Rapid Theory Prototyping: An Example of an Aviation Task

Bonnie E. John[1] Marilyn Hughes Blackmon[2] Peter G. Polson[2] Karl Fennell[2] Leonghwee Teo[1]
[1]Human-Computer Interaction Institute, Carnegie Mellon University
Pittsburgh PA 15213 USA
{bej, teo}@cs.cmu.edu
[2]Institute of Cognitive Science, University of Colorado
Boulder, CO 80309 USA
{blackmon, Peter.Polson}@colorado.edu, karlfennell@aol.com

We present our experience using CogTool, a tool originally designed for ease of use and learning by non-psychologist design practitioners, as a means for rapid theory exploration. We created seven iterations of a "model prototype" of an aviation task where each iteration produced errors that pointed to additional theory or device knowledge that should be incorporated to prevent those errors. This theory and knowledge was put into the next iteration by modifying the mock-up of the device, not by changing the implementation of the underlying cognitive model. This trick allowed us to rapidly change theory and knowledge and understand what must eventually migrate to the underlying cognitive model to provide general support for predictions of novice behavior in multi-step procedures with complex devices.

INTRODUCTION

Human performance models have been painstakingly constructed and validated by researchers in domains relevant to Human Factors for decades, but few practitioners routinely use modeling as a tool in their design and evaluation work. This has spurred several efforts to create tools that allow non-researchers to achieve the benefits of modeling without investing as much time in learning to model and constructing each new model (Blackmon. et. al., 2005; John, et. al., 2004). The time advantage of these tools is achieved by adopting sets of assumptions that simplify theories of human interaction. The fact that the target users of these tools are practitioners, and that such tools simplify theory, may lead researchers to dismiss the utility of such tools for basic research in new domains and for developing or extending theory. In this paper, we present an example from the aviation domain that belies that conjecture; we demonstrate how using a practitioner-oriented tool, CogTool, provides a platform for researchers to rapidly explore theory and knowledge requirements in a new domain.

BACKGROUND

CogTool was created to allow UI designers with a graphics design background to use the Keystroke-Level Model (KLM; Card, et. al., 1980). UI designers often express their designs in storyboards, with frames representing the states of the device, widget "hot spots" representing the interactive controls in the device, and transitions connecting widgets to frames representing the actions a user takes to accomplish a task. When a UI designer demonstrates the correct way to do tasks, CogTool turns these storyboards and demonstrations into ACT-R code (Anderson, et. al., 2004) that emulates the KLM, runs the code, and returns a prediction of skilled performance time for the task on that UI. This modeling-by-demonstration substantially reduced learning time and increased accuracy for novice modelers with no background in psychology, and showed an order of magnitude reduction in time for a skilled modeler to produce predictions on a new device and task compared to doing KLM by hand (John et. al., 2004). CogTool has been taught in HF and HCI classes at universities and professional conferences (e.g., HFES 2008, 2009) for several years and is used professionally by participants in these classes.

CogTool is now being expanded to make predictions beyond the KLM. Teo and John (2008) reported an integration of CogTool, an Information Foraging Theory model, SNIF-ACT (Fu & Pirolli, 2007), and a minimal model of visual search (Halverson & Hornof, 2007), that produced predictions of exploration behavior. This expanded tool, CogTool-Explorer, uses Latent Semantic Analysis (LSA; Landauer, et. al. 2007) to calculate semantic relatedness metrics to feed to SNIF-ACT. CogTool-Explorer uses the name of the task as the goal and the labels on the widgets in the storyboard as the terms to calculate semantic similarity.

Our goal in this research is to expand CogTool-Explorer beyond text-only, encyclopedia-like websites and simple search tasks to predict exploration behavior on complex devices with multi-step procedures and expert domain knowledge, for example, flight automation systems in an airliner cockpit. After presenting an aviation task, we report our expansion into this new domain, using CogTool-Explorer to rapidly explore what new theory, cognitive mechanisms, and domain knowledge may be necessary to include in a general model of flight deck tasks.

