In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016). University Park, PA: Penn State.

ACT-Droid: ACT-R Interacting with Android Applications

Lisa-Madeleine Dörr (lisa-madeleine.m.doerr@campus.tu-berlin.de)
Department of Cognitive Modeling in Dynamic Human-Machine Systems
Technische Universität Berlin

Nele Russwinkel (nele.russwinkel@tu-berlin.de)
Department of Cognitive Modeling in Dynamic Human-Machine Systems
Technische Universität Berlin

Sabine Prezenski (sabine.prezenski@tu-berlin.de)
Department of Cognitive Modeling in Dynamic Human-Machine Systems
Technische Universität Berlin

Keywords: ACT-R; Android; granularity; usability testing; modeling; mobile context; tool.

Abstract

A tool for directly connecting ACT-R with Android applications on smartphones or tablets is introduced. The advantage of this tool is that no prototyping of the application is needed. This tool is especially useful to evaluate applications according to usability by using general modeling approaches.

Motivation

The number of Smartphone applications is growing rapidly. Likewise the demand for efficient usability testing methods is increasing. Cognitive models have the potential to meet this demand. Cognitive models developed for one application can be reused for testing similar applications (Prezenski & Russwinkel, in submission). Thus, costs of intensive user testing can be reduced to a minimum.

Nevertheless, the necessity of connecting the interface to ACT-R remains a major issue. The interface has to be translated into a format the ACT-R model can interact with. A number of tools have been developed to connect ACT-R to simulations (e.g. ACT-CV: Halbrügge, 2013, Hello Java: Büttner, 2009; Agimap: Urbas et al., 2009; SIMCog-JS: Halverson, Reynolds & Blaha, 2015, JNI: Hope, Schoelles & Gray, 2014 and others).

For tasks involving interactions with an interface one solution is to develop prototypes as, for example, in CogTool (John, Prevas, Salvucci, & Koedinger, 2004). The solution of creating prototypes of apps for the cognitive model is problematic for three reasons. First, it is a time consuming process. Second, the granularity of the prototype can affect the validity of the results. And third, depending on the specific usability questions new prototypes might be necessary, e.g. for questions addressing finer granularity.

This paper introduces ACT-Droid, a tool that allows ACT-R models to directly interact with Android smartphone applications. Thus, prototyping of applications becomes obsolete and testing usability with cognitive models a realistic goal.

With ACT-Droid no artificial tools need to be developed, ACT-Droid is a further development of Hello Java (Büttner, 2009). It directly connects to the Android app, identifies buttons and other items and can interact with them.

Technical Details

The two main tasks ACT-Droid fulfills are: performing motor output of ACT-R at the app and updating the visicon of ACT-R according to the changing app screen. These functionalities are provided by the *model interface* and the *app interface*, which communicate with each other.

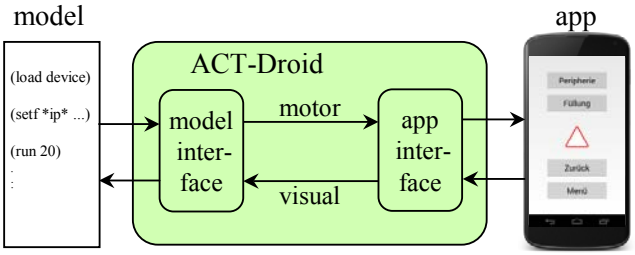

Figure 1: Architecture of ACT-Droid.

The app installed on a smartphone communicates with ACT-R over TCP/IP sockets. If the extended app is started, the *app interface* establishes a server socket and the *model interface* connects ACT-R as a client.

Motor

Currently, ACT-R's mouse commands are interpreted as fingertip touches by the Android app. So, each time the cursor is moved by the model, the *model interface* sends the new cursor position to the *app interface*. The *app interface* saves the current position of the cursor. Furthermore, if the command to click is received, the *app interface* performs a click at the saved cursor position.

Visual

The most important functionality of the *app interface* is to provide all visible information whenever the visicon of ACT-R requires updating. All visible information is recursively searched and descriptions of any visible

checkboxes, buttons and textfields are generated. This description consists of: kind, value (usually its text), color, size and position.

For the FiguRapp example (Lindner & Russwinkel, 2015) included in the figure above, the description contains four buttons (which in ACT-R is considered of the kind "oval") with their respective values "Fuellung", "Peripherie", "Zurueck" and "Menue" and a triangle of the kind "triangle" with the color "red" and no specified value.

The description provided by the *app interface* is send to the *model interface*. The *model interface* then reloads the visicon according to the received information.

