Predicting User Error for Ambient Systems by Integrating Model-based UI Development and Cognitive Modeling

Marc Halbrügge[1]
marc.halbruegge@tu-berlin.de

Michael Quade[2]
michael.quade@dai-labor.de

Klaus-Peter Engelbrecht[1]
klaus-peter.engelbrecht
@alumni.tu-berlin.de

Sebastian Möller[1]
sebastian.moeller
@tu-berlin.de

Sahin Albayrak[2]
sahin.albayrak@dai-labor.de

[1]Quality and Usability Lab
Technische Universität Berlin
Ernst-Reuter-Platz 7, 10587, Berlin

[2]DAI-Labor
Technische Universität Berlin
Ernst-Reuter-Platz 7, 10587, Berlin

ABSTRACT

With the move to ubiquitous computing, user interfaces (UI) are no longer bound to specific devices. While this problem can be tackled using the model-based UI development (MBUID) process, the usability of the device-specific interfaces is still an open question. We are presenting a combined system that integrates MBUID with a cognitive modeling framework in order to provide usability predictions at development time. Because of their potential impact, our focus within usability problems lies on user errors. These are captured in a cognitive model that capitalizes on meta-information provided by the MBUID system such as the abstract role of a UI element within a task sequence (e.g., input, output, command). The free parameters of the cognitive model were constrained using data from two previous studies. A validation experiment featuring a new application and UI yielded an unexpected error pattern that was nonetheless consistent with the model predictions.

Author Keywords
Automated Usability Evaluation; Human Error; Model-Based Engineering; Smart Home; Cognitive User Model

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation (e.g. HCI): User Interfaces; H.1.2 Models and Principles: User/Machine Systems

INTRODUCTION AND RELATED WORK

Have you ever forgotten to take your credit card after paying at a vending machine? Human errors like this happen regularly when we interact with software applications. Although the consequences are not always disastrous, they usually make further interaction more unpleasant and lengthy. Thus, UIs should be designed to minimize human error.

This goal is already hard to achieve in classic development contexts, but nearly unattainable in ambient environments where an application is distributed to a large number of devices with varying interaction concepts and form factors. Here, the goal is to develop *plastic* UIs [12], which adapt to different interaction contexts while preserving certain usability properties.

Unfortunately, designing for human error is not easy. One reason for this is that errors are hard to predict. Research on human error is fragmented, and predictive models exist only for small, well-defined areas (e.g., post-completion error; [9]). A second reason is that changing the UI to avoid one error can result in the introduction of new sources of errors of a different kind [51]. And thirdly, while consequences like these could be revealed during user studies, the scarcity of errors makes user studies on human error complicated and expensive.

Thus, for safety-critical systems, several methods exist which allow to consider appearance and consequences of errors in a systematic way [47]. For example, HAMSTERS [14] allows to include potential user errors in a task model representation of the UI to assess the effect of the errors and of design changes. NGOMSL allows to compute the working memory load for a task and estimate errors based on this [25]. GLEAN [53] can be used to asses consequences of errors by simulating error prone sequences of task steps. Using such methods requires extensive expertise in error analysis and the respective tools and is thus too costly for non safety-critical applications. This is even more so as many applications are developed for a multitude of different devices, with potentially different types of user errors.

Thus, automation of error prediction is desirable to simplify error analysis in terms of expertise and time spent. Work on CogTool showed that a cognitive model can be created automatically from a UI mock-up and a demonstrated sequence of task steps [24]. However, the model predicts execution times for the task sequence specified by the analyst and does not predict errors. An extension of CogTool, CogTool Explorer [49], allows to predict website navigation problems; however,

UbiComp '16, September 12-16, 2016, Heidelberg, Germany
© 2016 ACM. ISBN 978-1-4503-4461-6/16/09...$15.00
DOI: http://dx.doi.org/10.1145/2971648.2971667

this method only applies to exploratory search for information on a website by browsing lists of links.

Thus, more work on automatic error prediction is needed. Error types addressed in such work should be generally applicable to different types of UIs. Furthermore, error probabilities should depend on UI characteristics in such a way that they can be avoided with a better UI design.

