Halbrügge, M., Quade, M. & Engelbrecht, K.-P. (2015). A Predictive Model of Human Error based on User Interface Development Models and a Cognitive Architecture. In Taatgen, N. A., van Vugt, M. K., Borst, J. P. & Mehlhorn, K. (Eds.), *Proceedings of the 13th International Conference on Cognitive Modeling* (pp. 238-243). Groningen, the Netherlands: University of Groningen.

A Predictive Model of Human Error based on User Interface Development Models and a Cognitive Architecture

Marc Halbrügge (marc.halbruegge@tu-berlin.de)
Quality and Usability Lab, Telekom Innovation Laboratories
Technische Universität Berlin, Ernst-Reuter-Platz 7, 10587 Berlin

Michael Quade (michael.quade@dai-labor.de)
DAI-Labor, Technische Universität Berlin
Ernst-Reuter-Platz 7, 10587 Berlin

Klaus-Peter Engelbrecht (klaus-peter.engelbrecht@telekom.de)
Quality and Usability Lab, Telekom Innovation Laboratories
Technische Universität Berlin, Ernst-Reuter-Platz 7, 10587 Berlin

Abstract

The concept of device- vs. task-orientation allows to identify subtasks that are especially prone to errors. Device-oriented tasks occur whenever a user interface requires additional steps that do not directly contribute to the users' goals. They comprise, but are not limited to, initialization errors and post-completion errors (e.g., removing a bank card after having received money). The vulnerability of device-oriented tasks is often counteracted by making them obligatory (e.g., by not handing out the money before the bank card has been removed), making it even harder to predict where users will have problems with a given interface without dedicated user tests. In this paper we show how cognitive modeling can be used to predict error rates of device-oriented and task-oriented subtasks with respect to a given application logic. The process is facilitated by exploiting user interface meta information from model-based user interface development.

Keywords: Human Error; Cognitive Modeling; Model-based User Interface Development; Memory for Goals; ACT-R

Introduction

Our everyday life is dominated by routine activities, i.e., well-learned, rule-based tasks like making coffee or buying a train ticket. And even though we have performed them hundreds of times, we are still making occasional errors during their execution (Reason, 1990). While the base error rate for routine tasks is low, some procedural steps are more problematic than others. The best-known examples are post-completion errors, when users fail to perform an additional step in a procedure *after* they have already reached their main goal (Byrne & Davis, 2006; Trafton, Altmann, & Ratwani, 2011; Ratwani & Trafton, 2011). This concept can be expanded to any step that doesn't directly contribute to the users' goals, but is imposed on them by the system and has been coined *device-orientation* (Ament, Blandford, & Cox, 2009; Gray, 2000). The opposite, i.e., steps that are noticeably related to the users' goals, is called *task-orientation* in contrast.

Jef Raskin popularized the complaint that "a dialog box that has no choices (e.g., you can only press ENTER before you can do any other task) has a productivity of 0" (Raskin, 1997), because the user cannot transfer any knowledge to the system using it. Raskin's information theoretic concept fits nicely with our understanding of device-oriented tasks: they do not convey any information that is specific to the user's current goal.

Even when human interface designers try to avoid device-oriented tasks in the first place, software or hardware constraints do not allow the removal of every single one. As long as bank cards are used, they need to be placed into some kind of slot and removed later on. One popular design strategy in this situation is to make device-oriented tasks *obligatory*, i.e., their position in the action sequence is pulled forward so that the user's goal cannot be achieved without performing the device-oriented task. But how does this change affect error rates?

The objective of this paper is to shed some light on the impact of device-orientation and subtask necessity[1] on user errors. We derive a computational user model from the memory for goals theory (MFG, Altmann & Trafton, 2002) and apply it to a class of user interfaces (UI) that have been created using the model-based UI development process (MBUID, Vanderdonckt, 2008; Calvary et al., 2003). As a by-product of this process, such interfaces provide meta-information about their elements (e.g., buttons) that can be used for the creation of more generic cognitive models (Quade, Halbrügge, Engelbrecht, Albayrak, & Möller, 2014).

