ACT-Droid Meets ACT-Touch: Modelling Differences in Swiping Behavior with Real Apps

Nele Russwinkel (nele.russwinkel@tu-berlin)

Sabine Prezenski (sabine.prezenski@tu-berlin.de)

Lisa Dörr (lisa-madeleine.m.doerr@campus.tu-berlin.de)

Department of Cognitive Modeling in dynamic Human-Machine Systems, TU Berlin,

Marchstr. 23, 10587 Berlin, Germany

Frank Tamborello (frank.tamborello@cogscent.com)

Cogscent, LLC, Houston, Texas, USA

Abstract

This paper presents a tool (ACT-Droid) that integrates user models with mobile devices and enables modelling time-sensitive touch interactions via ACT-Touch. It demonstrates that using ACT-Droid in combination with ACT-Touch is a step towards model-based user studies of real smartphone apps. A special focus is set on modeling swipe interactions. Hereby, different interaction strategies depending on pre-knowledge of the interface type (e.g. sorted or unsorted lists) are modeled.

Results of an empirical study conducted with a real smartphone app indicate that searching through lists results in different kinds of swipe behavior for sorted vs. unsorted lists. Sorted list interfaces elicited fewer and quicker swipes than unsorted lists. This paper introduces ACT-R model approaches capable of capturing such behavior.

Finally, the importance of understanding top-down strategies involved in such regular tasks is discussed.

Keywords: Applied research, ACT-R, User Modelling, item search, swipes, Touch interaction, tool.

Applied Cognitive Modelling

To test the validity of theoretical approaches in real word situations, applied tasks are needed. The question is, whether developed theoretical concepts also apply for realistic, less-controlled tasks. Already experiments outside the modeling community are conducted on mobile devices (e.g. tablets, smartphones) or on online platforms (e.g. mechanical Turk), since this is often more convenient. Modelling such tasks often ignores the kind of interaction.

Not only different devices, but also different interaction modalities (such as swiping, steering, tapping) not only influence how we interact with devices but also how we plan

Amazon Mechanical Turk:

https://www.mturk.com/

MTurk is a crowdsourcing marketplace that makes it easier for individuals and businesses to outsource their processes and jobs to a distributed workforce who can perform these tasks virtually.

and represent tasks internally and how we judge an interaction as successful, since the feedback could be different.

Another interesting question is: How do we adapt our behavior with specific devices we use every day? Over time we develop strategies to be more efficient in our tasks. These strategies help users to achieve easier and more effortless interactions. In this way, interfaces that allow such effortless interaction will be judged as being of high usability. More so, if users find efficient strategies for device interaction this changes the perceived usability of a device. For example, Burnett et al (2013) showed that the speed and length of touch screen interaction differs if users are preoccupied with another task (e.g. driving) then when they are focusing on just interacting with a touchscreen. The authors speculate that observed slower swiping behavior during driving is due to reduced resources available in such task. A cognitive modeling approach has the potential to shed light on such claims.

Gray & Boehm-Davis (2000) identified that people are sensitive to the costs of low-level processes in their interactive task behavior and adapt their behavior accordantly. Especially for reoccurring strategies in low level steps of interactive behavior, these strategies quickly add up and produce relevant time differences that user models should be able to account for.

However, to address these topics appropriate tools and approaches are needed. These should shed light on understanding what factors, strategies and task demands influence interaction with mobile devices. To support research in this area we will present cognitive modeling approaches for ACT-R.

In this paper we present the newest version of **ACT-Droid** (Dörr, Prezenski & Russwinkel, 2016), a tool that allows a cognitive model to connect with an application on a mobile device. ACT-Droid can be used with any open-source Android application. It allows ACT-R models to directly interact with the interface of a mobile device. A new feature is that ACT-Touch commands are now integrated in ACT-Droid. Thus, directly modeling long tap, touch, swipe and other kinds of interactions on the actual Android application is now possible.

We want to show that our modeling approach allows the capture of user behavior occurring in natural tasks such as efficiently finding and selecting items. Therefore, the differences in search time, and behavior evolving due to the use of different strategies are examined in an empirical study. A modeling approach which integrates these different swipe interactions is presented.

