Initial ACT-R Extensions for User Modeling in the Mobile Touchscreen Domain

Kristen K. Greene (kristen.greene@nist.gov)
National Institute of Standards and Technology (NIST), 100 Bureau Drive
Gaithersburg, MD 20899-8940 USA

Franklin P. Tamborello, II (frank.tamborello@cogscent.com)
Cogscent, LLC, 2711 Centerville Rd, Ste 120
Wilmington, DE 19808-1676 USA

Abstract

Advances in mobile touchscreen computing offer new opportunities to test traditional cognitive architectures and modeling tools in a novel task domain. ACT-Touch, an extension of the ACT-R 6 (Adaptive Control of Thought-Rational) cognitive architecture, seeks to update and expand methods for modeling touch and gesture in today's increasingly mobile computing environment. ACT-Touch adds new motor movement styles to the existing ACT-R architecture (such as tap, swipe, pinch, reverse-pinch and rotate gestures) and also includes a simulated multi-touch touchscreen device with which models may interact. An ACT-Touch model was constructed to explore the nature of human errors qualitatively observed during previously conducted formative usability testing, where participants occasionally missed taps on a particular interface button while completing a biometric sensor configuration task on a tablet computer. Due to features unique to the mobile touchscreen environment—finger size relative to target size—these objectively small errors in motor movement combined with interface usability issues to produce disproportionately large effects on cognition and task performance. This finding improved both the interface (practical application) and the model (theory).

Keywords: ACT-R 6; ACT-Touch; cognitive architectures; computational cognitive modeling; mobile handheld devices; motor execution errors; motor movement variability; tablet computers; touch and gesture; usability testing; user modeling.

Introduction

Just as technology continues to evolve, so too should our modeling and simulation techniques. As the use of mobile devices expands into historically desktop-bound application areas, the modeling community must be able to explain and predict how this rapidly evolving interaction paradigm (e.g., multi-touch gestural input on smaller displays) impacts human cognition and performance. The current work describes and makes use of recent motor extensions to the ACT-R cognitive architecture, ACT-Touch. ACT-Touch adds several basic motor movement styles (tap, swipe, pinch, reverse-pinch, and rotate gestures) that are commonly used across a variety of today's handheld mobile devices. In conjunction with ACT-Touch's included simulated multi-touch touchscreen device, these updated movement styles are a first step towards supporting higher-fidelity, longer-term model exploration in a task domain still fairly new in the computational cognitive modeling community, mobile touchscreens.

This is not to suggest that models using small devices (e.g., Das & Stuerzlinger, 2007; Luo & John, 2005) or touchscreens (Abdulin, 2011) do not exist, but rather to emphasize firstly that models of tasks on mobile multitouch devices are overall newer, fewer, and far less mature than models of tasks in the traditional desktop environment, and secondly that we wish to bring to bear mature tools such as the ACT-R cognitive architecture (Anderson, 2007) to Human-Computer Interaction problems involving mobile touchscreen computers.

There are particular human interface challenges for mobile touchscreen computers, such as the relatively small display area compared to desktop environments. Although smaller display sizes offer significant benefits in mobility, they also pose specific challenges for human cognition and motor performance. Reduced screen sizes may mean that task performance depends more upon human memory processes, since the user can view less information at a time. The use of fingers rather than mice as pointing/input devices has new implications for motor movement accuracy that are unique to the small touchscreen environment. With a traditional desktop computer, the pointer size (especially the very tip of the mouse cursor) is smaller than even the minutest radio buttons or checkboxes. In the handheld touchscreen computing environment (where stylus use is rare for the most common mobile devices), a person's finger can easily be larger than the target, and occludes a greater portion of the display than does a mouse pointer. Depending on the difference between fingertip size and touchscreen target size, this may make certain errors (e.g., missing a small target entirely or tapping a neighboring one instead) both more frequent and more costly to recover from when using a mobile device in comparison to a similar task with a desktop computer. Computational frameworks like ACT-R can help researchers address such human-computer interaction challenges by extending traditional modeling of perceptual, cognitive, and motor processes from the desktop to the mobile touchscreen environment; ACT-Touch and the current model are an initial attempt to do just that.