Action Control and Human Error

According to Rasmussen (1983), human action control can be described on three levels: skill-, rule-, and knowledge-based behavior. Skill-based behavior on Rasmussen's lowest level is generated from highly automated sensory-motor actions without conscious control. Knowledge-based behavior on the other hand is characterized by explicit planning and problem solving in unknown environments. In between the skill and the knowledge levels is rule-based behavior. While being goal-directed, rule-based actions do not need explicit planning or conscious processing of the current goal. The stream of actions follows stored procedures and rules that have been developed during earlier encounters or through instruction.

[1] With respect to user tasks, we are using the terms *obligatory*, *mandatory* and *necessary* synonymously in the course of this paper.

Halbrügge, M., Quade, M. & Engelbrecht, K.-P. (2015). A Predictive Model of Human Error based on User Interface Development Models and a Cognitive Architecture. In Taatgen, N. A., van Vugt, M. K., Borst, J. P. & Mehlhorn, K. (Eds.), *Proceedings of the 13th International Conference on Cognitive Modeling* (pp. 238-243). Groningen, the Netherlands: University of Groningen.

Interacting with computer systems after having received some training is predominantly located on Rasmussen's rule-based level, with contributions of the skills level to a lesser degree. We are therefore concentrating our modeling effort on rule-based behavior. On this this level of action control, human error is characterized by deviation from the stored procedure, i.e., either leaving out a step or adding an additional unnecessary one.

Model Based User Interface Development (MBUID)

The main goals of model-based user interface development are the reduction of complexity and an increased reusability of existing patterns and solutions during UI development (Vanderdonckt, 2008). For this purpose, there exist frameworks and description languages that conform to the CAMELEON reference framework (Calvary et al., 2003) for structuring the processes and information of MBUID. While *abstract UI* (AUI) models describe UI elements on a modality-independent level, the more *concrete UI* models hold modality-specific information and the *final UI* models represent specific implementations for different platforms. Besides these models there exist conceptual models related to the domain of the application and the task model for describing the interaction logic.

A widely used approach for modeling tasks in MBUID is the Concurrent Task Tree (CTT) notation (Paternò, 1999) which also allows executing tasks in conjunction with UI models (Mori, Paternò, & Santoro, 2004) by establishing mappings between these models. CTT comes with several operators for grouping tasks into hierarchical structures, defining temporal relationships and for describing information flow between user, application and tasks. Namely *interaction tasks* describe tasks that user and application perform together, while *application tasks* relate to actions by the system.

By combining these different UI development models, information about the application becomes available that goes far beyond what is visible to the user. In the context of this paper, the task information contained within the UI development models is especially interesting, as it corresponds directly to Rasmussen's rule-based behavior level.

Experiment

The scarcity (yet pervasiveness) of procedural errors during routine tasks (Reason, 1990) makes statistical analysis difficult. Researchers have used secondary tasks (e.g., Byrne & Davis, 2006; Ruh, Cooper, & Mareschal, 2010) or interruptions (e.g., Li, Blandford, Cairns, & Young, 2008; Altmann, Trafton, & Hambrick, 2014) to increase error rates. We rejected both options for reasons of ecological validity, and chose repeated measures and a medium sized sample instead.

The experiment focuses on procedural errors during the usage of a kitchen assistance system for ambient assisted living. The kitchen assistant helps preparing a meal for a given number of persons with its searchable recipe library, adapted shopping list generator, and by providing interactive cooking or baking instructions.

Ambient systems like the one used here are characterized by interconnecting a multitude of physical devices. We selected among these a personal computer with large touch screen and a tablet computer for the experiment. To match the characteristics of each device, we created two versions of the user interface of the kitchen assistant. The first version is a tablet-oriented, *simple* one, optimized for portrait mode, with larger buttons and fonts, and rather few elements per screen. The other UI version is more *complex*, using smaller buttons and fonts so that more elements fit on the screen. Some screens of the simple UI are shown side-by-side in the complex version, thereby reducing the necessity of navigation between screens. The complex UI targets the large personal computer and is optimized for landscape mode. Annotated screenshots of the two UI versions are shown in Figure 1.

