This is an authors version. The original paper can be found at http://www.scitepress.org/DigitalLibrary/PublicationsDetail.aspx?ID=unW9c8ilLOQ=&t=1
This paper was presented at HUCAPP 2017 http://www.hucapp.visigrapp.org/?y=2017
Please cite: Prezenski S., Bruechner D. and Russwinkel N. (2017). Predictive Cognitive Modelling of Applications.
In Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 2: HUCAPP, (VISIGRAPP 2017) ISBN 978-989-758-229-5, pages 165-171

# Predictive Cognitive Modelling of Applications

Sabine Prezenski[1], Dominik Bruechner[1] and Nele Russwinkel[1]
[1]*Cognitive Modelling in dynamic Human-Machine Systems, Technische Universität Berlin, Marchstraße 23, Berlin,Germany*
{*sabine.prezenski, d.bruechner, nele.russwinkel*}*@tu-berlin.de*

Keywords: Cognitive Modelling, Usability, ACT-R, Tool, Usability Criteria, Usability Test

Abstract: This paper argues that important usability aspects of mobile applications can be automatically evaluated using computational cognitive models based on the cognitive architecture ACT-R. A tool incorporating cognitive models for specific tasks, users, applications and usability aspects is proposed. Explanations provided by the tool for usability flaws are based on simulations of cognitive mechanisms. A use-case of the tool is introduced, which is based on an ACT-R model that simulates how users search and select a specific target in a hierarchical android application and predicts efficiency and learnability for average users. The model has been empirically validated in four studies with two different applications. To fully automate the usability evaluation of the use-case, two basic requirements need to be fulfilled. First, the application and the cognitive model have to be connected. A tool called ACT-Droid acts as an interface between the Android application and the cognitive model. Second, the models knowledge of the world, which is application specific, has to be provided automatically by using an automated user interface analysation approach. Therefore, the open-source tool AppCrawler was extended to allow the extraction of the required information.

## 1 INTRODUCTION

More than 2.2 Million applications are available at Google Play store today in comparison to 1.7 Million one year ago (AppBrain, 2016). Last year, the number of applications that were available for Android increased by over half a million. Every day, over a thousand new applications appear on the market with some of them being successful. Upon other aspects, such as marketing and visibility, very good usability is a key to successful applications. It appears unfeasible that each newly launched app is tested in terms of usability. Nevertheless, developers ask themselves if users will like their app and if it is better than those of their contestants. There are a lot of rivaling applications and good usability can be the critical factor of success. Examples of relevant questions addressing the usability of mobile applications are: How long do users take to perform a task with a specific app? Does user performance increase as users have more practice with an app? Should an app adapt to the users' preselection? How should an adaption look like? Will an update of the app irritate the user? Different methods (e.g. user testing, expert interviews) provide answers to these questions. However, these methods are both time and money consuming, and the results often do not provide a clear explanation of the findings. Furthermore, changes in an app often require a new evaluation. We are currently developing a fully automated approach for Android applications that will automatically predict important usability aspects (such as efficiency) based on cognitive models that simulate cognitive processes of users. Our approach will furthermore offer insight into the cognitive reasons behind usability flaws.

We want to present a prototype aimed on one specific use-case, namely the automated usability evaluation of a hierarchical style Android application for an average user (Prezenski and Russwinkel, 2016). Therefore, it is required to extend an existing cognitive model (Prezenski and Russwinkel, 2016) to be applicable to a multitude of hierarchical list style applications without any prior modification of the model. Thus, any model aspects that are specific for the application (such as relevant semantic knowledge) should to be automatically extracted from the user interface of the mobile application.

In this paper first a theoretical introduction about usability and cognitive models in HCI is given. Secondly, a general automated usability evaluation approach based on ACT-R cognitive models is introduced. This is followed by the definition of a more specific use-case for hierarchical list style applications to base a working prototype on. Requirements

on the approach are then defined to make it applicable on our use-case. Finally, future steps are elaborated followed by a discussion.