Architecture

While ACT-R's perceptual-motor modeling capabilities are well-developed overall, its basic motor movement styles were born of the traditional desktop research environment (PC, monitor, physical keyboard, mouse) and may benefit from updating to better reflect the use of modern touch screen devices. To begin addressing modeling challenges in the mobile touchscreen task environment (such as simulating smaller displays, virtual keyboards, and direct manipulation of interface elements), ACT-Touch adds new

motor movement styles to the existing ACT-R architecture, such as tap, swipe, pinch, reverse-pinch and rotate gestures.

ACT-Touch is a novel tool for addressing issues of Human-Computer Interaction in mobile touchscreen task environments in combination with the theoretical broadness and rigor of a cognitive architecture, namely ACT-R. For now, ACT-Touch's nascent state precludes address of many low-level details, such as accounting for how much the area of a finger increases as it compresses against the touchscreen surface. Another limitation is that when ACT-Touch touches an interface element within its included simulated touchscreen device, it always touches the upper-left corner of that element. This has to do with low-level details underlying ACT-R's motor module and can be addressed with further technical development.

ACT-Touch: Motor Extensions for Touch HCI

ACT-Touch extends ACT-R's motor feature preparation and execution framework by adding additional movement styles applicable to the mobile touchscreen environment to ACT-R's extant motor module. In ACT-R parlance, ACT-Touch extends ACT-R's motor module manual requests to include new requests to the standard ACT-R manual request repertoire, such as tap:

```
+manual>
        isa swipe
        hand right
        finger index
        r 200
        theta 0
        num-fngrs 2
```

The movement feature and preparation framework, borrowed from the EPIC (Executive Process-Interactive Control) cognitive architecture (Kieras & Meyer, 1997), extends readily to a new task environment because the same features used to perform movements such as typing and mouse pointing are readily recomposed into movements appropriate to the mobile multi-touch screen task environment such as taps and swipes.

In the swipe movement type example above, standard EPIC-derived features present in ACT-R are used to emulate a swipe style movement across the face of the simulated touchscreen. A swipe consists of pressing *num-fngrs* number of fingers onto the surface of the touchscreen, moving them a distance of r pixels in the direction of θ radians (where 0 radians is to the right), and then lifting the fingers off of the surface of the touchscreen device.

In ACT-R as in EPIC, motor movements occur in phases. First there is preparation, in which ACT-R collects the features of the movement such as which finger and which hand will perform the movement, as well as action type (e.g., ply), and direction and distance as applicable. Preparation time increases with increasing number of movement features.

Following the preparation phase, ACT-R executes the movement. The execution phase simulates the model's hands or fingers physically performing the action in the model's simulated physical space. Finally, the movement may have a finish phase in which the model moves its finger or hand back to its starting position, if applicable.

ACT-Touch follows the same sequential pattern as ACT-R in constructing motor movements, but it composes the motor movement features provided by ACT-R into new gestural commands appropriate to touch screen computer task environments. Input commands such as swipe, scroll, flick, pinch, are now common in the mobile domain, yet simply did not exist in the traditional desktop research environment.

Virtual Multitouch Device

ACT-Touch includes a simulated multi-touch touchscreen device with which ACT-R models using the ACT-Touch manual request extensions may interact during model runs. The *virtual-multitouch-device* is an ACT-R device capable of presenting visual and aural stimuli to ACT-R and taking input from model actions. The virtual-multitiouch-device interprets ACT-R motor events of the types supplied by the ACT-Touch manual request extensions as touchscreen events, i.e. taps, swipes, etc. Furthermore, ACT-Touch's virtual-multitouch-device includes library code for generating and running experiments with blocks of trials or as continuous sequences of events, as in a lengthy procedural task. The library code is also instrumented so that it can record information such as action types and latencies. It is based on experiment management library code developed by Dr. Michael D. Byrne at Rice University. The ACT-Touch software library and accompanying reference manual are available for download as source code from http://www.cogscent.com/.