Analysis of the UI Development Models

Because we are working with an already existing application, we cannot freely manipulate whether a UI element and its related subtask is device- or task-oriented. But we can use the MBUID models to derive this property from them in a generalizable way. By analyzing the operators of the task model and the types of AUI elements it is possible to identify device- and task-oriented subtasks and make these explicit to the evaluator or modeler.

On the one hand, task-oriented interaction steps are characterized by interaction on UI elements that are modeled on the level of the AUI model as *FreeInput* or *Choice*. These elements are generally used to provide task-oriented information from the user to the application using *interaction tasks*, e.g., by using text fields, radio buttons or checkboxes. On the other hand, device-oriented interaction steps describe actions which let the dialogue proceed to a further step, e.g., when executing buttons labeled "Next" or "Done". The abstract type *Command* denotes such UI elements on the level of the AUI model which can be used to trigger application tasks.

In order to identify device-oriented subtasks, the models have to be checked for *interaction tasks* that are modeled using *Command* elements which then *enable* application tasks. The process is visualized in Figure 1. In the case of the kitchen assistant, device-oriented elements are buttons that lead to the next screen or modify entries from the ingredients list. Task-oriented UI elements are buttons for toggling search attributes and selections in search results lists.

Method

Participants Twenty members of the Technische Universität Berlin paid participant pool took part in the experiment. There were 5 men and 15 women, with an age range from 18 to 59 (M=32.3, SD=11.9). As the instructions were given in German, only fluent German speakers were allowed to take part.

Materials The experiment was conducted in our fully equipped lab kitchen. A personal computer with 27" (68.6 cm, landscape mode) touch screen and a 10" (25.7 cm, por-

Halbrügge, M., Quade, M. & Engelbrecht, K.-P. (2015). A Predictive Model of Human Error based on User Interface Development Models and a Cognitive Architecture. In Taatgen, N. A., van Vugt, M. K., Borst, J. P. & Mehlhorn, K. (Eds.), *Proceedings of the 13th International Conference on Cognitive Modeling* (pp. 238-243). Groningen, the Netherlands: University of Groningen.