In this paper, we present an integrated system which allows to predict erroneous omissions of task steps depending on UI element characteristics. The system combines a cognitive model that is based on the Memory for Goals theory [1] with a model-based user interface development framework. Model-based user interfaces address the problem of high diversity of devices an application may run on by providing the interaction logic and UI characteristics as models. These models encode high-level semantic information. We will show that this information can be employed to distinguish UI elements into classes with differing probabilities of user errors. Thus, accurate error predictions are possible without further intervention by the analyst. Furthermore, in our system the user model interacts with the application model directly, i.e., no additional tool-specific mock-up of the application is required. Thus, one of the most important bottlenecks of model-based usability evaluation, the cumbersome creation of a UI mock-up, is avoided.

The rest of this paper is organized as follows. We first give an overview on MBUID and the theory of human error that forms the basis of our cognitive user model. Next, we present data that allows to elaborate on the assumptions underlying our error model, and we explain how the model was integrated with the MBUID system. A validation experiment provides evidence that the whole approach generalizes to other UIs. We show that the model predicts the probability of errors very well despite unexpected results in the user study. We conclude with a discussion of the strengths and limitations of our system and a summary of our contributions.

MODELS FOR UI DEVELOPMENT AND EVALUATION

Providing UIs that are not just adaptive but also follow usability criteria remains an important research question. The ability of user interfaces to adapt to different variations within the context of use (i.e., different situations) and thereby being able to preserve certain usability aspects (e.g., observability and predictability; [16]) within a predefined range of properties is defined as *plasticity* [12]. Striving for plasticity produces additional requirements on the development processes of user interfaces. A conceptual model of a development process for plastic user interfaces is the CAMELEON reference framework [10], which is based on the development process of Model Driven Architecture [29]. Examples of implementations of the framework are UsiXML [28] and TERESA [30].

Model-based UI development specifies information about the UI and interaction logic within several models that are defined by the designer [52]. The model types that are part of the CAMELEON framework belong to different levels of abstraction. The process starts with a highly abstract task model, e.g., using ConcurTaskTree (CTT) notation [33]. In contrast to other task analysis techniques, the CTT models contain both user tasks (e.g., data input) and system tasks (e.g., database query). On the next level, an Abstract User Interface (AUI) model is created that specifies platform-independent and modality-independent interactors (e.g., 'choice', 'command', 'output'). At this level, it is still open whether a 'command' interactor will be implemented as a button in a graphical UI or as a voice command. In the following Concrete User Interface (CUI) model, the platform and modality to be used is specified, e.g., a mock-up of a graphical UI. On the last level, the Final User Interface (FUI) is the UI that users actually interact with, e.g., a web page with text input fields for data input and buttons for triggering system actions.

This conceptual approach has also been applied in runtime architectures for model-based applications (e.g., [11, 44, 6]). These runtime architectures derive the FUI from current information in the models and thus are able to adapt the UI to changes in the models, thereby reducing complexity during development.

Using Development Models for Evaluation

While the main focus of MBUID is on how adaptable UIs can be *developed* efficiently, the MBUID approach can also help to *evaluate* the usability of plastic user interfaces [46]. By having a defined syntax and semantics, the models provide computer-processable information about interaction flow, layout, and design decisions. Consequently, valuable information can be accessed from these development models if they are still available during usability evaluation. This allows for a far deeper analysis than evaluating the surface of the UI only.

Combining approaches for model-based usability evaluation with MBUID have proven to predict valuable results, e.g., simulations of task models [34] or predictions using cognitive architectures in conjunction with UsiXML [15]. Yet, none of these approaches has addressed an automated extraction of information required for simulating human error. In previous work, we described an integrated system for automated usability evaluation of model-based applications that accesses required information from relevant UI development models [39, 38]. This system is based on the MeMo workbench for usability testing [13]. MeMo's user simulation progresses towards a given goal state by utilizing a combination of breadth-first and depth-first search on the state graph of the application under evaluation. By integrating MeMo with the Multi Access Service Platform (MASP; [6]), a CAMELEON-conformant runtime framework, this state graph can be extracted directly from the application (see Figure 1), thereby saving the analyst from creating a mock-up UI with MeMo [37].