Model

Model Purpose

The current model was constructed to explore the nature of a specific human error qualitatively observed during prior formative usability testing of biometric sensor configuration on a tablet computer (Greene, Fiumara, & Micheals, 2013). During several user testing sessions, experimenters observed that participants occasionally failed to tap the *Done* button as they should have when they finished configuring sensor properties (Figure 1).

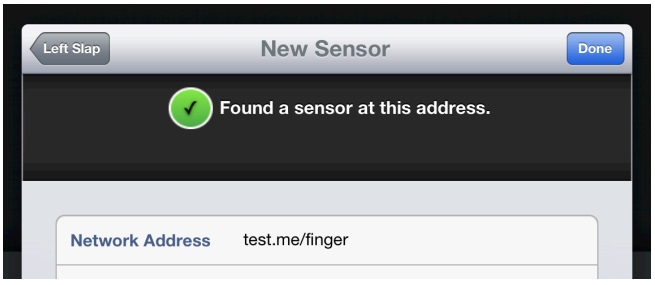

Figure 1: Partial screenshot of sensor properties dialogue.

When misses occurred, it appeared that participants tapped very near the *Done* button at that point, which would indicate that it may have been their intended target. Based on experimenters' qualitative visual observations in conjunction with participant comments, it seemed highly likely that these errors were motoric rather than cognitive in origin. Participants in the original formative usability testing

sessions were novice users (intentionally given no training and recruited specifically to have little knowledge of biometrics); if expert users with training and biometric knowledge were also to miss tapping the *Done* button, it would lend greater support to the claim that motor execution errors rather than lack of task knowledge caused participants to miss the *Done* button. Therefore, we constructed an ACT-Touch model to explore the *Done* button error and make initial predictions regarding that specific motor error for expert users. Because this model was based on an infrequent error qualitatively observed during prior formative usability testing, we did not start out with quantitative error data to initially inform the model. In order to begin collecting the quantitative data needed for future model validation, we incorporated custom touch-logging within the application, then piloted the task with two expert users in order to compare their data with ACT-Touch model predictions regarding expected frequencies and tap locations for *Done* button misses. The current ACT-Touch model was intended solely to further explore that very specific motor movement error of missing the *Done* button. Why is a simple button miss like this worth modeling? 1) Because of the high frequency with which buttons like it appear across other, more common mobile applications, and 2) Due to the abnormally high cost of error recovery in this particular task, as described below.

Sensor Configuration Task

Sensor configuration consists of a series of subtasks during which the user taps various menu buttons to select the desired biometric modality and submodality[1] (finger and left slap, respectively), then enters a compatible sensor's network address and name via the native iOS virtual keyboard (Figure 2). For the current model, we focused solely on the sequence of menu button taps and ignore the typing subtask, as modeling text entry with virtual keyboards was out of scope for the project. We do not address the initial knowledge acquisition or transfer process here. Our focus is intentionally limited to modeling behavioral data from *experienced* operators configuring a single sensor repeatedly.

To configure a sensor, one must step through sequentially ordered subtasks, essentially advancing through different screens for selecting biometric modality, submodality, and sensor settings, in that order. If the error occurred in the first two subtasks, when participants were selecting the desired biometric modality (finger capture type) and submodality (left slap), participants erred by accidentally tapping the menu button immediately below the target. In both subtasks the target was the topmost menu button. While interesting, these errors were relatively minor in their impact on task performance; they required only two corrective tap actions, one to navigate back to the previous step, and a second to select the correct menu option.

In sharp contrast to the minor consequences of motor execution errors during the modality/submodality subtasks, consequences of error commission in the final sensor settings subtask were much more severe. Whereas the former only required re-execution of the immediately preceding subtask (in two quick taps), the latter required re-execution of all the preceding sensor configuration subtasks. The latter errors occurred when participants accidentally tapped just outside of the *Done* button after entering sensor network address and name. (As noted previously, we do not attempt to model typing errors—of which there were several—during the sensor information entry subtasks.) Note that the *Done* button is initially gray and inactive (Figure 2) until the system checks for a sensor at the specified network address; after this check is completed, the *Done* button changes to blue and becomes active (Figure 1). This is where the critical, yet infrequent, *Done* button misses occurred.