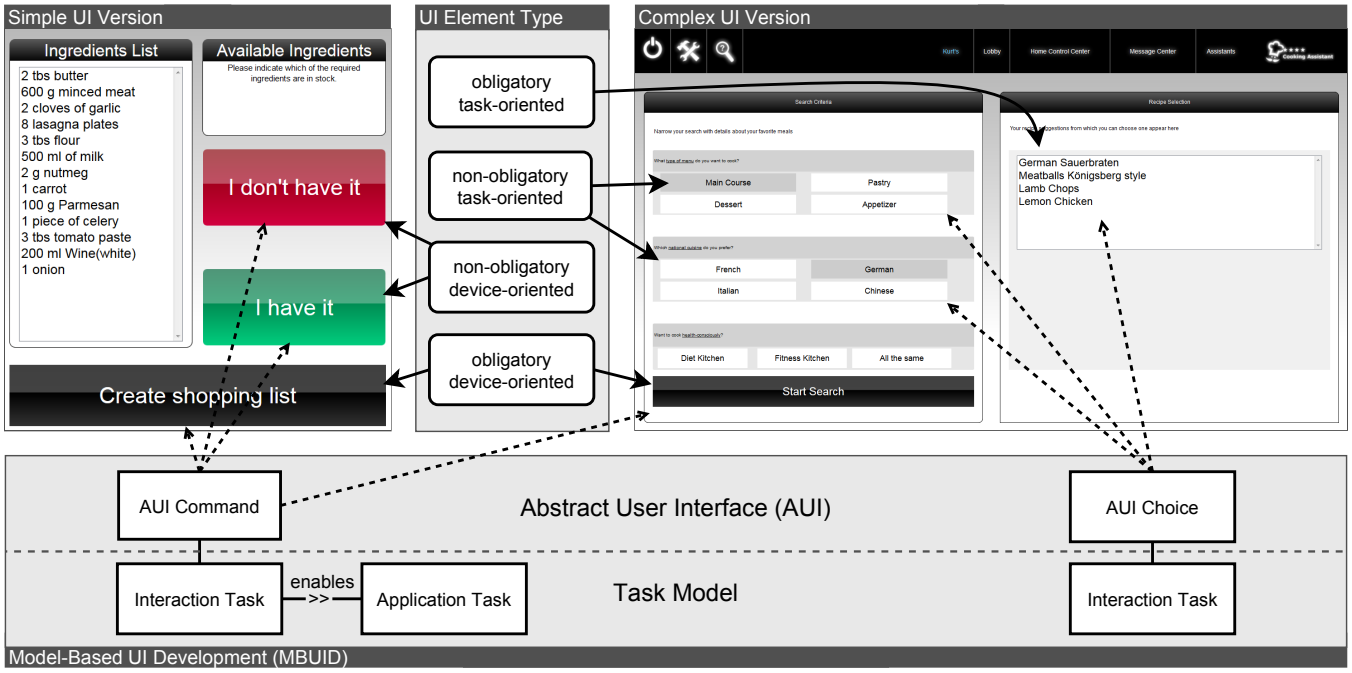

Figure 1: Screenshots of the kitchen assistant. Ingredients list of the simple UI on the left, recipe search of the complex UI with two screens side-by-side on the right. UI element type is indicated by solid arrows, AUI type indicated by dashed arrows.

trait mode) tablet, both placed near the sink, were used to display the interface of the kitchen assistant. All user actions were recorded by the computer system. The subjects' performance was additionally recorded on videotape for subsequent error identification.

Design We applied a four-factor within-subjects design, the factors being UI version (simple vs. complex), physical device (screen vs. tablet), device- vs. task-orientation, and task necessity (non-obligatory vs. oligatory). Every subject went through all four combinations of UI version and physical device in randomized order. User tasks were analogously grouped into four blocks of eleven to twelve individual tasks. Block orders were counterbalanced across participants as well. We call the combination of the device-orientation and task necessity factors *UI element type* in the following. While device-orientation can be derived from the MBUID models (see above), task necessity is only implicitly represented in the assistant's interaction logic. Mandatory subtasks are related to all elements that lead to the next screen or unhide buttons on the current screen, the latter being the case for the selection of a recipe in the search result list (see Figure 1).

Procedure Every block started with comparatively easy recipe search tasks, e.g., "search for German main dishes and select lamb chops".[2] Users would then have to change the search attributes, e.g., "change the dish from appetizer to dessert and select baked apples". The second half of each block was made of more complex tasks that were spread over more screens of the interface and/or needed memorizing more items. The subjects either had to create ingredients lists for a given number of servings, or had to make shopping lists using a subset of the ingredients list, e.g., without salt and flour. All instructions were read to the subjects by the experimenter. Every individual trial was closed by a simple question the subjects had to answer to keep them focused on the kitchen setting, e.g., "how long does the preparation take?" With an initial training phase and exit questions the whole procedure took less than an hour.

Results

We identified errors by comparing the observed click sequences with optimal ones. Whenever a step of the optimal sequence was missing, we recorded this as an *omission*. Unnecessary additional steps performed by the subjects were analogously recorded as *intrusions*. A special case of intrusions are *perseverations*, when an action is erroneously repeated. The resulting taxonomy follows Ruh et al. (2010) and is deliberately using phenotypic descriptions instead of genotypic explanations (in contrast to, e.g., Norman, 1988).

In total, 6359 clicks were observed. 104 (1.6%) of these were classified as user errors. 56 were omissions, 38 were intrusions, and 10 were perseveration errors. The perseveration errors were bound to a button for increasing the number of servings that needed several consecutive presses. As the video recordings clearly showed that all perseverations were not caused by the users, but by occasional excesses of UI lag, they were removed from further analysis.

