Novice2Expert - a Cognitive Model within a Usability Evaluation Framework

Karen Insa Wolf [*] **Rashik Thalappully** [*] **Yves Wagner** [**]
Frank Wallhoff [*,**] **Jens-E. Appell** [*]

[*] *Fraunhofer Institute for Digital Media Technology IDMT Marie-Curie-Str. 2, 26129 Oldenburg (e-mail: {insa.wolf, rashik.thalappully, jens.appell} @idmt.fraunhofer.de).*
[**] *Jade University of Applied Sciences (e-mail: {wager,wallhoff} @jade-hs.de)*

Abstract: The motivation of the usability evaluation framework CogUA (Cognitive Usability Analysis) is to support usability analysis processes of graphical user interfaces (GUI) based on software tools deriving objective and reproducible evaluation parameters in a partly automatic process. In the current prototype status, it consists of five modules: (i) Observer, (ii) Trace Illustrator, (iii) Screen Shot Analyser, (iv) Use Case Detector and (v) Predictor. The framework and its modules are shortly described in the paper. The focus of this paper lies on the fifth module, the Predictor. It is based on a cognitive model to simulate human users while interacting with a GUI. For the investigation, a test application is set up in which a label is presented as a task on the screen. The user, respectively the model, has to find and click the button with the presented label. The derived interaction times based on the model are compared with results of a small user study in order to evaluate the different versions of the cognitive model. Three different scenarios are investigated: (a) the model as a novice, searching each time for the correct button on the screen, (b) the model, which is able to remember the position of buttons of the GUI, and (c) the model, which is able to remember the position of buttons as well as the sequence of the tasks. The interaction time necessary to execute a specific task sequence is reduced from scenario (a) to (b) to (c). The relative reduction of interaction times derived from the user study and predicted by the model is comparable.

Keywords: automated usability analysis, user interaction simulation, cognitive modelling, ACT-R, computer vision

1. INTRODUCTION

The basic concept of CogUA was described in Wolf et al. (2016). Originally the work is motivated by the idea that software tools can support usability analysis especially in application areas where classical usability methods are rarely implemented. One such application area is the production of special purpose machines. Future users come commonly only in a very late phase in touch with the user interface. Additionally, the direct exchange between the developer of the user interface and the future user is limited. The CogUA software tools aim at providing relevant information for usability considerations. For this different areas of computer science are combined within the CogUA framework: computer vision, data mining and cognitive modelling. The framework has now been extended as described in the following paper.

2. USABILITY EVALUATION FRAMEWORK COGUA

In the current prototype status CogUA consists of five modules:

- Observer:
  The Observer records user interactions like mouse and keyboard events while interacting with an application. The application is selected by the user. Interactions linked to other applications are neglected by the Observer to respect privacy aspects. In case of a mouse click, a screenshot of the application is taken. The recorded events are stored in a human-readable log file, named as *traces*.
- Trace Illustrator
  This module is used to visualize the trace information time- and space-based. In Fig. 1 a time-based overview is shown with different interaction events. In the time-based overview, events can be selected to replay the interaction sequences as a film based on the screenshots including mouse movement and click information, as depicted in Fig. 2. One can jump from event to event to get a fast overview of the content of the session.
- Screen Shot Analyser
  This module is based on computer vision approaches to identify the graphical elements like button and text on the screen. This knowledge is applied in two different ways. Firstly, it is used to link a mouse click to a specific functional element of the GUI. Augmented traces including this additional information for the mouse events are generated. The second

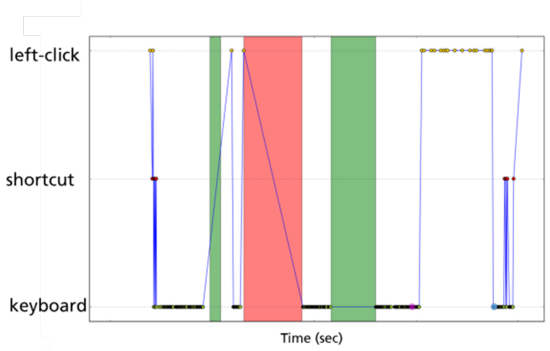

Fig. 1. A time based overview of a recorded interaction session. Green areas mark active breaks in which the selected application is active but no action from the user (especially no mouse movements), red areas mark passive breaks in which another application than the selected one is active.