The blue *Done* button (along with the similarly sized edit/cancel buttons) is a commonly used native iOS[2] control, and often missing the *Done* button in many applications has no effect other than forcing the user to try again; in these instances error recovery usually consists of a single tap as users aim for the same button again. Unfortunately, in the current application, the error recovery process was much more costly, since tapping the *Done* button was a crucial final step in the last subtask; tapping it was the last action required to exit the sensor configuration task sequence and save all modality, submodality, and sensor information settings from preceding subtasks. A tap that missed the *Done* button was registered by the system as a "tap outside to dismiss" command, to which the system dutifully responded by dismissing the current sensor settings screen—along with "dismissing" all the information entered in the preceding configuration screens—and returning to the main WSABI screen. Some users did not realize that this had occurred (after all, shouldn't tapping the *Done* button also dismiss the screen?). For those who did realize it had occurred, they had to repeat the entire left slap configuration procedure; none of the information previously entered was saved. Clearly a significant usability issue (subsequently fixed), the tap-outside-to-dismiss feature had actually been implemented by request, to make it easier to jump out of the sensor configuration workflow at any time. If modeling been used to determine whether to implement the dismiss feature in the first place, we may have been able to predict the observed motor movement errors *a priori* with a more complete and validated model.

[1] The most commonly used biometric modalities are finger, face, and iris. Examples of less commonly used biometric modalities include voice, gait, vein, and DNA.

[2] Disclaimer: Any mention of commercial products or reference to commercial organizations is for information only; it does not imply recommendation or endorsement by the National Institute of Standards and Technology nor does it imply that the products mentioned are necessarily the best available for the purpose.

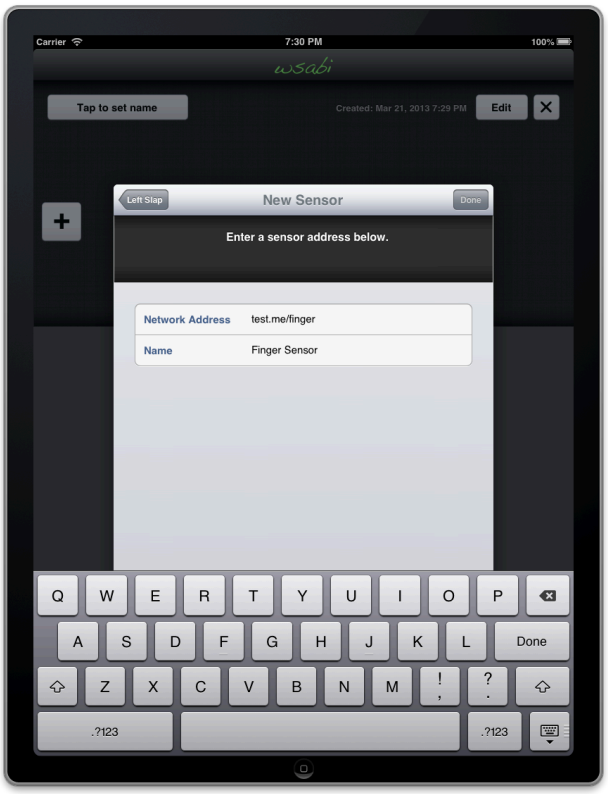

Figure 2: Full screenshot of sensor properties dialogue. (*Done* button inactive).

Modeling Missed Touchscreen Button Taps

The ACT-Touch sensor configuration model uses only two of the several new motor actions that ACT-Touch adds to ACT-R: tap and move-hand-touch. The former models the user moving the finger from just above the simulated touch-screen surface to contact the surface briefly, then return to its starting position retracted from the display device. The latter movement type, move-hand-touch, moves the hand from its current position to that of a location or object that the ACT-R model sees.