# 2 THEORY

## 2.1 Usability

The term usability defines how efficient, effective, and satisfying the use of a technical system is (ISO-9241-11). The PACMAD (People at the Centre of Mobile Application Development) complements this usability definition with aspect important for mobile usability, namely learnability, memorability, users, task, context, and mental load (Harrison et al., 2013). Note that the usability of mobile applications should not be seen as a smaller desktop, but different aspects need to be considered.

While applications exist for almost every imaginable task, from selecting a recipe to planning a vacation, a single app mostly supports only one or two main tasks (e.g. messenger applications are mainly used for reading and sending messages or recipe applications for searching for recipes). And although on the surface applications appear incredible divers (e.g. different colors, font, framing of images), the core principles (e.g. menu-structure, organization) are similar for a large amount of applications. In our view these underlying core structures have a strong influence on many usability criteria. This similarity of structures in a wide range of applications and the focus on one or two core functions for most applications is why we propose that usability of applications can be analyzed using computational cognitive models. We aim to develop cognitive models for all aspects of PACMAD, but will begin developing models for the usability aspects of efficiency and effectiveness and learnability. In (Nielsen, 1994) a set of nine heuristics for different usability aspects were presented which can be applied during the application design process. In the past these were not automated testable. For some of those heuristics for good usability cognitive models could be developed as well.

## 2.2 Cognitive Models

In our understanding, computational models are cognitive models if they can simulate human behaviour by simulating ongoing cognitive processes and mechanisms both on a symbolic and sub-symbolic level. We want to simulate the process of humans in order to understand and predict their behaviour.

### 2.2.1 ACT-R

Our cognitive modelling approach uses the cognitive architecture ACT-R (Adaptive Control of Thought Rational) (Anderson, 2007). In general terms, cognitive architectures are computational platforms that allow the creation of models on the basis of how humans process information. Cognitive architectures constrain models in a way that only cognitively feasible models are possible. More than simply reproducing the behavioural output, cognitive architectures reveal underlying cognitive mechanisms. ACT-R is the most widely used cognitive architecture and it has been applied in many domains from air traffic control (Raufaste, 2006) to mathematical tutors (Ritter et al., 2007). It is open source and used by an active community of researchers. The ACT-R approach aims to achieve an unified theory of cognition in the future, and is the most successful computational approach of modelling human cognition on a symbolic level that we are aware of.

The ACT-R architecture consists of different modules (representing the modular structure of the human processing) and buffers (the communication interfaces of these modules). See Figure 1 for an overview of the most important buffers (arrows) and modules (boxes) of ACT-R. For example, there is a motor module for motor processes such as key-presses and a visual module (for visual processing). ACT-R also has a working memory module (imaginal module), which is important for learning and combining information and a declarative memory module where knowledge (chunks) is stored and retrieved from.

Writing an ACT-R model normally requires the modeler to specify the models' fact knowledge manually (declarative knowledge or chunks) and its rules (production rules, procedural knowledge). These production rules are the core part of the model. They consist of an *if* and a *then* part. The *if* part refers to the state and content of the buffers (e.g. if they are used and if they hold specific chunks) and the *then* part modifies the content and state of the buffers.

Next to the productions rules, the processing of information is governed by numerous sub symbolic processes, which have been added to the architecture only after empirical studies have confirmed them. For example, the retrieval of chunks from declarative memory is determined by the sub-symbolic activation value of the chunk, which depends on when the chunk was last used.