ACTION CONTROL AND PROCEDURAL ERROR

Human interaction with software systems is mainly controlled based on stored rules and procedures that have been formed during earlier encounters and/or training [40]. Errors during the application of such rules are called *procedural errors*. They differ from knowledge-based *mistakes* [31] in that the intentions behind the actions are correct, only their execution went wrong. The latter is also true for sensory-motor *slips*

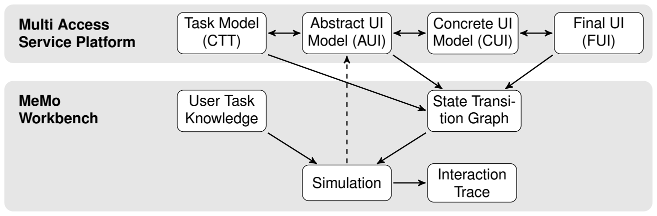


Figure 1. Structure of the integrated system. Information flow is denoted by solid arrows. The MeMo simulation engine applies simulated user interactions on the AUI level (dashed arrow) and the MASP propagates their effects to the other levels of the runtime framework.

[31], but contrary to slips, procedural errors are memory-based (e.g., forgetting a subtask).

More formally, procedural error can be defined as the violation of the (optimal) path to the current goal by a non-optimal action. This can either be the addition of an unnecessary or even hindering action, which is called an *intrusion*. Or a necessary step can be left out, constituting an *omission*. In this paper we focus on omissions, only. This is done for several reasons. First, especially in applied contexts, omissions are far more frequent than intrusions (e.g., [17, 26, 5]). Second, omissions are better researched than intrusions, e.g. omissions after completion of the main goal [9] or omissions of initialization steps [22]. And finally, omissions are rather a function of the UI design, while intrusions are more dependent on interactions between several user tasks [17]. The focus on omissions therefore fits best to the main goal of this work, i.e., to provide usability predictions to UI designers at development time.

Memory for Goals Theory

An explanation of procedural *error* must incorporate the generation of *correct* behavior as well. A very promising theoretical model of sequential action is the Memory for Goals (MFG) theory [1]. The MFG proposes that subgoals, i.e., atomic steps towards a goal, are represented in human memory, thereby underlying memory effects like time-dependent and noisy *activation*, *interference*, and associative *priming*. Within the MFG theory, errors arise when the activation of a goal is not high enough to surpass interfering goals or even falls below a general retrieval threshold. Cognitive models based on the MFG assumptions have been shown to explain procedural errors in the HCI domain, namely omissions, very well [50, 51, 21, 19, 27].

The Role of the Environment

The MFG theory is clearly focused on how humans manage task sequences in memory, i.e., within their head. This has been criticized for neglecting the role of the environment [43]. As embodied beings, humans strive to reduce cognitive complexity by exploiting the content and structure of the external world. In our own research, we have shown that by extending the MFG with an activation process relying on external cues, better predictions can be achieved and new error domains can be covered [19]. In the following, this will be referred to as the *knowledge-in-the-world assumption* [31]. It proposes that when a user cannot retrieve the next goal, they revert to an externalization strategy. The visual scene (i.e., the UI) is searched[1] for interactive elements. Whenever an element is found, the user tries to retrieve a goal that relates to this UI element. Because visually attending the element increases the activation of related goals through priming, a goal that had previously been forgotten may now surpass the retrieval threshold. As a consequence, the planned sequence of actions can be resumed.

Task- and Device-Orientation

The best known examples of procedural error during system use are post-completion errors [9], e.g., forgetting the originals in the copy machine, and initialization errors [17], e.g., forgetting to reset Caps Lock before typing a password.

Common to both of them is that these errors happen during procedural steps that do not directly contribute to the users' actual goals (i.e., making copies; logging into a system). This common property of goal-irrelevance of a sub-task has been coined *device-orientation* [2, 17], its opposite is analogously called *task-orientation*. The concept of device-orientation builds upon the MFG by assuming that device-oriented tasks are "more weakly represented in memory". While Ament et al. [2] discuss different encoding or lack of rehearsal as possible reasons for lower activation of device-oriented tasks, we are assuming lack of priming in this case. We will call this the *task priming assumption* in the following.