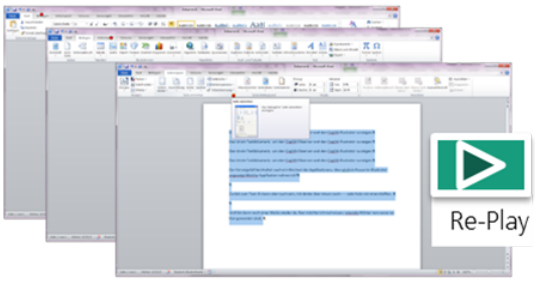

Fig. 2. Re-constructed interaction session based on trace information and screenshots recorded by the Observer, shown in the Trace Illustrator.

way is the derivation of quality parameters of the graphical layout considering the structure and layout of the functional elements (e. g. contrast, size) as very first rough usability check. The parameters are categorized into different classes as shown in Fig.3. Itention is to give a quick overview of the overall quality focussing on the probably most critical pages of the user interface.

- Use Case Detector
  Based on the augmented traces this module detects use cases as a repetitive pattern with the help of a sequential data mining approach, cf. Wolf et al. (2016).
- Predictor
  This module simulates user interactions based on a cognitive model to predict interaction times. It will be explained in more detail in the following section.

3. MODULE COGUA PREDICTOR

In this paper, the focus lies on the cognitive model part, the CogUA Predictor, which is evaluated based on a small user study. The application of a cognitive model in usability analysis is motivated by the full control of the setting, the behaviour of the subjects and the options to evaluate the detailed results, Ritter et al. (2001), West and Emond (2002), followed also in current approaches, e.g. Halbrügge (2013); Quade et al. (2014); Russwinkel and Prezenski (2014); Prezenski et al. (2017).

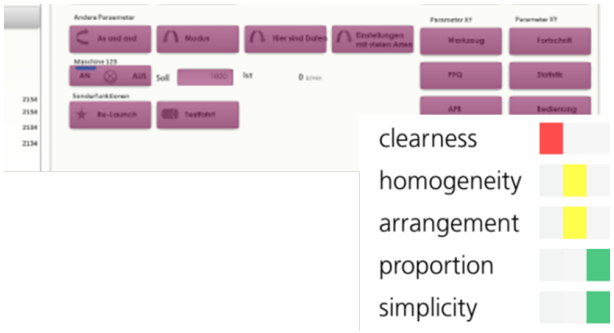

Fig. 3. Concept of the score based quality check as part of the Screen Shot Analyser. An extract of the GUI is shown in the background with detected functional elements marked in purple.

3.1 Concept

The use of a cognitive model to simulate user interactions linked to a user interface is at least in research a common practice, e.g. in gaming scenarios (Smart et al., 2016) or car driving scenarios (Kujala and Salvucci, 2015). A motivation for applying a cognitive model in usability analysis is given in Ritter et al. (2001) and West and Emond (2002). Summarizing, the usability analysis based on a virtual user can supplement a standard usability evaluation. The advantages are the full control of the settings (background of the subjects, the number of subjects, motivation validity, exclusion of an experimental bias e.g. by comments of the experimenter), and the drawbacks are the technical limitations of the cognitive model which is complex and time-consuming to implement.

In the next section, a short overview of existing approaches of automated usability analysis with a link to a cognitive model is given, followed by the description of main features of the tool CogUA, which is under development.

3.2 State of the art

Cognitive models aim at simulating processes of the human brain. Relevant processes for the mentioned application are the perception (vision), motor functions (moving a hand, fingers) and decision making using pieces of knowledge stored in memory. There are different known realizations of a cognitive model, as EPIC (Executive Process Interactive Control, Kieras and Meyer (1997)), SOAR (States, Operators And Reasoning, Laird et al. (1987)) and ACT-R (Adaptive Control of Thought-Rational, Anderson et al. (2004)).

Different approaches of automated usability analysis in the past 15 years also have a similar motivation as sketched above, e.g. Misker et al. (2001); Ritter et al. (2001, 2002); John et al. (2004); Heinath and Urbas (2007); Halbrügge (2013); Quade et al. (2014); Russwinkel and Prezenski (2014); Prezenski et al. (2017). Especially, the integration of a cognitive model into usability analysis should be simplified to allow a practical application in daily work of a software developer of user interfaces. A framework of templates of common user interactions (like mouse click, keyboard stroke) linked to the cognitive model ACT-R is proposed in Salvucci and Lee (2003). A similar idea is realized in Heinath and Urbas (2007). The purpose is that

the person who sets up the usability test does not require detailed knowledge of a cognitive model.