[2]We give English translations of the actual instructions here for reasons of comprehensibility. The original instructions are available for download at http://www.tu-berlin.de/?id=135088

Halbrügge, M., Quade, M. & Engelbrecht, K.-P. (2015). A Predictive Model of Human Error based on User Interface Development Models and a Cognitive Architecture. In Taatgen, N. A., van Vugt, M. K., Borst, J. P. & Mehlhorn, K. (Eds.), *Proceedings of the 13th International Conference on Cognitive Modeling* (pp. 238-243). Groningen, the Netherlands: University of Groningen.

Due to the scarcity of errors and the use of repeated measures, χ^2-based significance tests could not be used (Agresti, 2014). The error rates were instead evaluated using a mixed logit model with subject as random effect (Bates, Maechler, Bolker, & Walker, 2014). They do not vary with device, UI version, or task necessity (all $p > .3$). The main effect of device-orientation points to the expected direction with device-oriented subtasks showing higher error rates, but narrowly missed significance ($z = -1.82, p = .069$). We found a significant interaction between necessity and device-orientation ($z = 4.07, p < .001$). Error rates and 95%-confidence intervals are given in Figure 2.

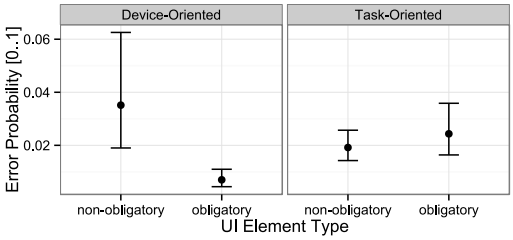

Figure 2: Error probabilities for different UI elements.

Cognitive User Model

The theoretical foundation for our model is the memory for goals theory (MFG, Altmann & Trafton, 2002). The MFG postulates that goals and subgoals are subject to the characteristics of human memory traces, in particular time-dependent and noisy *activation*, *interference*, and associative *priming*. Lack of activation of a subgoal, possibly connected to little priming, can cause omissions, while interference with other subgoals can result in intrusions. While the MFG theory initially has been validated on the basis of Tower-of-Hanoi experiments, i.e., rather artificial problem-solving tasks in the laboratory, it has been shown to generalize well to sequence errors during software use and has been extensively used in the human-computer interaction domain (e.g., Li et al., 2008; Trafton et al., 2011).

The cognitive user model presented here has been developed using the cognitive architecture ACT-R (Anderson et al., 2004). As shown in previous research, associative priming can be considered an acceptable explanation of *temporal* disadvantages of device-oriented steps (Halbrügge & Engelbrecht, 2014). The work presented here provides a direct link from device-orientation to user errors.

This was achieved by using two additional concepts: firstly, *partial matching* mimics human memory imperfections by responding to memory retrievals with similar, but not completely fitting chunks of information. And secondly, humans as embodied and situated beings tend to use environmental cues to reduce cognitive load (Wilson, 2002). This led to the addition of a *knowledge-in-the-world* (Norman, 1988) strategy where the user scans the UI for "inviting" elements instead of relying on the internal representation of the current task, only. This attempt also goes in line with Salvucci's (2010) criticism of the MFG theory being too focused on memory while neglecting the user's interaction with the environment.

Technically, the model is run inside the standard Lisp distribution of ACT-R 6.0.[3] The ability to interact with the HTML interface of the kitchen assistant is provided by ACT-CV (Halbrügge, 2013). The model receives its instructions through ACT-R's auditory system and tries to memorize the necessary steps for the current trial. No specific knowledge about the kitchen assistant is hard-coded into the model. When pursuing a goal, the model first uses a knowledge-in-the-head strategy, i.e., it tries to follow the memorized step sequence (left part of Figure 3). Once memory gets weak, the knowledge-in-the-world strategy takes over. Elements on the screen are randomly attended and a memory recall heuristic is used to determine whether this element was part of the current action sequence. If no matching goal chunk is found, the visual search for possible targets is continued (right part of Figure 3).