THE TASK AND DEVICE

The task is to enter the landing speed (an *approach reference speed*) into the Boeing 777 Flight Management Computer (FMC) using the Control and Display Unit (CDU). This task is very frequent and an experienced commercial airline 777 pilot instructor (the fourth author) considers it an easy task for commercial airline pilots new to the Boeing 777 FMC. (Note that these pilots have been commercial airline pilots for many years before entering 777 training, but flight automation devices like the CDU may be new to them.)

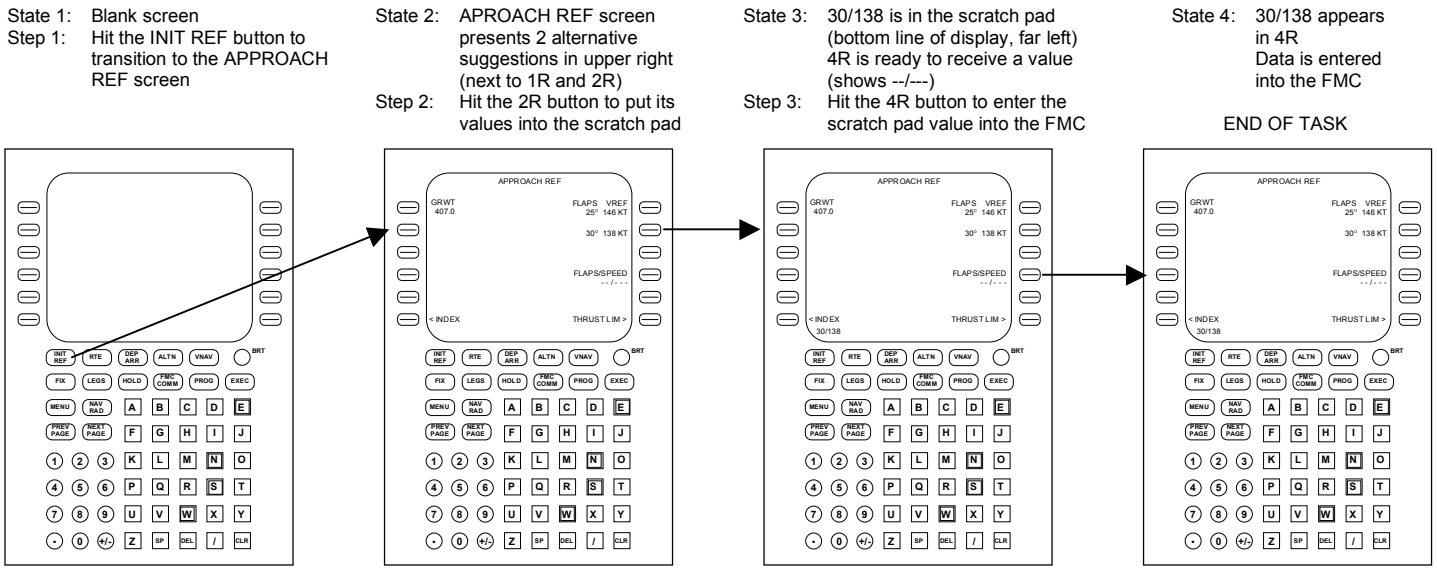

Figure 1. The steps to enter the approach reference speed using the CDU

The CDU is the primary input-output device for the FMC (Figure 1). Information in the FMC is organized into pages. A typical CDU data entry task is begun by pressing a function key to display the correct page (Step 1). A page display includes a title line at the top, right and left data fields, and a scratch pad line at the bottom. Six line select keys (LSK, referred to as 1R, 2R,…6R, 1L,…6L) are adjacent to the display on each side. These are "soft keys", i.e., their functions are indicated by the characters (commands or data and labels) on the display next to them. For example, in State 2, a pilot trained on this device knows that the approach reference speed can be entered into LSK 4R because it has the label FLAP/SPEED and a blank data display line (--/---). Hitting an LSK that contains a suggested value fills the scratch pad (Step 2), as does typing with the alphanumeric keys (not used here). The contents of the scratch pad are then entered into a data display line by hitting its LSK (Step 3).