### 2.2.2 Cognitive Models on HCI aspects

A number of ACT-R models concern menu search or mobile interaction: (Byrne et al., 1999) demonstrated

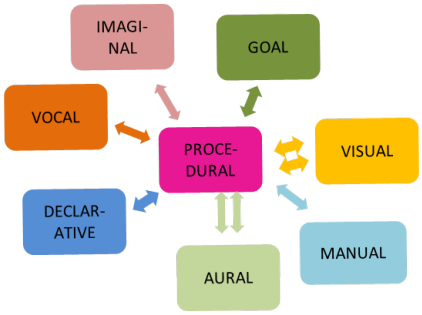


Figure 1: Here a selection of the modules (boxes) and buffers (arrows) of the ACT-R architecture is represented. Note that some modules have two buffers (e.g. the visual module).

that ACT-R models are capable of humanlike menu search in a controlled study. In this study, performance in a random menu selection task on a computer was compared with eye tracking and behavioural data. (Das and Stuerzlinger, 2007) modelled the learning of mobile phone usage for elderly novice users. The increase in performance in their model was realized through remembering locations of keys. (Gallagher and Byrne, 2015) investigated password entry on smartphones and offered model-based explanations on how requirements for complex passwords on smartphones should be designed to meet both users' abilities and safety requirements. These studies indicate that expert and novice behaviour, menu search, as well as learning behaviour can be simulated using cognitive architectures.

### 2.2.3 Modelling approaches of usability

The modelling approaches presented above show only limited usability prediction based on existing cognitive architectures. The most successful approach is CogTool (John et al., 2004) and its' successor CogTool Explorer (Teo and John, 2008). These approaches can predict expert and novice search behaviour on websites. Some aspects of the approaches are based on ACT-R, but they do not use the full power of ACT-R due to a lack in memory and learning mechanisms. Thus usability aspects based on learning or related aspects, such as repeated usage, cannot be addressed by the CogTool approaches. Furthermore, much manual work (e.g. preselecting ideal paths) is required when working with CogTool.

Other modelling approaches of user behaviour are GOMS-based cognitive models. GOMS models are often used in HCI since their implementation is effortless and simple. Modelers specify Goals, Operators, Methods and Selection Rules to simulate user behaviour. Many GOMS models provide an acceptable prediction on the time that skilled users need for a predefined task. The straightforward approach of GOMS models makes them convenient to use; task are divided into subsets of operators with assigned time value. Although these models are declared "cognitive" they mainly consist of different motor and visual operators. A single mental operator (with a specific time value) represents the entire cognitive process of "thinking", which is oversimplifying for many evaluation questions. GOMS models are a great approach whenever the objective is to find out how much time a skilled user will need for a task. Other (non-cognitive) usability modelling tools exist as well, in that they use different computational algorithms to analyze aspects such as font-size, color, contrast (Amalfitano et al., 2012; Choi et al., 2013).

Two major challenges have prevented modelers from evaluating usability of mobile applications with cognitive architectures and are the reasons why tools with diminished explanatory power have been used instead. The first challenge is the necessity to (re)-construct an interface the model can interact with. We have developed a new tool, ACT-Droid (Dörr et al., 2016), that directly connects Android applications with an ACT-R environment, making mock-up creations or translations obsolete. The second challenge is the high amount of expertise that is currently required to construct cognitive models with cognitive architectures. The more different the applications are in terms of general interaction mode, the more effort it takes for the modeler to transfer a model to another app even if the cognitive mechanisms in the model (e.g. learning mechanism) are the same. Therefore, we are focusing the modelling work on transferable cognitive models. These can be reused for other similar applications without effort. Such a model exists for hierarchical list style applications. It has successfully predicted empirical user behaviour (such as repeated application usage) for two different applications. The studies and results are described in more detail in (Prezenski and Russwinkel, 2016) and are summarized in the following: The two applications were a shopping-list application, which allows users to select items out of different stores and categories and a real-estate application from which users can select search criteria (e.g. search for an apartment with 60m). Four empirical user studies, two with each application, were conducted. The users were required to repeatedly search for seven criteria with the same version of the application twice, and then the application was modified either due to an update (shopping-list application) or due to adaptation to prior selected search criteria (real estate application). Two different versions of both applications existed – thus, in each

of the four studies, the participants started with a different application. The ACT-R model predicts mean overall item selection time of users for first and second time search. Furthermore it predicts search behaviour after updates accurately in terms of correlation and mean standard deviation (see Figure 2, originally appeared in (Prezenski and Russwinkel, 2016, p. 205).