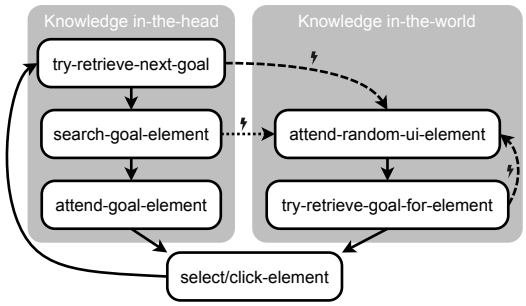

Figure 3: Schematic flow chart of the knowledge-in-the-head and knowledge-in-the-world strategies of the cognitive model. Dashed arrows denote retrieval failure, dotted arrows denote visual search failure.

How does the Model Predict Errors?

Memory activation (and its noise) is the main explanatory construct used by the model. Omissions are caused by the activation of the respective subgoal being too low. As activation decays over time, omissions are more likely for longer task sequences and for subgoals that appear late in a sequence. Task-oriented elements of a sequence receive additional activation through priming (Halbrügge & Engelbrecht, 2014) and are therefore in principle less prone to omissions. This effect can be mitigated by the application logic, though. If a subgoal can no longer be retrieved by the knowledge-in-the-head strategy, its corresponding UI element can nevertheless be found by the knowledge-in-the-world strategy (see Figure 3). This is especially probable for mandatory subtasks like navigating to the following screen because these mark

[3]The Lisp source code of the model is available for download at http://www.tu-berlin.de/?id=135088

Halbrügge, M., Quade, M. & Engelbrecht, K.-P. (2015). A Predictive Model of Human Error based on User Interface Development Models and a Cognitive Architecture. In Taatgen, N. A., van Vugt, M. K., Borst, J. P. & Mehlhorn, K. (Eds.), *Proceedings of the 13th International Conference on Cognitive Modeling* (pp. 238-243). Groningen, the Netherlands: University of Groningen.

situations where no other applicable UI element can be found by the model.

Intrusions happen when the activation of a similar subgoal of a previous trial exceeds the activation of the current subgoal. The partial matching mechanism adds an additional penalty to the intruding subgoal's activation depending on its dissimilarity to the retrieval request.[4] Activation noise is essential for intrusions, but they can also be caused by an old subgoal receiving "misguided" priming from the current goal, e.g., when there is substantial overlap between two consecutive trials. Because task-oriented subgoals receive more priming than device-oriented ones, the model predicts higher intrusion rates for task-oriented UI elements.

Another cause for intrusions is the knowledge-in-the-world strategy. ACT-R's activation spreading mechanism allows priming from the current focus of the visual system to declarative memory elements, comparable to the horizontal dimension of the "Dual Systems" theory (Cooper & Shallice, 2000). When the model searches the screen for "inviting" elements (*attend-random-ui-element* in Figure 3), the currently attended element primes subgoals that correspond to it regardless of whether they belong to the current goal or a previous one.

Model Fit

The model predictions are sensitive to several global ACT-R parameters that affect activation. We kept activation decay (bll) at the default of .5, varied activation noise (ans) between .5 and .6, set priming (mas) to 3.5 and varied the partial matching penalty (mp) between 4.0 and 4.5.

The model was run 1000 times and all errors made were collected (dotted lines in Figure 4). The quantitative fit as computed based on the error probability for each combination of omissions and intrusions and the four UI element types defined above is promising with $R^2 = .915$ and RMSE = .003.

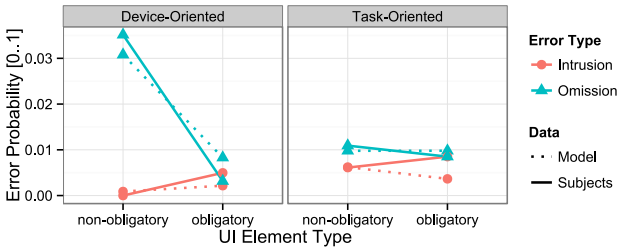

Figure 4: Empirical error probabilities and model predictions for different UI elements.