Human performance at the *Done* step in the sensor configuration task revealed a new constraint on motor movements in the touch screen task environment. The small size of the *Done* button relative to a human fingertip, combined with the button's close proximity the edge of the dialog window, meant than even when subjects were aiming to tap the *Done* button, they occasionally missed it due to physical constraints of the finger-to-target size ratio. The pattern of taps suggests that the errors were motoric in origin.

Unlike all the other simulated steps, which use tappable target regions measuring 950 pixels wide by 90 pixels tall, the *Done* step uses a tappable target region measuring only 100 pixels wide by 60 pixels tall. At the display resolution used by the iPads in data collection the area of the *Done* button is smaller than that of a typical adult's index finger tip.

The apparent difficulty subjects had in tapping the *Done* button carried with it an important implication for task performance. When subjects missed the *Done* button and instead tapped at a location not only outside the *Done* button, but also outside the sensor configuration dialog window, the application cancelled the sensor configuration and returned to its main menu.

Tapping the *Done* button happened to be the final step of the sensor configuration task, and it was located in the upper-right corner of the sensor configuration dialog window. The dialog used only a portion of the iPad's display in the middle of the screen so missing the *Done* button often resulted in tapping outside of the sensor configuration dialog window. Because of the "tap outside to dismiss" capability, taps that fell outside of the sensor dialog cancelled the sensor configuration, forcing subjects to unnecessarily repeat their previous steps in order to recover from the missed tap error.

The sensor configuration model introduced to ACT-Touch's move-hand-touch command a method to model subject performance with small interface items such as the *Done* button (i.e., taps outside of, yet fairly close to, the target). The revised move-hand-touch calculates the area of the movement's target and compares it to the area of the model's simulated index finger tip, which ACT-Touch defaults to 45 pixels wide by 27 pixels tall, or 5/8 inches by 1/4 inch at 72 pixels per inch. If the target area is less than the model's index finger tip area, then with probability proportional to the size of the difference, the move-hand-touch returns to the ACT-R simulation a tapped location that is outside the requested target. When ACT-Touch determines that it has missed the target, it then computes the distance by which it has missed the origin (of the virtual-multitouch-display-device) and adds that distance to the dimensions of the missed widget in order to return the miss tap location to the ACT-R simulation. The amount of deviation is also proportional to the size of the difference in areas.

ACT-Touch calculates the probability of a miss by first taking the $\log_2$ of the difference of the index finger tip area (normed to the second author's right index finger tip) and target area. If that quantity is greater than a randomly chosen integer out of {1:100}, then, at random, ACT-Touch would either add or subtract that $\log_2$ areal difference to the sum of the target's center and half its width and height, respectively, to produce X and Y coordinates to where the model's fingertip would actually move.

Model Results and Discussion

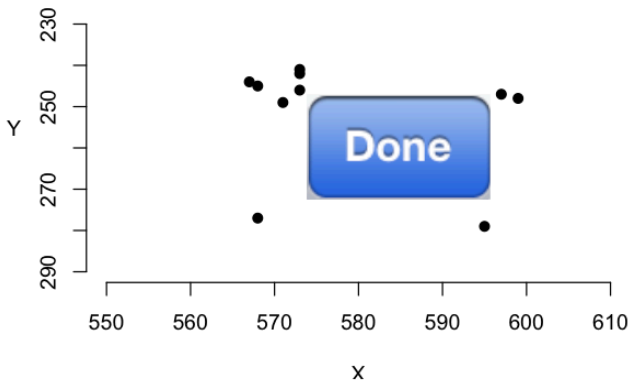

Figure 3: Scatterplot of model tap actions (misses) around the Done button of the sensor configuration task.