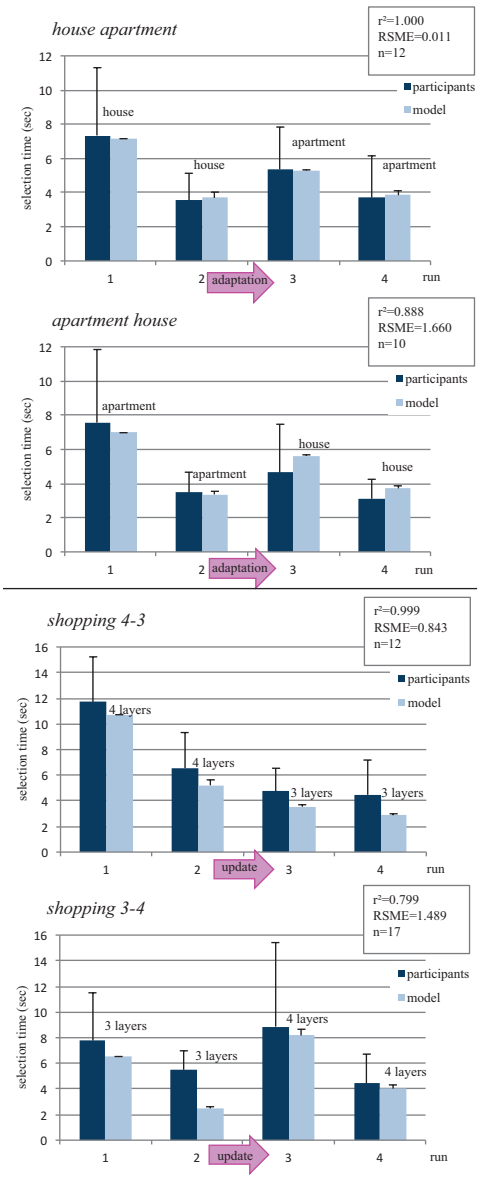


Figure 2: Mean target selection time for the four different studies for the modelled and empirical data (Prezenski and Russwinkel, 2016).

# 3 PROPOSED USABILITY EVALUATION APPROACH FOR MOBILE APPLICATIONS

## 3.1 Overall Aim

Our longterm goal is to develop a toolset which can automatically access important usability aspects of mobile Android applications as defined in PACMAD. In the development phase, features will be added subsequently and evaluated empirically.

Our approach is aimed at helping developers evaluate design alternatives in terms of cognitive aspects of usability. The toolset will output numerical values for different usability criteria such as efficiency (mean time on task) and effectiveness (proportion of successful attempts on whole attempts). Comparative values will be incorporated in the future as well. See Table 1 for an overview on the toolsets' proposed functionality. Testing an application will require the tool-users to pre-select certain application properties as shown in Table 1.

Table 1: Proposed functionality

| | |
|---|---|
| Application type | text-style |
| | menu type - hierarchical |
| | icon-style |
| | ... |
| Usability criteria | efficiency |
| | effectiveness |
| | learnability |
| | cognitive load |
| | satisfaction |
| | ... |
| Type of task | search & select specific target |
| | search & select unspecific target |
| | ... |
| Type of user | average user |
| | elderly user |
| | initial use |
| | second time use |
| | context |
| | ... |

The level of detail of the output will also depend on the developers previous selections. The output will consist of numerical values (such as mean and standard deviations, a comparative value in relation to other similar applications) and a more cognitive output, revealing potential cognitive causes of usability flaws (e.g. retrieval failures for uncommon words).

For different applications (e.g. icon based vs. text based), different tasks (e.g. search and selection of a specific targets, search for an unspecific target), and different usability concerns (e.g. efficiency, learnability), the cognitive modelling community has come up

This is an authors version. The original paper can be found at http://www.scitepress.org/DigitalLibrary/PublicationsDetail.aspx?ID=unW9c8ilLOQ=&t=1
This paper was presented at HUCAPP 2017 http://www.hucapp.visigrapp.org/?y=2017
Please cite: Prezenski S., Bruechner D. and Russwinkel N. (2017). Predictive Cognitive Modelling of Applications.
In Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 2: HUCAPP, (VISIGRAPP 2017) ISBN 978-989-758-229-5, pages 165-171

with some solutions; however, much work is still required.