Discussion

We conducted a usability study to examine the effects of device-orientation and task necessity on procedural errors. Unsurprisingly, mandatory tasks were less likely to be omitted. They showed slightly higher intrusion rates than their non-obligatory counterparts, though. Screen elements that were both device-oriented and non-obligatory showed by far the highest error, i.e., omission rates.

These results are valuable in themselves for several reasons. First, they show that human error can be studied well without adding secondary tasks or interrupting the subjects during their tasks. Secondly, the concept of device- vs. task-orientation has proven beneficial in principle. And finally, our results highlight how little we can predict when we base our predictions on nothing but theoretical concepts (here device-orientation). Only when we take the application's interaction logic into account (here focused on obligatory vs. non-obligatory task steps) we get significant differences that are worth further analysis.

At first thought, this might lead into a problem: The application logic is specific to a single application most of the time. If we base our analysis on it, we are in danger of limiting the scope of our research to nothing but that single application. The choice of cognitive modeling as a methodology helps resolving this issue. Cognitive models can integrate other knowledge as long as it is machine readable. In our case, the meta-information encoded within the MBUID user interface models provides the possibility to model the activation gains of task-oriented subgoals in a generalizable way. And even more important, the influencing factor of task necessity is only implicitly represented within the application. The interaction effect of device-orientation and task necessity is an emergent phenomenon that is only elicited during the execution of the cognitive model, i.e., by performing tasks using the actual application.

We created a cognitive model based on the memory for goals theory, integrated the MBUID information, and added a knowledge-in-the-world strategy that the model applies after facing memory retrieval failure. While the overall goodness-of-fit of the model is good and especially the case of non-obligatory device-oriented tasks showing the highest error rate is reproduced well (see Figure 4), there are several limitations. The sheer number of mechanisms used (decay, priming, etc.) leads to a complex cognitive model that is rather sensitive to changes of the respective ACT-R parameters. Future research will show whether this affects the generalizability of the model. The same holds for the empirical basis of the model and the representativity of the tasks used during the experiment.

The model nevertheless provides several improvements when compared to existing MFG models of procedural error. Most important, while Altmann and Trafton (2002) discuss how the environment can provide cues that prime pending goals, our model is the first one that actively uses this strategy. As a by-product, this leads to the prediction of intrusions, an important error class that is often not well captured (e.g., Trafton et al., 2011; Li et al., 2008).

[4]The dissimilarities are computed by ACT-R based on the number of mismatching information units, here trial and current subgoal. No user-specified similarity function is used by this model.

Halbrügge, M., Quade, M. & Engelbrecht, K.-P. (2015). A Predictive Model of Human Error based on User Interface Development Models and a Cognitive Architecture. In Taatgen, N. A., van Vugt, M. K., Borst, J. P. & Mehlhorn, K. (Eds.), *Proceedings of the 13th International Conference on Cognitive Modeling* (pp. 238-243). Groningen, the Netherlands: University of Groningen.

Concluding Remarks

As of today, the body of theory and empirical research on human error is growing, but validated methods for predicting user errors exist only for restricted areas (e.g., Ratwani & Trafton, 2011). We present a computational user model grounded in cognitive science research that aims at more general error predictions. Taking advantage of UI meta information collected during model-based development of different user interfaces, our cognitive user model can well reproduce the findings from a usability study conducted before.

In the future, we plan to apply the model to new and different interfaces and develop the connection to MBUID towards higher automation. This way, interface designers could receive error predictions at early stages of the development cycle, making error prevention much easier.

Acknowledgements

We gratefully acknowledge financial support from the German Research Foundation (DFG) for the project "Automatische Usability-Evaluierung modellbasierter Interaktionssysteme für Ambient Assisted Living" (AL-561/13-1).

References

Agresti, A. (2014). *Categorical data analysis*. John Wiley & Sons.

Altmann, E. M., & Trafton, J. G. (2002). Memory for goals: An activation-based model. *Cognitive science*, *26*(1), 39–83.