The CogTool (John et al., 2004) therefore provides a graphical user interface with the help of which a user can set up a visualization of the specific user interface he/she wants to analyze. Alternatively, Hyper Text Markup Language (HTML) code can be imported as the description of the user interface. Based on this visualization of the user interface a specific task consisting of different user interactions (mouse click, keyboard strokes) can be defined interactively with the CogTool GUI. Finally, the user runs the analysis based on the cognitive model. The outcome is a protocol of the predicted interactions including time stamps. The duration of interaction is a key criterion in the usability evaluation, cf. ISO 9241-210. The approach of Misker et al. (2001) goes one step further by avoiding the re-implementation of the user interface. On a Microsoft Windows system Misker et al. (2001) take the information about the user interface of an application from the window handlers. This information is transferred to the cognitive model. Actions, like a keyboard stroke, are returned from the cognitive model to the application. In this way, the cognitive model is directly linked to the user interface of the original application. The approach is evaluated for specifically defined use cases consisting of a sequence of interaction steps. A newer approach focussing on Android platform application is described in Prezenski et al. (2017). A crawler software is used to get information about the interaction elements of the selected application to derive in an automated way pre-knowledge for the cognitive model.

The intention of the last mentioned approaches is to minimize the workload to set up a cognitive model based test. The high workload due to the complexity of the model is one main obstacle for practical use of cognitive models. The CogUA framework follows this motivation to automate as much of the process as possible.

The approaches cited above, especially the one of Misker et al. (2001) and CogTool, John et al. (2004), provide a good basis, as no or only litte knowledge of a cognitive model is required. But in the case of CogTool, it is still necessary to re-implement the user interface (except for web pages coded in HTML). In both approaches, each specific use case must be defined step by step to set up the test scenario. This definition of the use case requires manual work by analyzing the user interface in detail, e.g. by applying the Hierarchical Task Analysis, cf. Heinath and Urbas (2007).

CogUA aims at the reduction of the effort of defining the use cases manually. The idea in CogUA is to monitor user interactions and to derive interaction traces. These traces can then be analyzed to identify use cases. Additionally, the approach of Misker et al. (2001) is extended by automated analysis of the graphics of the user interface. Thereby, the information of the GUI elements is not limited by the restricted content of the window handler.

The recording of user interaction traces is applicable if an implementation of the user interface already exists, either (a) as a prototype or (b) as running application, which should be modernized. In case (a), the steps of a use case can be defined by executing them on the prototype user interface. A manual pre-analysis of the workflow or expert

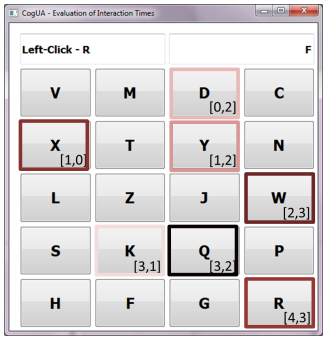

Fig. 4. Screen shot of one layout of the application for the user study. The button which are used repetitively are marked, in scenario Novice and Task in a fixed order which is colour coded from light to dark color.

knowledge is here necessary in order to select appropriate use cases. In case (b), the idea is that use cases are derived based on interaction traces recorded during the use of the software in practical work. The identified use cases are the basis to define the final use cases for the cognitive model. Additionally, the use cases can already be helpful for the design of a new version of the user interface with a better mapping of the workflow. Former workarounds, detected in the traces, can be re-designed as a standard process.

3.3 Architecture

The model is implemented based on the ACT-R architecture (Adaptive Control of Thought-Rational, Anderson et al. (2004)). A prototypic implementation of CogUA allows to link the model to any windows based application as the input information for the model is derived using the recording and subsequent computer vision analysis functionality. Within the user study comparison, a source code based link between the application and the model is realized to exchange in real-time visual and motoric information (mouse clicks). The GUI used in the user study is implemented in Python as shown in Fig. 4.

The framework consists of two parts and is based on the client-server architecture. The first part, the task environment which is developed in Python acts as the server. Task environment's duty is to define that tasks and communicate the tasks to the ACT-R model which in turn will perform the specified task.

The second part is an ACT-R module developed in Lisp in order to interact with the external environments. The implementation is similar to the module described in Hope (2013). This module acts as the client in the client-server model. The information exchange between client and server is happening over TCP-IP and is encoded in JSON Format.