ITERATIONS ON A MODEL PROTOTYPE

We performed seven iterations on a "model prototype" of this CDU task, that is, the ACT-R model that makes the predictions *did not change* between iterations, only the *semantic space it used* and the *storyboard it worked on* were changed. Thus, analogous to a UI prototype that works only enough to do user tests, our model prototypes work only enough to allow us to examine errors made at each stage and make changes to include new theory and knowledge to

	Success Metric				Changes in model prototype to simulate the changes in theory and knowledge & how long that change took to make		Types of Errors					
							Step 1	Step 2 - Hit 2R			Step 3 - Ht 4R	
Model Prototpye Iteration	**Total task success**	Success on Step 1 (Hit INIT REF)	Success on Step 2 (Hit 2R)	Success on Step 3 (Hit 4R)	Change in CogTool	Time to make change (minutes)	Incorrect region	Repeated INIT REF key	Hit screen label or blank button	Hit wrong alternative 1R	Repeated 2R key	Hit other alternative - 1R
1 Out-of-the-box CogTool-Explorer: Theory: Information Foraging Knowledge: College-level LSA	**0**	0					29					
2 Knowledge: Experienced pilots	**0**	10	0		Used CDU_skills corpus instead of 1st_year_College	1	86	6	3	0		
3 Theory: People elaborate Knowledge: Soft key labels are on the display	**0**	19	1	0	Elaborated goal & labels using flight manuals Put labels on the soft keys, not on the screen	180	70	15	-	1	1	0
4 Theories: Hierarchical visual regions + inhibition of return	**3**	99	60	3	Added intermediate frames with regions and paths for recovery that remove an incorrect region after selection	60	*16	0	-	35	41	15
5 Theory: Memory of last step Knowledge: Pressing buttons twice doesn't help this task	**4**	100	58	4	Removed button from frame after it was hit	1	*22	0	-	34	-	53
6 Knowledge: How alternative suggestions work	**4**	100	86	4	Added frames to visually check the alternative selection and to recover from an error	5	*25	0	-	*36	54	26
7 All of the above	**92**	100	95	92	Combined previous 2 changes	2	*14	0	-	*40	-	0

Table 1. Results of seven iterations of model prototypes of setting an approach reference speed on the CDU. The first column of results shows the number of model runs that successfully completed the task out of 100 runs. The next three columns show how many runs completed each of the three steps successfully. The changes made in CogTool and how much time it took to make those changes (in minutes) is in the center of the table. The right side of the table shows the types of errors at each step and how many runs made that error. Numbers with * preceding them indicate that the CogTool storyboard allowed recovery from these errors as per the theory. Cells containing a dash mean that the error is not applicable in this iteration. Blank cells mean the model did not get far enough to make that error.

alleviate these errors. Table 1 shows a progression from a model prototype that never completed the task to one with a 92% success rate. (There is variability in the underlying ACT-R model in its visual search and its decisions to choose between items of high semantic similarity, so we ran each model 100 times to assess its behavior.) Below, we present how each iteration differs from its predecessors and how new theory and knowledge were rapidly prototyped by simple changes in the storyboard.

Iteration 1: Out-of-the-box CogTool-Explorer

The Out-of-the-Box model was constructed by making CogTool storyboard of the information in Figure 1. The four states of the CDU are four frames in the storyboard. In each frame, 69 button widgets represent the CDU's buttons, labeled just as they are on the CDU, e.g., "INIT REF", "1", "A". Each line of text on the display screen is represented as a non-interactive widget (e.g., the model can look at it, but not touch it or transition to another frame from it). The goal was represented as a task, just as it was stated in a training document, "Select landing flap and reference air speed for an approach." The steps to accomplish this task are represented as transitions between button widgets and frames (as in Figure 1), i.e., hitting the INIT REF button in the first frame transitions to the second frame. If the model hits any other button, it does nothing and is counted as an error.