To move towards a fully automated usability evaluation approach based on cognitive models, we want to present a prototype for a certain use-case in this paper. Our use-case is the automated usability evaluation of a hierarchical list style Android application (cf. (Prezenski and Russwinkel, 2016) model for an average user). The model makes predictions for initial and repeated use and can evaluate the usability criteria efficiency (average time on task) and learnability (comparing first and further time on task). The model supports the selection of specific targets as a task.

## 3.2 Current State

### 3.2.1 ACT-Droid

ACT-R is written in the programming language LISP, and mobile applications are mostly developed for either iOS or Android. Enabling interaction between ACT-R models and mobile applications is crucial. To allow the model interaction for Android, ACT-Droid was developed (Dörr et al., 2016). ACT-Droid uses TCP/IP sockets to directly link ACT-R models with Android applications. Currently, we are working on integrating Touch-Commands for ACT-R (Greene et al., 2013) into ACT-Droid. Touch-Commands, such as swipe and peck, have been implemented in ACT-R with the ACT-Touch approach.


Figure 3: Overview of the Shopping-List Application.

### 3.2.2 Android Demo Application

A model to automatically evaluate cognitive aspects of usability in mobile applications has been developed (Prezenski and Russwinkel, 2016). We tested the model with a shopping-list demo application (see Figure 3). It allows searching for items which are organized in categories and subcategories. E.g. if a person (or the model) wants to select *alcohol-free beer* for their shopping list, they must first select *shops*, then *bottleshop* and then *beer* before they can select the target *alcohol-free beer*. Each subcategory is on a different page of the app.

### 3.2.3 ACT-R Model

When the model searches for a target the first time (initial use), it does not know which subcategory it has to select in order to find its targets. Thus, every item of each category is read and for each item the model uses its knowledge of the world to check if the target can be found under the current category. In other words, if there is an association between the target and the current item. For example, the model sees the word ”vegetables”. It will check its knowledge of the world if there is connection between ”vegetables” and its target ”alcohol-free beer”. If it can not find a connection in its’ knowledge of the world, it will read the next item ”beer” and search for a connection in its’ knowledge of the world. If it finds a connection, the model will select the category, remembering the position of the category and build up a path (in working memory) containing the categories leading to the target. Thus, it builds up experience and can later (e.g. when it is looking for the same target again) use its experience to navigate the target using the paths and position chunks stored in its declarative memory. Sub symbolic mechanisms, such as activation, influence if the chunks (paths, position and associations from world knowledge) can be retrieved.

Currently, the *knowledge of the world chunks* need to be added by hand for every new app which needs manual effort.

## 3.3 Towards a Fully Automated Interface Evaluation Approach

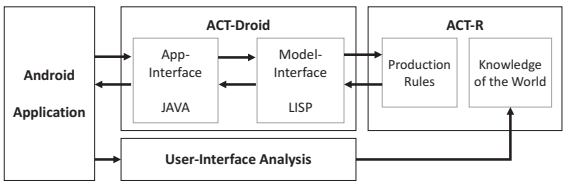


Figure 4: Data flow

In order to create a working prototype of our automated approach, it is required to extend the current ACT-R model to be applicable to a multitude of hierarchical list style applications without prior modification. Furthermore, we need to supply the knowledge of the world to the ACT-R model by automatically extracting it from the user interface of the mobile application. The data flow of our approach is presented

in Figure 4. ACT-Droid acts as an interface between the Android application and ACT-R. Additionally an UI analysis is executed beforehand (see 3.3.2).