ACT-Touch Introduction

ACT-R includes a library of HCI routines. However, these routines assume a traditional desktop computing environment. Hereby, the modelled user views an upright, freestanding computer monitor while holding hands upon a keyboard's home row and occasionally reaching for a one-button mouse (Bothell, 2017). However, since the iPhone popularized the handheld, multi-

touchscreen mobile computer starting in 2007, Apple has produced 1.2 billion iPhones and Android has more than two billion monthly active users. Mobile computing has eclipsed desktop computing and HCI modeling tools must address this reality.

ACT-Touch (Tamborello, 2018) is an add-on to the ACTR software distribution. It provides a library of additional motor commands for the model to execute in the condition of a simulated multi-touch display computing environment. For example, contrasting with common ACT-R's default hand position, ACT-Touch assumes both hands are positioned relative to the display, rather than the keyboard, on a coordinate system corresponding to the pixels of the display, floating approximately 1 inch above the display surface. ACT-Touch enables ACT-R's hands to move around the display and to perform many of the gestures familiar to users of mobile multi-touchscreen devices, such as taps, swipes, and pinches. While some of these motor commands, such as tap, are essentially repackaged extant ACT-R commands renamed to reflect the multi-touchscreen environment, other commands, such as swipe, depart further from ACT-R's desktop environment. [Some motor commands are still as same as common ACT-R]

ACT-R movements use features, such as hand, finger, distance, and direction, to describe movement. Models compose these features into a movement and ACT-R calculates movement preparation, execution, and finish time from those features.

1) **Tap** is peck renamed, including hand, finger, distance, and direction.
2) **Swipe**, on the other hand, includes two additional features: number of fingers and speed. Number of fingers is important to the swipe gesture because some touch interfaces, such as iOS on the iPad, interpret the varying numbers of fingers in the swipe gesture as unique commands, to, for example, scroll a document or switch to iOS' multitasking display (Patterson, 2013).

The project reported herein implemented a new speed feature for the swipe motor command to allow models to control swipe speed. This is important because, e.g. iOS scrolls more of a document with each pixel of the display traversed by the fingers the faster the fingers move, accelerating scrolling behavior in response to accelerated user input.

ACT-Droid Tool

In the future it may be possible to use cognitive models instead of user testing (see Prezenski & Russwinkel, 2014). However, studies with human participants are necessary for most user-study related questions and thus, also real human computer interfaces. In a smartphone setting these are either Android or iOS applications. Thus, until recently an ACT-R modeler interested in modelling user-behavior had to translate the entire GUI into Lisp.

ACT-R's GUI, the AGI, is limited in several ways. For example, there are only three different types of objects that can be created for interactions: text, line and button. These objects have

limitations themselves, as one cannot create a button with background color and font color.

