1998). *Contextual design: Defining customer-centered systems*. San Francisco, CA: Morgan Kaufmann Publishers.

Card, S. K., Moran, T. P., & Newell, A. (1980). The keystroke-level model for user performance time with interactive systems. *Communications of the ACM. 23* (7), 396-410.

Card, S. K., Moran, T. P., & Newell, A. (1983) *The Psychology of Human-Computer Interaction*. Mahwah, NJ: Lawrence Erlbaum Associates.

Chaudhri, V. K., Clark, P. E., Mishra S., Pacheco, J., Spaulding, A., Tien, J. (2009) *AURA: Capturing Knowledge and Answering Questions on Science Textbooks*. Technical Report SRI International, http://www.ai.sri.com/pubs/files/1768.pdf

Dow, DD (1976). The use and misuse of the coefficient of variation in analysing geographical variation in birds. *Emu* 76, 25–29.

Gluck , K., A., and Pew, R. W., (2005) *Modeling human behavior with integrated cognitive architectures: comparison*. Mahwah, NJ: Lawrence Erlbaum Associates.

Gaines, B. R., (1994). Problem statement for Sisyphus: Models of problem solving. *International Journal of Human-Computer Studies*, 40, 187-192.

Jacobsen, N. E., Hertzum, M. & John, B. E. (1998) The Evaluator Effect in Usability Studies: Problem Detection and Severity Judgments. *Proceedings of the Human Factors and Ergonomics Society 42nd Annual Meeting* (Chicago, October 5-9, 1998), pp. 1336-1340. HFES, Santa Monica, CA.

Hertzum, M. and Jacobsen, N.E. (2001), The evaluator effect: A chilling fact about usability evaluation methods, *International Journal of Human-Computer Interaction, 13*(4), 421-443.

John, B. E. (1995) Why GOMS? *interactions, 2*(4). pp. 80-89.

John, B. E. (2009) *CogTool user guide version 1.1*. Carnegie Mellon University, Pittsburgh PA, September 17, 2009.

John, B. E., Prevas, K., Salvucci, D. D., & Koedinger, K. (2004). Predictive human performance modeling made easy. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI '04* (pp. 455-462). New York: ACM.

Lebiere, C., Gonzalez, C., Dutt, V., and Warwick W. (2009) Predicting cognitive performance in open-ended dynamic tasks a modeling comparison challenge. In A. Howes, D. Peebles, R. Cooper (Eds.), 9th International Conference on Cognitive Modeling – ICCM2009, Manchester, UK.

Nielsen, J. and Molich, R. 1990. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Empowering People* (Seattle, Washington, United States, April 01 - 05, 1990). J. C. Chew and J. Whiteside, Eds. CHI '90. ACM, New York, NY, 249-256.

Polson, P.G., Lewis, C., Rieman, J., and Wharton, C. (1992). Cognitive walkthroughs: A method for theory- based evaluation of user interfaces. *International Journal of Man-Machine Studies* 36, 741-773.

Author Biography

BONNIE E. JOHN (B. Eng. 1977, The Cooper Union; MS 1978, Stanford; PhD, 1988 Carnegie Mellon University), is a Professor, founding member of Carnegie Mellon University's Human-Computer Interaction (HCI) Institute, and a member of the ACM SIGCHI Academy. She has been researching human behavior modeling and using it to guide HCI design since 1983. As the Director of the Masters program in HCI, Dr. John has researched and taught many HCI design and evaluation techniques. She has brought these experiences together, through human-centered design and automating substantial portions of the modeling process, to create modeling tools that are easier to use, one of which (CogTool) is described in this paper.