The model predicted missing the *Done* button 10% of the time (10 out of 100 trials); these 10 misses around the *Done* button are depicted in Figure 3. The remaining 90 out of a 100 trials were hits (i.e., accurate taps on the *Done* button). Hits are not included in the scatterplot, as there was no variability in the distribution of hit touch coordinates; for every hit, the model tapped the exact same location in the upper left corner of the *Done* button (575, 250, corresponding to the virtual-multi-touch-display device's origin). The lack of variability in hit locations is due to a limitation in the current ACT-Touch implementation: it only touches the upper-left corner of interface widgets. Based on the literature, this is clearly not representative of the pattern of variability seen in human touch distributions. In fact, merely observing the two pilot participants (discussed in the next section) was sufficient to emphasize the importance of addressing this limitation in future work. ACT-Touch should make move-hand-touch optionally noisy, to make ACT-Touch's movements distributed appropriately in the location of the target most commonly tapped by humans. This may be a way to make small target misses fall more naturally out of the theory.

Collecting Data for Model Evaluation

To move from the previously described qualitative observations to the quantitative data collection needed for ACT-Touch model validation, we updated existing customized iOS touch-logging (Greene, Tamborello, & Micheals, 2013). We now have some capability to record an integrated, timestamped log of user touch events and corresponding system responses. For applicable events, the log contains a description of the item tapped, the event type (e.g., tap, scroll), the local touch coordinates, the global touch coordinates, the dimensions of the tapped object, and the top-left coordinate of the tapped object. ACT-Touch currently outputs global touch coordinates only; object dimensions and locations are located within its virtual-multitouch-display device, but are not automatically included in the model output.

During informal pilot testing of these touch-logging capabilities (N=2, experienced users who each performed 100 repetitions of the sensor configuration task previous described), we identified a significant logging issue: because the critical *Done* button is an iOS-provided control, along with the navigation bar upon which it rests, logging XY coordinates for touches on those items will require implementing additional custom code. While we unfortunately do not have fine-grained quantitative data (i.e., XY tap coordinates for the error type of interest) from our pilot testing, we do have basic counts of the number of *Done* button misses: eight misses for one participant, and only a single miss for the other. We also observed a new type of motor error, one that would not have been predicted with the current ACT-Touch model: accidental touch input. Accidental touch input was a repeated problem for one user, but it was unclear whether this was due to motor fatigue vs. continual switching between tapping with the index and middle fingers (or some other factor not observed). While ACT-Touch already provides support for modeling multi-touch input, such input was assumed to be intentional, the result of a goal-directed action.

Future Modeling Directions

In addition to the unnatural consistency in hit locations, the current model was limited to simulating experienced users who already possessed knowledge of task steps and their corresponding screen locations. In the future, a more complex model would ideally address the knowledge acquisition process for truly novice users. Issues of motor learning (including along the z-axis), motor fatigue, dexterity limitations, handedness, and hand size may also prove promising avenues for future model expansion.

Practical Applications

The use of handheld mobile devices for biometric configuration and capture is relatively new in the biometrics community. Biometrics (information about a person, such as fingerprints, face images, and iris images) are used by both government and industry for a variety of applications such as screening, border control, physical access control, and enrollment. Currently, biometric sensors (hardware devices that capture the biometric data) are frequently constrained to the traditional desktop computing environment, with separate systems requiring proprietary software that is vendor- and device-specific. Operators need substantial training on each piece of software, and switching vendors or adding new devices can mean significant user retraining costs. NIST's Biometric Web Services (BWS) project (http://bws.nist.gov) is developing technical specifications for biometric sensors to use Web Services[3], enabling interoperability and mobility across devices and platforms.

The BWS reference implementation uses a handheld touchscreen computer to wirelessly control multiple biometric devices using a single application, where sensor configuration is generalized to work similarly across different biometric modalities and devices. Unlike many existing systems, operators do not have to switch between different software programs—*the operator performs the same basic sequence of task steps regardless of biometric modality[4] or sensor.* One would expect significant human performance benefits from this type of cognitive and procedural consistency. However, testing large numbers of trained FBI and DHS operators to explore such benefits is simply not feasible.