When CogTool-Explorer is run, it automatically sends the task statement and the widget labels to an LSA server to calculate semantic similarities between them, with reference to a default semantic space (1st_year_college). It uses those semantic similarities to make decisions about which widget to press to accomplish the task. We expected this out-of-the-box model to fail, as it knows virtually nothing about the vocabulary of the domain nor operation of the device, and, as expected, it failed in all 100 runs. In fact, it never succeeded at the first step, hitting the INIT REF button.

Iteration 2: Using experienced pilot's knowledge

The next iteration took only 1 minute to do with CogTool-Explorer because it already had the ability to select between different LSA semantic spaces. We simply selected (in a drop-down list) a new semantic space against which to calculate similarities between the task and widget labels. The new semantic space was built from 2192 documents, containing 4670 unique terms, extracted from four instruction manuals that pilots are likely to have read while learning to use the CDU. Calculating similarities against a domain-appropriate semantic space is analogous to using 777 pilot trainees in a user study instead of non-pilot college freshmen. This iteration did marginally better, selecting INIT REF in the first step 10 times, but those runs then failed to select 2R in the second step. This means that at the first step, INIT REF was more attractive than chance, but with 69 buttons to examine, 86% of the time it chose from an incorrect region of the device. Looking at the runs that made it to the second step, 6 of the 10 hit the INIT REF button again and 3 hit the characters on the screen instead of an LSK button. Both of these errors reflect lack of device knowledge, i.e., that repeatedly pressing a button doesn't make progress in this task and that the screen is not a touch screen but has soft keys that select the data displayed on the screen. The latter is remedied in the next iteration; the former in iteration 5.

Iteration 3: Elaboration & knowledge of soft keys

In the third iteration, we brought to bear prior research in semantic relatedness. First, measures of semantic relatedness in LSA and similar tools are more valid and reliable with longer texts than with short labels. Previous research has delivered accurate predictions of human behavior by automatically elaborating short labels, simulating how people elaborate text with background knowledge during text comprehension (Blackmon et al., 2005, 2007). Ultimately, this elaboration process could be built "under the hood" in CogTool-Explorer, but we could rapidly examine its potential by hand-editing the label on each button to include fully spelled out terms as well as abbreviations. For example, the label on the INIT REF button became "INIT REF Initialization Reference Position". Likewise, we elaborated the goal beyond what was originally written by the instructor; CogTool-Explorer allows an arbitrarily long string to be entered as the task name, so it was trivial to enter an elaborated goal into this field. The first two authors spent a few hours looking up terms in aviation glossaries and entering them into the button labels and task name.

We used a similar method to simulate the device knowledge that LSKs are labeled by the text on the display adjacent to them. The text on the line next to each button (and its elaboration) was simply copied into the label for each LSK in the storyboard. This simulated the pilot's knowledge that the label of an LSK is the text on the screen next to it.

These changes resulted in a model prototype that still does not complete the entire multi-step task. However, it hits INIT REF 19 times in the first step, and one run hit 2R in the second step (then hit 2R again in the third step, failing to complete the task). Hitting a key in the wrong region of the CDU on the first step is still a frequent error despite elaborations (70 runs); repeatedly hitting the same key on the second step emerges as a frequent error (made in 15 of the 19 runs that reached Step 2). The next iteration addresses the former, and iteration 5 addresses the latter.

Iteration 4: Prototyping hierarchical visual regions

Prior research, both empirical and theoretical, has indicated that a model of using a complex device should choose a region of the device to focus attention on rather than treating all the buttons as equally possible at all times (Blackmon et al., 2005, 2007). To simulate this theory in the storyboard, we inserted frames that had large widgets encompassing regions of buttons rather than individual button widgets (Figure 2, Frame A). The model was forced to select a region to focus on first, which transitioned it to a frame where it could see only the individual buttons in that region (Frame B). Following Blackmon et. al. (2005, 2007), the labels on these regions were the concatenation of all the button labels in