3.4 Implementation of the model

The idea is to investigate the behaviour of a novice user in comparison to users who already know a specific interface. Therefore three different versions of the cognitive model are implemented:

- Model_Novice:
  The model is a novice, who has no information where

the correct button is. The model has to start a visual search to find the correct button.
- MODEL_BUTTON:
  This model starts as a novice but is able to remember the position of the buttons. As consequence, the interaction times are reduced when the same tasks occur as no visual search is necessary.
- MODEL_TASK:
  This model is able to remember the position of the buttons and additionally remembers the sequence of the tasks. This reduces the interaction time again as there is no need to check for the new task.

To keep the implementation and the evaluation simple in the first step no noise, no subsymbolic learning is activated for the model. As consequence the model behaves as "perfect learner", never forgetting something. It is clear that this does not simulate a human user but is chosen to prove easily the correct functionality of the first versions of the model.

Additionally, a simple visual search strategy for the model is implemented. It searches row-wise from upper left corner to lower right corner for the correct button. The search is based on visual search window of size 100 by 100 pixels.

4. SET UP OF THE USER STUDY

The intention for the user study is to evaluate the three first versions of the cognitive model. The tasks for the users is based on a very simple GUI application which is shown in Fig. 4. Similar to a calculator application 20 buttons are ordered in a five by four matrix. Above this button matrix, the user gets presented new tasks on left white area, on the right the result of the last click.

Three different scenarios are considered to match the different capabilities of the three versions of the model described above. The first scenario asks for a novice user. Looking at the model it is quite easy to generate a novice: it is just not able to remember something. But how to set up a user study avoiding that humans do not remember? The simplest way to realize this is to shuffle the labels of the buttons after every click. The user is then forced to search the correct button exactly the same way the model has to.

In the second scenario, the user should be able to remember the position of the button just like the model. Therefore, the layout is fixed, the labels of the buttons remain the same. But the tasks are presented in a random order.

Finally, in the third scenario, also the sequence of the tasks is fixed. This allows the user to remember it and to directly click on the next button without looking at the task window.

Based on this, the following cases for the user study were set up. All study cases consist of three runs with a sequence of click tasks which should be executed as concentrated as possible. After one run the user gets a break of at least five seconds. A subgroup of buttons is chosen as task sequence to make the recorded data comparable. The subgroup contains seven buttons as displayed in Fig. **??**. Within one run this task sequence is repeated.

- STUDY_NOVICE
  In this study, the selected task sequence is repeated three times, other - randomly chosen - buttons are introduced between the sequences to hide the repetitive pattern for the user. One run consists of 30 tasks.
- STUDY_BUTTON
  In this study, the selected task sequence is repeated four times in a shuffled order. One run consists of 28 tasks.
- STUDY_TASK
  In this study, the selected task sequence is repeated four times in a fixed order. One run consists of 28 tasks.

A number of $n = 12$ users executed the three different study cases as described above and the click times are recorded. The model was linked to the same application and the click times were recorded in the same way.

5. RESULTS OF THE USER STUDY

The reduction of interaction times from case STUDY_NOVICE to STUDY_BUTTON and to STUDY_TASK can be seen in the user data, as expected. The resulting interaction times for each of the three runs for three selected users and the model is shown in Fig. 5.

The difference between the three users is large. User-2 is a candidate with very few practical experience in computer interaction. User-3 matches very well the predicted time of the model. The results of the studies STUDY_BUTTON and STUDY_TASK show a plausible reduction of interaction time from run to run.

This reduction of interaction time is investigated based on the overall time necessary for completing one task sequence with seven buttons to be clicked. In study STUDY_NOVICE, this is done three times per run, so in total nine times. For the two other study cases, this is completed 12 times. Fig. 6 shows mean values of these times average over all users in comparison to the results of the model.

The range of the interaction time is approximately the same, the pattern is similar. Due to the fixed visual search strategy, the interaction times for the model are constant in the study STUDY_NOVICE. And as described above the model behaves in the current version as a perfect learner. Consequently, the effect of reduction in interaction times occurs directly for the second sequence. Thereafter the time remains constant, with slight differences in study case STUDY_BUTTON due to the different order of the tasks in each task sequence. The differences can be explained by Fitts' law based on the differences in the click paths.

Considering the standard deviations the mean values based on the user data are compared with the predicted times from the model. The standard deviation is marked in black color if the difference between mean value of user data and the model prediction is non-significant, otherwise in red. In case STUDY_BUTTON the last two and in case STUDY_TASK the last three sequences of the model agrees with the data from the user study following a t-test with $\alpha = 5\%$.