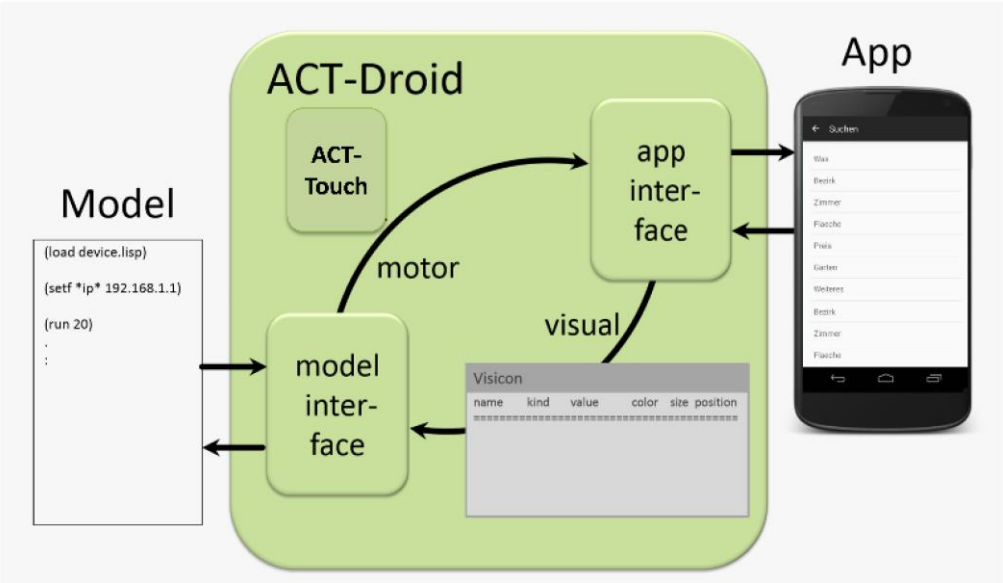

Figure 1: Structure of ACT-Droid

To avoid such restrictions and to realize direct interactions between ACT-R models and applications we developed the tool, ACT-Droid. This tool connects an ACT-R model with an Android app, thus enabling the model to execute its motor commands on the app and the app to fill ACT-R's visicon with the visible objects it displays. The first version of ACT-Droid could not cope with touch interactions.

In my study, it's important to define Touch interactions. (tap)
In my application, both of choose items and search items are through tap interactions.

Now, ACT-Droid performs the motor output via ACT-Touch commands (e.g. swiping) on the app and subsequently updates the visicon (if the app screen changes). These functionalities are provided by the model interface and the app interface, which communicate with each other (see figure1).

For our study an example real-estate app was installed on a smartphone. ACT-R communicates with the app over TCP/IP sockets. Hereby, the app interface establishes a server socket and the model interface connects ACT-R as a client.

ACT-Touch commands are used for motor output. So, each time the model's finger is moved, the model interface sends the new position to the app interface. The app interface saves the current position of the finger. Furthermore, if the command to tap, swipe etc. is received, the app interface performs it at the saved cursor position.

Matsumuro-sensei also commented that [When we use such type of interfaces, information about the position and features of the icon we have to touch is important. Even if we have only link information at the start of the task, over the trials, we acquire such information, and the icon selection, or tapping the icon, becomes more unconscious (not need to retrieve the memories).] (JCSS2020)

The app interface provides already a basic description of all visible information and it can be further adjusted to different apps. Hence, higher fidelity modeling of user behavior within a mobile context should be possible with ACT-Droid. Thus, it is now possible to write ACT-R models that capture slow or quick swipes occurring while using a real app.

In this paper a study is reported that investigates different scrolling behavior. This data is used to develop a modeling approach. This study can be regarded as a proof of concept to show the usefulness of this tool to test theories in applied settings. Since the focus of the paper is set on the tool only the most relevant data is presented.

Modelling Approach for Swipe Interactions

Swipe Mechanics

First, the Lisp environment loads ACT-R and then ACT-Touch. After this the motor commands ACT-Touch provides are available to any ACT-R model just as common ACT-R motor commands are. Within the manual request movement type other request slot values indicating the movement features are defined.

For example, to swipe with the right index finger rightward with two fingers for 100 pixels at a moderate speed, the production's righthand side should indicate:

```
+manual>
cmd swipe
hand right
finger index
r 100
theta 0
num-fngrs 2
speed 3
```

ACT-R uses Fitts Law to compute movement execution times. ACT-Touch's swipe execution time function calls ACT-R's Fitts law function with coefficient of difficulty inversely squared to the speed feature the model supplies to the swipe command. These are execution time, as well as a width parameter set to the width of the display, 500 pixels in the case of the example model. See the compute-exec-time method specified for the swipe class in act-touch.lisp, where it calls ACT-R's fits function with the motor module.

Model: Swipe in sorted and unsorted lists

In the following, we will introduce a simple, errorless performance model that uses pre-knowledge of whether a long list of items is sorted alphabetically or randomly to decide its interaction strategy. The model will either swipe with a fast speed to scroll the interface quickly to the end, where it “thinks” its target is positioned, or it will swipe slowly to scroll the interface

slowly as it searches the list for its target.