Altmann, E. M., Trafton, J. G., & Hambrick, D. Z. (2014). Momentary interruptions can derail the train of thought. *Journal of Experimental Psychology: General*, *143*(1), 215–226.

Ament, M. G., Blandford, A., & Cox, A. L. (2009). Different cognitive mechanisms account for different types of procedural steps. In *Proc. COGSCI '09* (pp. 2170–2175).

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

Bates, D., Maechler, M., Bolker, B., & Walker, S. (2014). lme4: Linear mixed-effects models using Eigen and S4 [Computer software manual]. (R package version 1.1-7)

Byrne, M. D., & Davis, E. M. (2006). Task structure and postcompletion error in the execution of a routine procedure. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *48*(4), 627–638.

Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., & Vanderdonckt, J. (2003). A unifying reference framework for multi-target user interfaces. *Interacting with Computers*, *15*(3), 289–308.

Cooper, R., & Shallice, T. (2000). Contention scheduling and the control of routine activities. *Cognitive Neuropsychology*, *17*(4), 297–338.

Gray, W. D. (2000). The nature and processing of errors in interactive behavior. *Cognitive Science*, *24*(2), 205-248.

Halbrügge, M. (2013). ACT-CV: Bridging the gap between cognitive models and the outer world. In E. Brandenburg, L. Doria, A. Gross, T. Günzlera, & H. Smieszek (Eds.), *Grundlagen und anwendungen der mensch-maschine-interaktion* (pp. 205–210). Berlin: Universitätsverlag der TU Berlin.

Halbrügge, M., & Engelbrecht, K.-P. (2014). An activation-based model of execution delays of specific task steps. *Cognitive Processing*, *15*, S107-S110.

Li, S. Y., Blandford, A., Cairns, P., & Young, R. M. (2008). The effect of interruptions on postcompletion and other procedural errors: An account based on the activation-based goal memory model. *Journal of Experimental Psychology: Applied*, *14*(4), 314.

Mori, G., Paternò, F., & Santoro, C. (2004). Design and development of multidevice user interfaces through multiple logical descriptions. *IEEE Trans. Softw. Eng.*, *30*(8), 507–520.

Norman, D. A. (1988). *The psychology of everyday things*. New York, NY: Basic books.

Paternò, F. (1999). *Model-based design and evaluation of interactive applications*. Springer-Verlag.

Quade, M., Halbrügge, M., Engelbrecht, K.-P., Albayrak, S., & Möller, S. (2014). Predicting task execution times by deriving enhanced cognitive models from user interface development models. In *Proceedings of the 2014 ACM SIGCHI symposium on engineering interactive computing systems* (pp. 139–148). New York: ACM.

Raskin, J. (1997). Looking for a humane interface: will computers ever become easy to use? *Communications of the ACM*, *40*(2), 98–101.

Rasmussen, J. (1983). Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models. *Systems, Man and Cybernetics, IEEE Transactions on*, *13*, 257–266.

Ratwani, R. M., & Trafton, J. G. (2011). A real-time eye tracking system for predicting and preventing postcompletion errors. *Human–Computer Interaction*, *26*(3), 205–245.

Reason, J. (1990). *Human error*. New York, NY: Cambridge University Press.

Ruh, N., Cooper, R. P., & Mareschal, D. (2010). Action selection in complex routinized sequential behaviors. *Journal of Experimental Psychology: Human Perception and Performance*, *36*(4), 955.

Salvucci, D. D. (2010). On reconstruction of task context after interruption. In *CHI 2010: Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 89–92).

Trafton, J. G., Altmann, E. M., & Ratwani, R. M. (2011). A memory for goals model of sequence errors. *Cognitive Systems Research*, *12*, 134–143.

Vanderdonckt, J. (2008). Model-driven engineering of user interfaces: Promises, successes, failures, and challenges. In *Proc. ROCHI 2008.*

Wilson, M. (2002). Six views of embodied cognition. *Psychonomic bulletin & review*, *9*(4), 625–636.