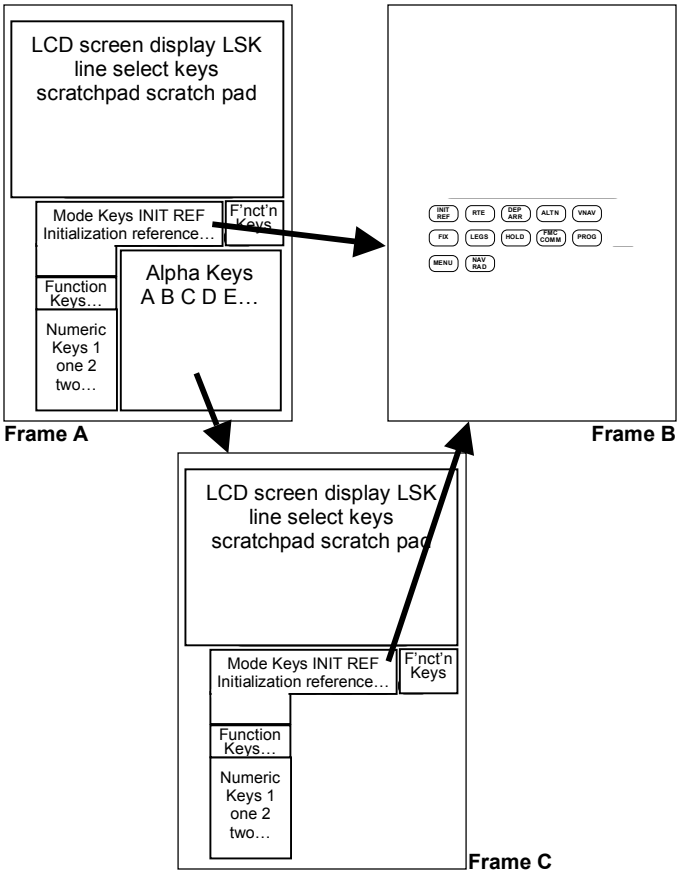

Figure 2. Portion of the prototype of hierarchical visual search.

the region. This easily prototyped a two-level visual search, focusing first on the region, (Mode keys, Function keys, etc.), and then on the specific buttons in that region

Because momentarily attending to an incorrect region would not prevent a pilot from completing the task, we augmented the storyboard to allow recovery from focusing on an incorrect region. Each incorrect selection transitioned to a frame that did not contain that region, forcing the model to focus on a different region (Frame C). If it chooses to focus on the correct region at this point, it transitions onto the correct path (Frame B) and continues with the task. Adding regions and the ability for the model prototype to recover from error took the first author 1 hour. By comparison adding a robust general mechanism for hierarchical visual search and recovery to the underlying ACT-R model has recently taken weeks of a skilled ACT-R modeler's time.

Prototyping hierarchical visual search caused big leaps in success rate on Steps 1 and 2, allowing 3 runs to successfully complete the entire task. Ninety-nine correctly completed the first step, with 16 runs recovering from initially focusing on an incorrect region, showing the value of the addition of these theories. Clearly additional knowledge is necessary to complete the entire task, but the augmented storyboard created in this iteration is a good foundation on which to explore which knowledge to add. Forty-one runs hit a key repeatedly and 50 runs hit 1R (the incorrect alternative for this task). Both errors reflect a lack of device knowledge, which we will explore separately in the next two iterations.

Iteration 5: Prototyping memory of the last action

The model repeatedly hitting the same button is understandable given the implementation of information foraging currently in CogTool-Explorer. If a button's label has a high semantic relatedness to the goal, the model hits it. A new screen appears, but the button's label is the same and the model sees it as still the most attractive button to hit even though the task has progressed to a different stage in the procedure. Therefore, the first bit of procedural knowledge we added was to not hit a key two times in a row. This is a reasonable piece of knowledge for this device and task, although it would not be for all devices (e.g., when scrolling down a contact list in a cell phone, the down button is hit repeatedly) or all CDU tasks (the CDU has Next Page and Previous Page buttons that are often hit repeatedly).