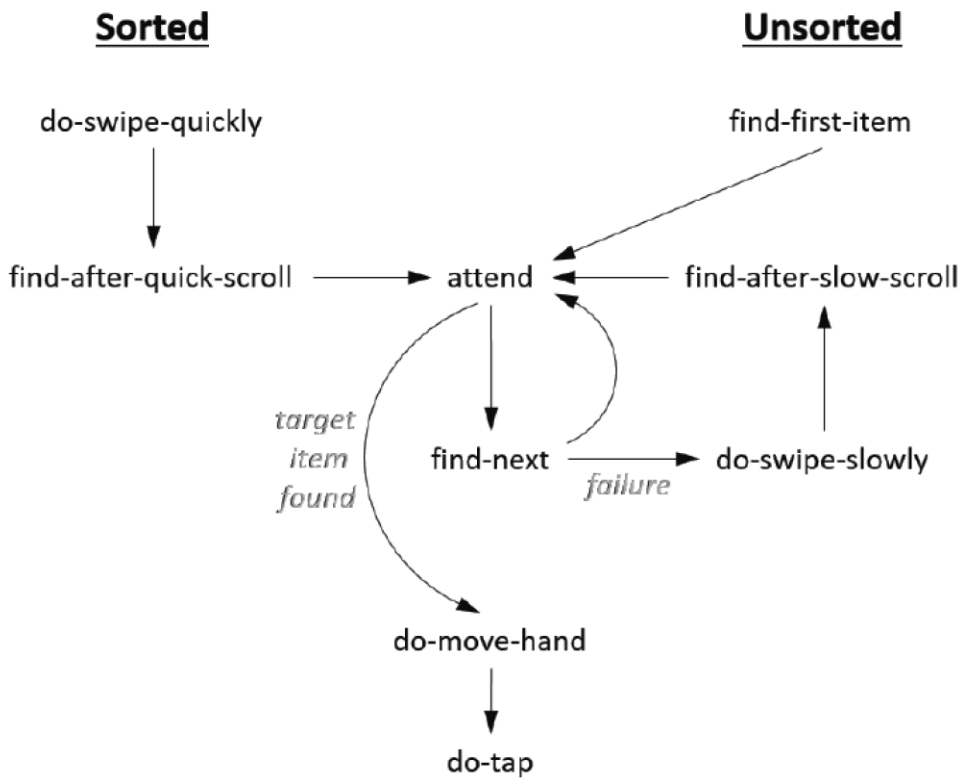

Figure 2: Production graph

If the model indicates that the list is ordered, the model will start the search task by performing a quick, short swipe to get to the end of the list (do-swipe-quickly) (see figure 2). Afterwards, a search through the visible items from top to bottom (find-after-quick-swipe, attend, find-next, attend, …) is performed. If the chunk in the visual buffer matches the correct item, it moves the hand (do-move-hand) and taps on the item (do-tap). This takes the model 3.008 seconds.

If the list is randomly ordered (pre-knowledge indicates a random list), then the model must search this list exhaustively. The model will omit the quick swipe in the beginning and start searching from the top (find-first-item, attend, find-next, attend, …). When no visual-location below the current can be found, the model "knows" that it reached the bottom of the screen and swipes slowly to make about four new items visible (do-swipe-slowly). This is repeated until it finds the correct item. It takes 4.112 seconds to find and select the target item.

Study Searching through Lists via Scrolling

To demonstrate that different cognitive strategies are used we looked at targets positioned at the same place on a searchable list (e.g. upper position vs. lower position in a list over several screens) that result in different search times and qualitative behaviors depending on the organization of the lists. Furthermore, we want to show that an ACT-R model connected with ACT-Droid and using

ACT-Touch can resemble these behavioral patterns, both quantitatively and qualitatively.

More precisely, **different** scrolling behaviors should emerge depending on whether the lists are ordered randomly. On the one hand, the participants are required to inspect each item in the random lists visually. It is expected that a random order will result in slower scrolling behavior. On the other hand, with the ordered version (either via alphabet or numbers) participants can use their expectations to guide their search behavior. Thus, it is expected that they scroll faster (or flick) until they reach the predicted position of the target.

A further proof of concept was to test to what extent participants behavior such as scrolling and regular behavior can be captured with ACT-Touch in combination with ACT-Droid.

Application: The study was conducted with a modified version of a real-estate application for Android (Prezenski & Russwinkel, 2016). This application has a hierarchical list style design. Different search condition for real-estate, such as city district, number of rooms, or additional features, can be selected to define a search for real-estate. The criteria are presented in hierarchically organized lists. Each list can range over 1 to 3 screens. For example, to a select 3-roomapartment, users are required first to select room-size and then select "3" in a list.