#### 3.3.1 Extension of the Current Model

As a next step we want to demonstrate that automatically evaluating cognitive aspects of usability in mobile applications is feasible using the (Prezenski and Russwinkel, 2016) model. We will extend the model, so that it will be applicable for all hierarchical list style applications without any manual modification. Thus, we can automatically predict efficiency, effectiveness, and learnability for first and further time use of these applications. In order to understand which of the models' aspects need to be modified, the main model mechanism and an exemplary app which the model processes will be explained briefly. For a more detailed description of the model, see (Prezenski and Russwinkel, 2016). The model is for menu-based hierarchical applications that are text-based. It simulates search and specific target selection and can model both initial and repeated use.

#### 3.3.2 Automated UI Analyzation to Create World Knowledge

Currently, the *knowledge of the world chunks* need to be added by hand for every new app. We want to automate this process using a program that automatically enables the creation of new chunks in ACT-R declarative memory module. The chunks will be the connection nodes between all possible targets in an app and the categories and subcategories leading to the targets. The chunks will always contain only two elements (e.g. the target and a category).

To acquire this *knowledge of the world chunks* automatically, it is required to analyse the complete user interface of the mobile app to evaluate beforehand. For our first use-case, the analyzation of an android app, an OpenSource approach to crawl user interfaces called AppCrawler, was extended. This application is an automatic UI testing tool for Android based on UIAutomator, a UI testing framework suitable for cross-app functional UI testing across system and installed Android applications.

AppCrawler uses a Depth-first search algorithm (Cormen, 2009) to traverse through an apps interface. DFS is an algorithm for traversing or searching tree or graph data structures (Cormen, 2009). By using the DFS for our use-case, every screen of the application is traversed.

The existing implementation needed to be extended to extract the text value of each clickable menu item. Additionally, it was necessary to extend the AppCrawler to store all menu-item to target relations which are then used to create the knowledge of the world for the ACT-R model. Targets in this case are the leaves of the search tree, e.g. alcohol-free beer, menu-item to target relations would be bottle-shop - alcohol-free beer and beer - alcohol-free beer. The automatically extracted information can then be supplied as *knowledge of the world chunks* to the ACT-R model.

#### 3.3.3 Pre-Activation of Targets

In a further step, the pre-activation of these targets will be weighted depending on how often they appear in the same sentence in written language databases (such as Wikipedia). Such an approach is used in combination with ACT-R and described in more detail in CogTool Explorer (Teo and John, 2008).

## 4 DISCUSSION

Some questions remain open and more work is required until a developers' version of the tool can be launched. For many aspects of the tool, cognitive models still need to be developed and evaluated.

Currently, we are exploring possibilities to model elderly users with ACT-R and incorporate such mechanism into the tool.

We would also like to extend the cognitive models for different cultural backgrounds (e.g. reading direction in arabic).

Furthermore, the UI analyzation approach needs to be extended to cover different kinds of app layouts (e.g. text-style, icon-style) as well as different app development platforms (e.g. iOS).

As a next step, we want to apply the modelling approach of the use-case to a common used application and compare the model results with user data. If this is successful, this is a strong proof of concept that ACT-R-based cognitive modelling is indeed useful for usability testing.

We are often asked why we attempt to model with ACT-R instead of using machine learning approaches to simulate user behaviour. With enough training and data, neural network approaches (e.g. machine learning algorithms) can probably predict user behaviour as well; however they cannot offer any insight into why these results are obtained. The cognitive processes of users leading to the outcome are a black box. Thus, no advice on cognitive aspects to designers on how to change the applications can be given by such an approach in comparison to an ACT-R approach. Given that the use-case is fully automated

(see 3.3.2) and pre-activation of the *knowledge of the world chunks* is available (see 3.3.3), an ACT-R model can help with finding problems related to cognitive aspects. Failures of a model related to a missing *knowledge of the world chunk* for a target (*e.g. the model is searching for alcohol-free-beer, it looks at bottle shop but can't retrieve a knowledge of the world chunk linking bottle shop to to alcohol-free beer*) can be interpreted to solve those problems. For example, a retrieval failure of the *knowledge of the world chunk* can occur because it had too low of an activation value and this should be interpreted as a bad labeling choice (*rename bottle shop*).