What about the relative reduction of interaction time? We compare the reduction in time based on the last sequence

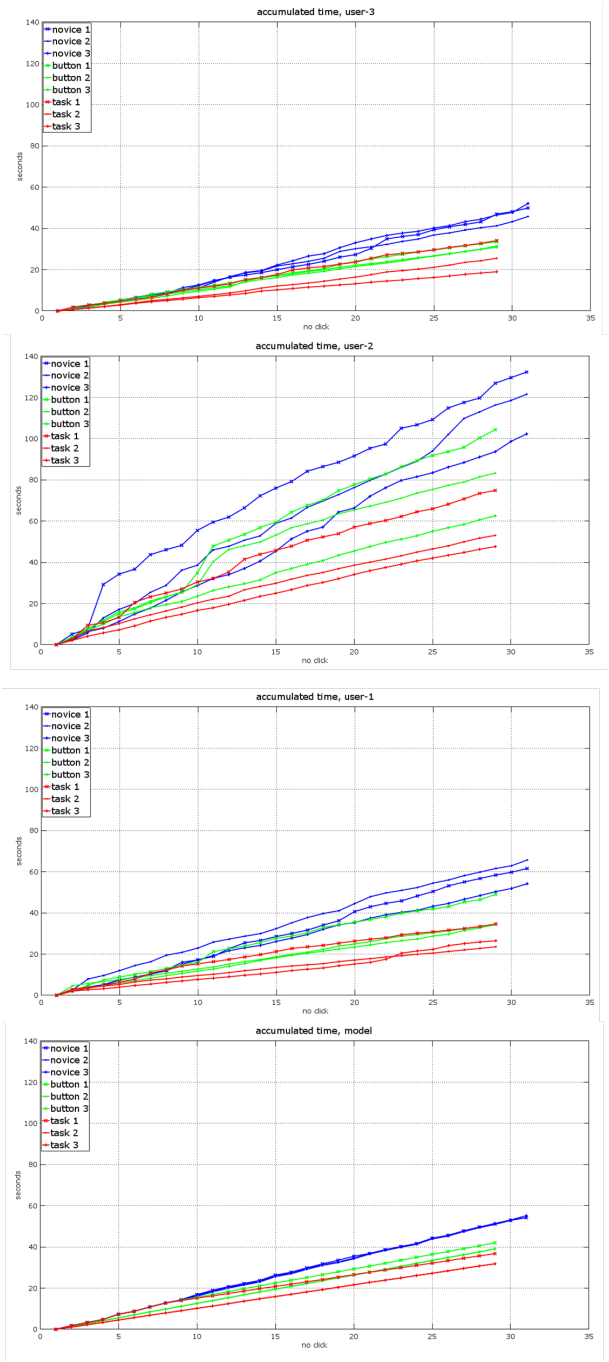

Fig. 5. Accumulated interaction time necessary to complete the task in each of the three runs in all three study cases. Overall time is reduced from Study_Novice to Study_Button to Study_Task as expected.

in Study_Button and Study_Task in comparison to the weighted mean of all sequences in Study_Novice. This results for the mean values of the users in a time improvement of 32% in case of Study_Button and 45% in case of Study_Task. The model predicts an improvement of 23% and 38%, respectively. Considering the propagated standard deviation the differences between model and user are not significant. Of course, one has to admit that the variance between the different users are relatively high, the standard deviation of a single time

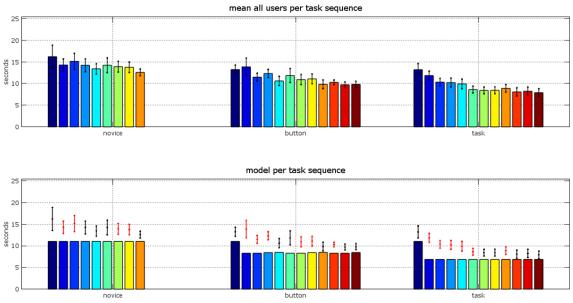

Fig. 6. Mean time based on all users ($n = 12$) necessary to complete one task sequence of length seven. Study_Button and Study_Task show a reduction of interaction time as the positions of the buttons or even the task sequence can be remembered. As the model is implemented as "perfect learner" the reduction occurs directly for the second sequence and remains constant in the following sequences. The vertical lines indicate the standard deviation of the mean values of the user data. The lines are marked red if the difference between the time predicted by the model is significantly different to those of the users.

value for one task sequence range from 2 to 9 seconds. Especially in the Study_Novice outliers occur when a user misses several times the correct button in the visual search.