In the study, two versions of the application were used. In one version the condition in the lists are ordered either alphabetically or numerically (ORDER) in the other version they are ordered randomly within the lists (RANDOM).

Task: The participants were asked to imagine that they were searching for real-estate together with a relative of theirs who is not familiar with smartphone usage. The task of the participants was to search for different properties. Hereby, they had to search and select criteria (e.g. "apartment" with "3 rooms" and "70 m²" in "Berlin" in the district "center" with a "balcony" for "800 €" and an "elevator") using the app. The criteria were read to the participants one by one. The criteria required the participants to scroll down in the app. The app was installed on a Google Nexus 4 smartphone (4.7- *inch*, 1280 x 768 *pixels, 320 ppi)*, running Android 4.4 (Kitkat).

The participants were instructed to hold their smartphone in their hand. This was done to achieve a more natural interaction posture.

Design: For each real-estate search, eight different criteria had to be selected. The participants searched for real-estate with varying criteria three times. After this, the version was changed without notifying the participants, and they searched three times again for different real-estate with eight criteria. Half of the participants began with the ORDERED version, the other half with RANDOM version.

To ensure that differences in search time resulted from the scrolling behavior and not on the

positioning of the criteria, the target criteria of both versions were positioned at the same places in both versions of the app.

The “task time” was measured from the selection of the correct criterion to the selection of the correct item. Or in other words, the time the participants needed to find an item in a list. Only the swipes during that “task time” and longer than 100 pixels were considered. Touch interactions for a shorter distance were assumed to be taps, long taps, or a gesture that had no consequences for this experiment. As a straightforward analysis we calculated the mean speed of every swipe during a search within a criterion and then took the average over them. This way we had an indication of how fast/slow the participants scrolled.

Participants: The participants were 30 students (17 female and 13 male), between 21 and 31 years old. 28 were right-handed and 2 were left-handed. Their average self-reported smartphone usage was 3.46 hours per day. They self-rated their experience with smartphone apps 3.47 (sd. 1.25) on a scale from 1 to 5 (1 being no experience and five being a lot of experience).

Results

Trials with small errors like swiping too far were not discarded by default. But outliers were handled systematically, data points of more than 2.5 interquartile distance from mean value were discarded. The calculated t-tests focused on general differences between interaction behavior for sorted and unsorted lists (compare with figure 3). Significant differences were found for task time, touches and speed. Paired (task time) $t(29) = -9.825$, $p \leq 0.001$; paired (touches) $t(29)=-6.125$, $p \leq 0.001$; paired (speed) $t(29)=6.275$, $p \leq 0.001$.

Thus, for a better analysis we divided items according to upper (beginning of list – first page), middle and lower items (at the end of the list). It was expected that the difference for sorted and unsorted lists was especially pronounced for items at the end of the list – we called lower items (see figure 4), that incorporated probably more touch interactions. Therefore, a special analysis for those cases was made. Again, a t-test analysis was conducted. Significant differences for lower items were found for task-time, touches and speed between sorted and unsorted lists. Paired(task-time_low) $t(29)=-9.393$,$p \leq 0.001$; paired(touches_low) $t(29)=-4.812$, $p \leq 0.001$; paired(speed_low) $t(29)=6.241$, $p \leq 0.001$.

The difference in task time between sorted and unsorted lists might also account for the visual search behavior and not just swipe behavior. Participants needed more visual attention to inspect numerous items especially for low positioned targets, because the target could have been anywhere. In the ordered search, participants needed this search behavior only for the items at the end of the list. The speed and number of touches in the different conditions give us a better impression of the different touch interaction user strategies, as explained above.