It is trivial to prototype this small theory (that the model has a memory of the last step) and device knowledge (that hitting a button twice doesn't help this task) in CogTool. Simply open the frame after a button is hit and delete that button. The button is no longer available for consideration at the new point in the task, prototyping that the model knows what it hit last and to not hit it again. This change to the storyboard took about 1 minute and increased the overall success rate by 1 run, from 3 to 4. Looking at the type of errors made, the last step had no repeated keys, but the model hit the alternative option as the last step in 53 runs. Thus, the next iteration adds device knowledge of alternatives.

Iteration 6: Prototyping knowledge of alternatives

When a pilot hits INIT REF, the flight automation uses the state of the airplane (Is it on the ground or in the air? Is it near its destination?) to suggest likely steps for the pilot to take. In this task, the Approach Ref CDU page displays two alternative flap settings and approach speeds to the pilot; 25º/146 KT (knots) on the top right line select key (1R) and 30º/138 KT on the second right line select key (2R). Pilots know that either option can be selected and that if the wrong one is inadvertently selected and appears in the scratch pad, they can select the other one by hitting the other button. This knowledge can be easily simulated by changing the CogTool storyboard (Figure 3).

In Frame E, the model can select either 1R (the incorrect alternative) or 2R (the correct alternative). It is then forced to attend to the scratch pad by transitioning to frames that have only the scratch pad available (simulating visually verifying its contents (Frames F or H). If the correct alternative was selected, the model can continue with the task (Frame G). If not, it must recover from its error by selecting the other alternative (Frame I), which puts it back on the correct path (Frame F).

This change in the storyboard took about 5 minutes and produced an overall success rate of 4 runs. The manipulation worked, in that 36 runs chose the incorrect option first but recovered from that choice, so 86 runs made it successfully past Step 2. In Step 3, however, 54 runs repeated 2R and 26 runs selected 1R, instead of proceeding with the task and entering the scratchpad into 4R. These errors are akin to the

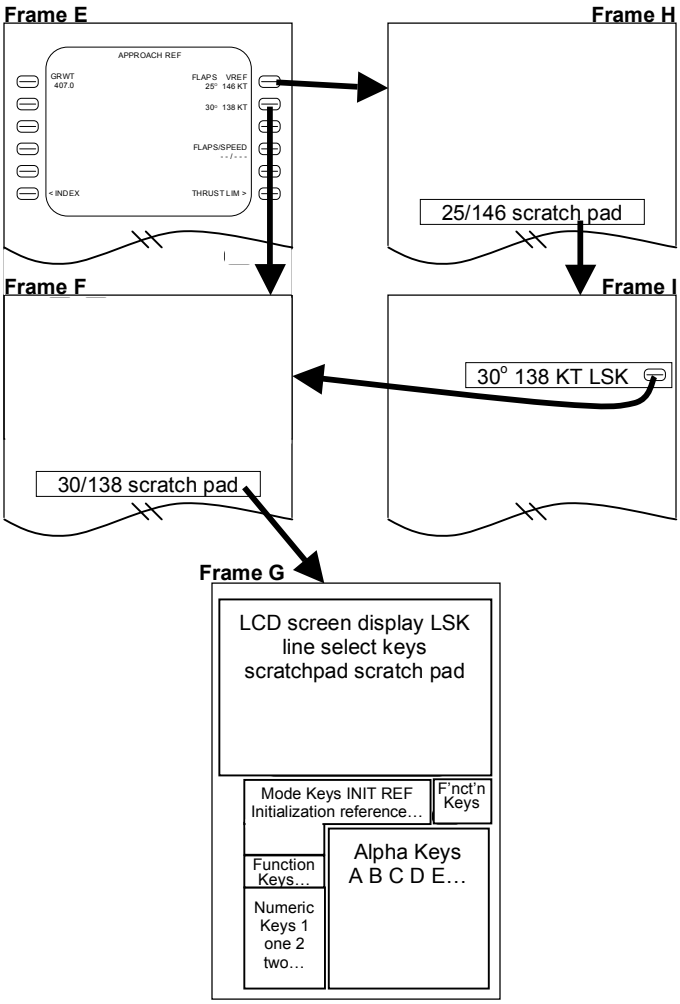

Figure 3. Portion of the prototype of knowledge of alternatives.