As in other areas where access to large numbers of subject matter experts is difficult (e.g., pilots, air traffic controllers, astronauts, submarine crew), computational cognitive modeling can augment existing research efforts. In this case,

[3] Web Services, as defined by the Internet standards body W3C (World Wide Web Consortium), refers to "a software system designed to support interoperable machine-to-machine interaction over a network." It uses machine-processable formats such as WSDL (Web Services Description Language), SOAP (Simple Object Access Protocol), HTTP (Hypertext Transfer Protocol), XML (Extensible Markup Language), and REST (Representational State Transfer).

cognitive modeling can help supplement costly and time-consuming usability testing in the BWS project to better demonstrate and predict the total human-system performance for different biometric configuration tasks. The current model is a small first step in ongoing efforts to optimize the interface and task structure, and most importantly, objectively quantify operator-training/re-training savings.

Conclusions and General Discussion

We discovered an interesting cognitive side effect arising from occasional, small inaccuracies in human motor movement. Simply missing a target button by a few pixels could cancel a current operation and require redoing several preceding steps, interrupting a well-practiced sequence of taps and forcing subjects to recognize and correct an error.

Issues modeling human-computer interaction in the particular domain of mobile devices are addressable by many of the same basic research methodologies used to date for traditional desktop HCI problems. Human performance in the mobile touchscreen domain can still be measured with traditional metrics of task completion time and errors. Those two things are a function of cognition, which itself is the mental transformation of information. That information is taken from the environment, encoded, operated on, and transmitted back to the environment through action. Furthermore, many mobile computing platforms adopt representational conventions pioneered in the desktop graphical user interface milieu, such as using icons to represent applications, files, and folders, and interactive form widgets such as checkboxes, radio buttons, and text fields. Thus perceptual, cognitive, and motor information processing paradigms developed for HCI research in the desktop domain tend to be fundamentally applicable to the mobile touchscreen domain as well. While cognitive architectures and modeling tools may require specific modifications to better address user modeling in this novel domain, the core paradigm of comparing model predictions with human data to advance a unified theory of human cognition remains unchanged.

Acknowledgments

This work is funded in part by a Measurement Science and Engineering grant from the National Institute of Standards and Technology's Information Technology Laboratory (ITL) Grant Program. Federal Funding Opportunity 2012-NIST-MSE-01. Grant 60NANB12D134, "Formal Model of Human-System Performance," awarded to Cogscent, LLC. Special thanks to Dr. Ross M. Micheals (NIST) for project guidance, and to Dr. Michael D. Byrne (Rice University) for the use of his code library.

References

Abdulin, E. (2011). Using the Keystroke-Level Model for Designing User Interface on Middle-Sized Touch Screens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI 2011*. ACM, New York, NY.

Anderson, J. R. (2007). *How can the human mind exist in the physical universe?* New York: Oxford University Press.

Biometric Web Services Project. http://bws.nist.gov

Byrne, M. D. http://chil.rice.edu/projects/RPM/index.html

Cogscent, LLC. *ACT-Touch.* http://www.cogscent.com/

Das, A., & Stuerzlinger, W. (2007). A cognitive simulation model for novice text entry on cell phone keypads. In *Proceedings of the European Conference on Cognitive Ergonomics: ECCE 2007* (pp. 141-147). London, UK.

Greene, K. K., Fiumara, G., & Micheals, R. J. (2013). Design and Testing of a Touchscreen Interface for Multi-Modal Biometric Capture. (Manuscript in preparation).

Greene, K. K., Tamborello, F. P., & Micheals, R. J. (2013). Computational Cognitive Modeling of Touch and Gesture on Mobile Multitouch Devices: Applications and Challenges for Existing Theory. To appear in *Proceedings of the 15th International Conference on Human-Computer Interaction.* Las Vegas, NV.

Kieras, D. E., & Meyer, D. E. (1997). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction*, 12(4), 391-438.

Luo, L. & John, B. E. (2005). Predicting task execution time on handheld devices using the keystroke-level model. In *Extended Abstracts of CHI 2005*. ACM, New York, NY.