What is Possible

With ACT-Droid the ACT-R model interacts with the actual Android app, the interaction is fully automated.

Furthermore, in the case of uncommon GUI elements, ACT-Droid enables the modeler to define how these elements should appear in the visicon. In the FiguRapp, for example, the image of a red triangle is defined to be of the kind "triangle" and of the color "red". But alternatively, it could also be of the kind "figure", of the color "red" and have the value "3" (for the nodes). Thus, the content of the screen can be described as detailed as necessary. The granularity is only limited by the structure of the visicon.

Due to different encodings, problems often occur with German umlauts. ACT-Droid replaces these by their respective two "normal" letters. This approach can easily be applied to other characters.

How to

The prerequisites for using ACT-Droid are the following: a computer with Lisp environment (e.g. Lispworks), ACT-R source files (the standalone does not work) and Android Studio with the source files of the app. Furthermore, an Android smartphone to run the extended app on is needed, because the emulator will not work.

ACT-Droid can be downloaded from http://dx.doi.org/10.14279/depositonce-5181. Detailed instructions in "Readme.txt" and the FiguRapp example are also included. A very basic ACT-R model that will also run on the FiguRapp is provided. This simple model will randomly explore and click on everything. To use other ACT-R models with ACT-Droid, model-interface.lisp has to be loaded and parameters have to be defined at the beginning of the model, e.g. the IP address of the smartphone.

Furthermore, a few modifications of the apps source code are necessary, i.e. adding the lispcom package and three lines of code to the apps main activity. All this is described in the material. Once the description in ACT-R's visicon is satisfying, there is no need for editing the app any further.

After everything is set up, the app on the smartphone has to be started first and then the ACT-R model can be run using the command do-experiment. The model will directly interact with the app.

Outlook

Until now, implementing scrolling with ACT-Droid has not been considered and a thorough test with different apps is pending. Another objective is the further simplification of the set-up and usage, e.g. when having more than one Android activity (common).

Currently, we are working on replacing mouse commands with the touch commands of ACT-Touch (Greene, Tamborello, & Micheals, 2013). This is the next step towards an adequate tool for efficient usability testing of apps.

References

Büttner, P. (2010). "Hello Java": Linking ACT-R 6 with a Java Simulation. In D. D. Salvucci & G. Gunzelmann (Eds.), *Proceedings of the 10th International Conference on Cognitive Modeling* (pp. 289-290). Philadelphia, PA: Drexel University.

Greene, K. K., Tamborello, F. P., & Micheals, R. J. (2013). LNCS 8007 - Computational cognitive modeling of touch and gesture on mobile multitouch devices: applications and challenges for existing theory. *Proceedings of the 15th international conference on Human-Computer Interaction* (pp. 449-455). Las Vegas, USA: ACM.

Halbrügge, M. (2013). ACT-CV: Bridging the Gap between Cognitive Models and the Outer World. In Brandenburg, E., Doria, L., Gross, A., Güntzler, T., and Smieszek, H., eds., *Proceedings of the 10th Berlin Workshop Human-Machine Systems* (pp. 205-210). Berlin, Universitätsverlag der TU Berlin.

Halverson, T. B. Reynolds, B. and Blaha., L. M. (2015) SIMCog-JS: Simplified interfacing for modeling cognition - javascript. *Proceedings of the 13th International Conference on Cognitive Modeling* (pp. 39-44). Groningen, The Netherlands.

Hope, R. M., Schoelles, M. J., & Gray, W. D. (2014). *Simplifying the interaction between cognitive models and task environments with the JSON Network Interface*. Behavior research methods, 46(4), 1007-1012.

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* (pp. 455–462). New York, NY, USA: ACM.

Lindner, S., & Russwinkel, N. (2015). The Effect of Design Decisions on User Expectations - A modelling approach. In C. Wienrich, T. O. Zander, & K. Gramann (Eds.), *Proceedings of the 11th Berlin Workshop Human-Machine Systems* (pp. 98–102). Berlin: Universitätsverlag der TU Berlin.

Prezenski, S., & Russwinkel, N. (in submission). Towards a general model of repeated app usage. In D. Reitter & F. E. Ritter (Eds.), *Proceedings of the 14th International Conference on Cognitive Modeling*. Pennsylvania.

Urbas, L., Heinath, M., Troesterer, S., Pape, N., Dzaack, J., Kiefer, J., & Leuchter, S. (2006). Agimap: A tool chain to support the modelling of the interaction level of dynamic

systems. *Tutorial at Trieste: Proceedings of the Seventh International Conference on Cognitive Modeling* (pp. 409). Trieste, Italy.