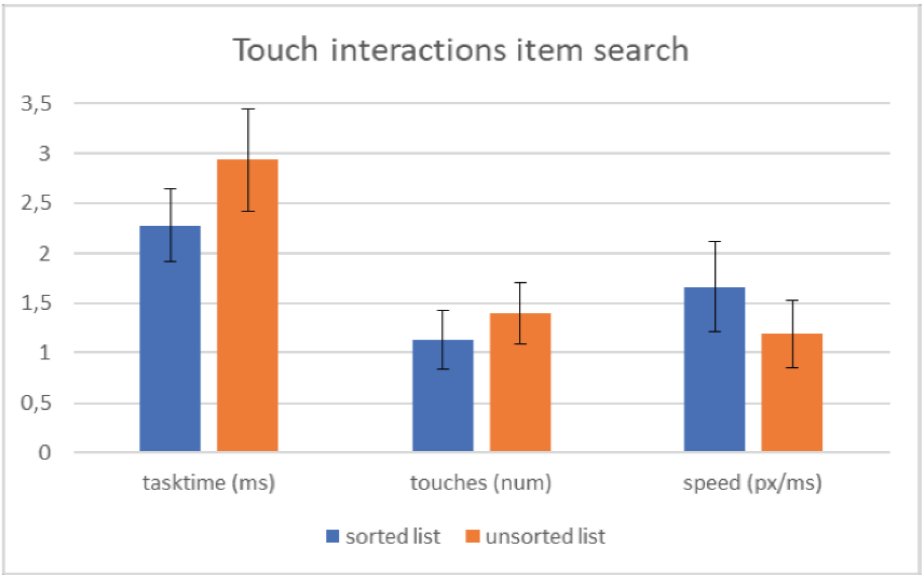

Figure 3: Touch interactions divided for sorted and unsorted lists (including standard deviation)

The results show more touches for unsorted lists. The number of touches would have been more pronounced for even longer lists than we tested. The results also show higher speed in swipes for sorted lists, probably used to quickly get to the part of the list where the target is expected to be.

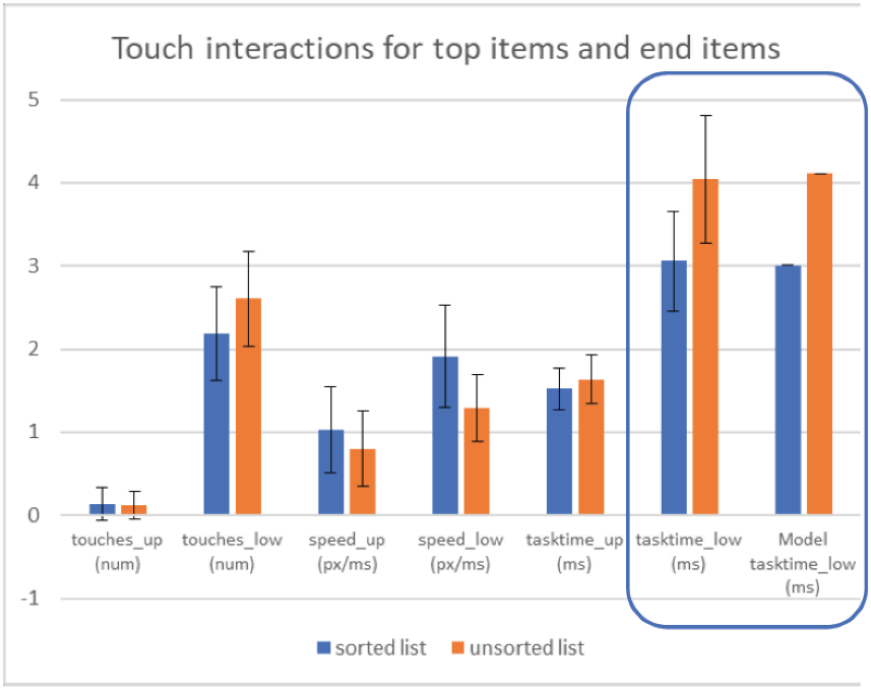

Figure 4: Touch interactions divided for lower and upper search targets of participants (incl. std.), and comparison to Model data for task time_low

Those time differences and touch differences do matter because they accumulate for every search list and already show a difference of one second for one selection step in our relatively short lists. These data represent what differences arise in such reoccurring general task components and that it is necessary to account for such differences in the task model in cases where reaction times are relevant.

A more detailed analysis of the specific kind of different touch interaction is the topic of another publication. For being able to capture interaction times for applied tasks and evaluation of technical systems we do not need to model the touch interaction in detail. It is sufficient to capture the general differences if we know that we find the item at the end of the list or not. This is not necessarily always caused by a sorted or unsorted list but can be the result of learning certain visual location or navigational heuristics. Having a good idea where and how to find the target quickly is what makes the difference.