Furthermore, our approach (other than machine learning approaches) does not require a large amount of data to work. Once the ACT-R model has been tested for smaller empirical sample sizes (e.g. 20 participants) it is possible to estimate reliable usability criteria if it predicts the main usability criteria accordingly. If the fit to the empirical data is satisfying, the model should be tested with another similar app, and if it is successful again, it will be incorporated in the tool.

# 5 LIMITATIONS

ACT-R's level of detail on predicting visual processes is not detailed enough to simulate how complex visual information is processed and perceived, as is used in map or game applications. Thus, deciphering usability effects of such applications is out of the scope of our tool. We are looking for cooperations with other usability approaches that focus more on analyzing visual processing.

# REFERENCES

Amalfitano, D., Fasolino, A. R., Tramontana, P., De Carmine, S., and Memon, A. M. (2012). Using gui ripping for automated testing of android applications. In *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, ASE 2012, pages 258–261, New York, NY, USA. ACM.

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* Oxford University Press, USA.

AppBrain (2016). Number of available android applications. https://www.appbrain.com/stats/number-of-android-apps. Retrieved 2016-01.

Byrne, M. D., Anderson, J. R., Douglass, S., and Matessa, M. (1999). Eye tracking the visual search of click-down menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '99, pages 402–409, New York, NY, USA. ACM.

Choi, W., Necula, G., and Sen, K. (2013). Guided gui testing of android apps with minimal restart and approximate learning. *SIGPLAN Not.*, 48(10):623–640.

Cormen, T. H. (2009). *Introduction to algorithms*, chapter 22.3: Depth-first search, pages 540–549. MIT press.

Das, A. and Stuerzlinger, W. (2007). A cognitive simulation model for novice text entry on cell phone keypads. In *Proceedings of the 14th European Conference on Cognitive Ergonomics: Invent! Explore!*, ECCE '07, pages 141–147, New York, NY, USA. ACM.

Dörr, L.-M., Russwinkel, N., and Prezenski, S. (2016). Act-droid: Act-r interacting with android applications. In Reiter, D. and Ritter, F. E., editors, *Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016)*, pages 225–227, University Park, PA: Penn State.

Gallagher, M. A. and Byrne, M. D. (2015). Modeling password entry on a mobile device. In *Proceedings of the 2015 International Conference on Cognitive Modeling*.

Greene, K. K., Tamborello, F. P., and Micheals, R. J. (2013). Computational cognitive modeling of touch and gesture on mobile multitouch devices: Applications and challenges for existing theory. In *International Conference on Human-Computer Interaction*, pages 449–455. Springer.

Harrison, R., Flood, D., and Duce, D. (2013). Usability of mobile applications: literature review and rationale for a new usability model. *Journal of Interaction Science*, 1(1):1.

John, B. E., Prevas, K., Salvucci, D. D., and Koedinger, K. (2004). Predictive human performance modeling made easy. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 455–462. ACM.

Nielsen, J. (1994). Enhancing the explanatory power of usability heuristics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '94, pages 152–158, New York, NY, USA. ACM.

Prezenski, S. and Russwinkel, N. (2016). Towards a general model of repeated app usage. In Reiter, D. and Ritter, F. E., editors, *Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016)*, pages 201–207, University Park, PA: Penn State.

Raufaste, E. (2006). Air traffic control in act-r: A computational model of conflict detection between planes. In *Proceedings of the International Conference on Human-Computer Interaction in Aeronautics (HCI-Aero'06)*, pages 258–259.

Ritter, S., Anderson, J. R., Koedinger, K. R., and Corbett, A. (2007). Cognitive tutor: Applied research in mathematics education. *Psychonomic bulletin & review*, 14(2):249–255.

Teo, L. and John, B. E. (2008). Cogtool-explorer: towards a tool for predicting user interaction. In *CHI'08 Extended Abstracts on Human Factors in Computing Systems*, pages 2793–2798. ACM.