repeat-key error that was addressed in iteration 5. 2R is exactly the repeat-key error; 1R is not exactly the key that was hit in Step2, but it is involved in the step as an alternative to 2R. Thus, expanding the notion of memory for last step to include knowledge of alternatives will be the final iteration.

Iteration 7: All theories and knowledge combined

We added all theories and device knowledge together into one model prototype, a 2-minute change once the previous ones were built, and this final iteration performs the task with a 92% success rate. Fourteen percent initially examined the wrong region in Step 1 and 40% initially chose the wrong alternative in Step 2, but these are recoverable errors both in real life and in the model prototype. The 5 runs that did not succeed in Step 2 and the 3 runs that did not succeed in Step 3 can be attributed to the variability in visual search and decision-making inherent in CogTool-Explorer, but not to any systematic lack of knowledge.

DISCUSSION AND FUTURE WORK

Our exploration has demonstrated that a model with semantics alone is unlikely to accomplish multi-step tasks on a complex device. This in itself is not surprising, but the very low success rates for each additional piece of knowledge and the jump to 92% success rate when the low contributors are combined may be. This exploration itself may have implications for training, e.g., that experienced pilots new to the CDU should be instructed in how alternative suggestions are presented and can be manipulated. Perhaps more importantly, though, any model that hopes to predict performance on such tasks will have to include many pieces of detailed semantic, theoretical, procedural, and device knowledge to succeed, each of which would be substantial work to include in a robust, general way in an underlying cognitive model. So, before putting in that modeling effort, we have demonstrated a way to rapidly prototype the addition of those theories, mechanisms, and knowledge using CogTool-Explorer's storyboard. These prototypes provide guidance about what will be valuable to build into CogTool-Explorer "under the hood".

ACKNOWLEDGEMENTS

The authors thank Matt Koch, Steve Casner, and Mike Matessa for their contributions to the content of this paper and Mike Byrne and Wayne Gray for their comments on its presentation. This research was supported in part by funds from NASA, Boeing, NEC, PARC, ONR, N00014-03-1-0086, and a scholarship from DSO National Laboratories to the last author. 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 NASA, Boeing, NEC, PARC, DSO, ONR, or the U.S. Government.

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.

Blackmon, M. H., Kitajima, M., & Polson, P. G. (2005). Tool for accurately predicting website navigation problems, non-problems, problem severity, and effectiveness of repairs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '05 (31-40). New York: ACM.

Blackmon, M. H., Mandalia, D. R., Polson, P. G., & Kitajima, M. (2007). Automating usability evaluation: Cognitive Walkthrough for the Web puts LSA to work on real-world design problems. In T. K Landauer, D. S. McNamara, D. Simon, & W. Kintsch (Eds.), *Handbook of Latent Semantic Analysis* (pp. 345-375). Mahwah, NJ: Lawrence Erlbaum.

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.

Fu, W.-T., & Pirolli, P. (2007). SNIF-ACT: A cognitive model of user navigation on the World Wide Web. *Human-Computer Interaction, 22,* 355-412.

Halverson, T. & Hornof, A. J. (2007). A minimal model for predicting visual search in human-computer interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI '07* (pp. 431-434). New York: ACM.

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.

Landauer, T. K., McNamara, D. S., Dennis, S., & Kintsch, W. (2007) *Handbook of Latent Semantic Analysis*. Mahwah, New Jersey: Lawrence Erlbaum Associates.

Teo, L. & John, B. E. (2008) Towards predicting user interaction with CogTool-Explorer. *Proceedings of the Human Factors and Ergonomics Society 52nd Annual Meeting* (New York, NY, Sept 22-26, 2008).