Discussion

We introduced a new tool, which connects an Android app with an ACT-R user model. Furthermore, the tool makes prototyping of the application in Lisp obsolete by allowing the model to directly act upon the real app. This tool uses the ACT-Touch commands to simulate realistic touch interactions thus making it possible to evaluate user interfaces and interactive behavior more precisely and provide a deeper understanding of the underlying processes.

In the current paper, empirical data is presented to show how different kinds of swipe interactions are used and what impact this has on task time. These behavioral differences may not only occur due to sorted and unsorted lists. Other reasons for different swipe behavior might be pre-knowledge about the position of certain icons or targets or strict time constraints. More research is needed to understand these kinds of influences on behavior.

Realizing scrolling behavior has been a major burden in several mobile device experimental setups. Scrolling behavior was a necessary touch interaction in most applications that we looked at. Furthermore, task time is a relevant key factor for usability tests evaluating whether one application is better than competing apps. Our presented approach solves these problems.

Since more and more touch displays are integrated into larger technical systems such as cars or the cockpit of airplanes, we find a pronounced request from industry to understand how operators work with such touch displays within a realistic task.

Now that scrolling behavior on touch displays can be modeled in simulation, this could be helpful for developing and evaluating assistant systems, especially in driving situations (e.g. Salvucci, 2006). These models would be tools to understand, describe and predict in detail how operators interact with technical systems. This method could help to decide which kind of interface design is more effective in some sense than another.

References

Anderson, J.R., Bothell, D., Byrne, M.D., Douglas, S.,Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review, 111, 1036-106. al Review*, 111, 1036-106.

Bothell, D. 2017. ACT-R Reference Manual. http://actr. psy.cmu.edu.

Burnett, G., Crundall, E., Large, D., Lawson, G.,& Skrypchuk, L. (2013). A Study of Unidirectional Swipe Gestures on In-Vehicle Touch Screens. In Proceedings of the 5th International Conference on Automotive User Interfaces and Interactive Vehicular Applications (pp. 22- 29). ACM.

Doerr, L.-M., Russwinkel, N., & Prezenski, S. (2016). ACTDroid: ACT-R Interacting with Android Applications. In D. Reitter & F. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (pp. 225-227). University Park, PA: Penn State.

Gray, W. D., & Boehm-Davis, D. A. (2000). Milliseconds matter: An introduction to microstrategies and to their use in describing and predicting interactive behavior. *Journal of Experimental Psychology: Applied, 6*(4), 322-335.

Halbrügge, M. and Russwinkel, N. (2016). The sum of two models: how a composite model explains unexpected user behavior in a dual-task scenario. In D. Reitter & F. E. Ritter (Eds.), *Proceedings of the 14th International Conference on Cognitive Modeling*. University Park, PA: Penn State.

Ofcom. 2017. International Communications Market Report 2017.

https://www.ofcom.org.uk/__data/assets/pdf_file/0032/108896/icmr-2017.pdf

Patterson, B. 2013. iPad tip: 3 nifty iPad gstures you need to try. https://heresthethingblog.com/2013/12/11/ipad-tip-3-nifty-ipad-gestures/ Prezenski, S.& Russwinkel, N. (2014). Combining cognitive ACT-R models with usability testing reveals users mental model while shopping with a smartphone application. Int.J. Adv. Intell. Syst., 7(3), 700- 715.

Prezenski, S. & Russwinkel, N. (2016). Towards a general model of repeated app usage. In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (pp. 201- 207). University Park, PA: Penn State.

Ritter, F. E., Van Rooy, D., & Amant, R. S. (2002). A user modeling design tool based on a cognitive architecture for comparing interfaces. In Computer-Aided Design of User Interfaces III (pp. 111-118). Springer, Dordrecht.

Salvucci, D. D. (2006). Modeling driver behavior in a cognitive architecture. Human factors, 48(2), 362-380.

Statista. 2017. "Number of Google Play Store apps 2017 | Statistic". Retrieved 2018-01-03.

Tamborello, F. 2018. ACT-Touch. https://github.com/tamborello/ACT-Touch/

Tsukayama, H. 2017. "Apple stock soars to a record high on great earnings and a strong forecast for the next iPhone". The Washington Post. Retrieved August 2, 2017.