6. CONCLUSIONS AND OUTLOOK

In this paper, results of the simulation of user interactions based on a cognitive model are shown as next development step of the CogUA framework. Interaction times of different simulation scenarios for the cognitive model compared to times recorded in user studies are discussed. The variance within the user data is high, especially under the condition of a novice, forced to search for the correct button. Three different scenarios are investigated with different memory capabilities of the model. The user study is designed in the way that the human users are limited to the same memory capabilities as the model. The more the model and the user are able to remember (button position, task sequence) the smaller are the necessary interaction times for a specific task, as expected. The relative reduction of interaction times of the model agrees statistically with the user data. Overall shows the cognitive model in the current version too low interaction times, it is too fast.

The described user study was used to check the principle plausibility of the model implementation. The simple these versions of the model are, they could be used as reference basis for comparison of different user interfaces in a relativ manner. Such a virtual user can supplement a standard usability evaluation by providing reproducible and tangible usability results that can easily be compared. A remaining disadvantage of this kind of cognitive model is the missing context knowledge.

REFERENCES

Anderson, J.R., Bothell, D., Byrne, M.D., Douglass, S., Lebiere, C., and Qin, Y. (2004). An integrated theory of the mind. *Psychological review*, 111(4), 1036–1060.

Halbrügge, M. (2013). ACT-CV: Bridging the gap between cognitive models and the outer world. In E. Brandenburg, L. Doria, A. Gross, T. Günzlera, and H. Smieszek (eds.), *Grundlagen und Anwendungen der Mensch-Maschine-Interaktion - 10. Berliner Werkstatt Mensch-Maschine-Systeme*, 205–210. Universitätsverlag der TU Berlin.

Heinath, M. and Urbas, L. (2007). Simplifying the development of cognitive models using pattern-based modeling. In *10th IFAC/IFIP/IFORS/IEA Symposium Analysis, Design, and Evaluation of Human-Machine Systems. Seoul, Korea.*, 130–135.

Hope, R.M. (2013). The JSON network interface programmers guide. Technical report, Rensselaer Polytechnic Institute. https://github.com/RyanHope/json-network-interface/tree/master/documentation, 15.6.2018.

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*, 455–462. ACM.

Kieras, D.E. and 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.

Kujala, T. and Salvucci, D.D. (2015). Modeling visual sampling on in-car displays: The challenge of predicting safety-critical lapses of control. *International Journal of Human-Computer Studies*, 79, 66–78.

Laird, J.E., Newell, A., and Rosenbloom, P.S. (1987). Soar: An architecture for general intelligence. *Artificial intelligence*, 33(1), 1–64.

Misker, J., Taatgen, N.A., and Aasman, J. (2001). Validating a tool for simulating user interaction. In *Proceedings of the Fourth International Conference on Cognitive Modeling*, 163–168.

Prezenski, S., Bruechner, D., and Russwinkel, N. (2017). Predictive cognitive modemodel 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)*, 165–171.

Quade, M., Halbrügge, M., Engelbrecht, K.P., Albayrak, S., and Möller, S. (2014). Predicting task execution times by deriving enhanced cognitive models from user interface development models. In *Proc EICS 2014*, 139–148.

Ritter, F.E., Baxter, G.D., Jones, G., and Young, R.M. (2001). User interface evaluation: How cognitive models can help. *Human-computer interaction in the new millennium*, 125–147.

Ritter, F.E., Van Rooy, D., and 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*, 111–118. Springer.

Russwinkel, N. and Prezenski, S. (2014). ACT-R meets usability. In *Proc 6th International Conference on Advanced Cognitive Technologies and Applications COGNITIVE, 2014. Venice, Italy: IARIA.*

Salvucci, D.D. and Lee, F.J. (2003). Simple cognitive modeling in a complex cognitive architecture. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, 265–272. ACM.

Smart, P.R., Scutt, T., Sycara, K., and Shadbolt, N. (2016). Integrating ACT-R cognitive models with the unity game engine. *Integrating Cognitive Architectures into Virtual Character Design. IGI Global, Hershey, Pennsylvania, USA.*

West, R.L. and Emond, B. (2002). Can cognitive modeling improve rapid prototyping. *Carleton University Cognitive Science Technical Report*, 5.

Wolf, K.I., Thalappully, R., Goetze, S., and Wallhoff, F. (2016). Concept of automated usability evaluation of graphical user interfaces. In *Proc 5. Interdisziplinärer Workshop Kognitive Systeme, 14.-16.3.2016, Bochum.*