The elimination of device-oriented tasks is a reasonable design strategy for error reduction. It can also serve as heuristic for usability experts, but no automatic tool exists that evaluates a UI design based on device-orientation. Why is this the case? Evaluation tools like CogTool Explorer [49] can only work on what is visible on the surface of an interface, e.g., textual labels on the UI's interactive elements. While it is relatively easy for a human to sort UI elements into the task- vs. device-oriented categories, automatic tools lack the necessary information about the relationship between users' goals, the chains of subtasks towards goals, and the corresponding UI elements. How can we provide information that goes beyond the surface of the UI to automatic tools?

AUTOMATING MODEL-BASED USABILITY EVALUATION

Our integrated architecture is implemented using application models from the MASP runtime framework [6]. Interaction with application models from the MASP is provided through integration with the MeMo workbench for user simulation. In [38] we described the generic architecture and processes for extracting relevant UI information and performing interaction between all models. An overview of the structure is shown in Figure 1. Based on the additional information from the AUI and CTT models, the system can distinguish between more types of user interactions than what could be derived from the FUI alone, e.g., a button on the FUI level can be of AUI type 'choice' as part of an input task, but it can also be of AUI type 'command', subsequently enabling a system task. This distinction may seem trivial, but it provides substantially better

[1]Note: We currently do not make specific assumptions about the nature of the search process apart from that it is at least partially stochastic [8]. This is not a strong theoretical assumption, but rather a means of keeping the theory simple (Occam's razor).

predictions (explained variance increased from $R^2 = .735$ to $R^2 = .965$ in the validation experiment; [38]). Furthermore, the system can automatically derive semantic relationships between FUI elements (e.g., buttons) from their grouping on the AUI level. This knowledge is exploited for additional improvements of the time predictions that would otherwise need human intervention by a usability expert [38].

The actual simulation is driven by the tasks of the user. A single task is defined by its start state (e.g., the home screen of the UI), its goal state, and an arbitrary number of *user task knowledge* items. The task knowledge is the set of information items that the user has to transmit to the software application in order to attain their goal. The individual items in the user task knowledge are applied on the path from start state to goal state by comparing them to the captions of the currently visible elements of the UI. If an element matches, a corresponding interaction is performed by the simulation (dashed arrow in Figure 1). Such interactions usually correspond to task-oriented user goals.

Additional steps may be necessary to proceed towards the goal state, e.g., navigation to subsequent UI screens. These are found by the MeMo based on path search on a state transition graph that it derives from the task model, AUI model, and FUI of the MASP application (see Figure 1). Interactions without corresponding user task knowledge items correspond to device-oriented user goals.

An example of how the system works is visualized in Figure 2. It is based on the task "Search for German main dishes and select lamb chops" from the user studies presented in the following section (see UI in Figure 3). The verbal task description is first divided into three items: 'German', 'main dish', and 'lamb chops'. The start state is the home screen of the UI, the goal state is a screen that gives recipe information like preparation time and calorie content.

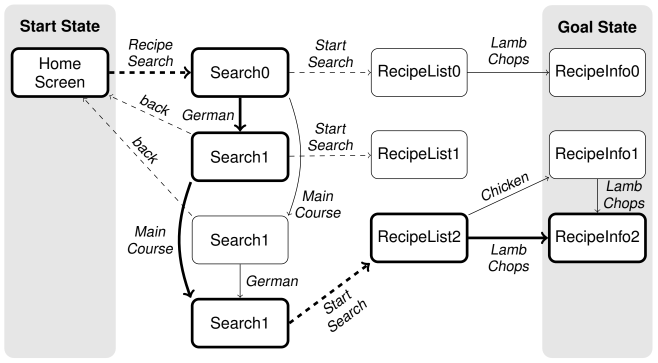


Figure 2. MeMo state transition graph (detail) of a recipe search. Nodes denote states, arrows denote transitions. Transitions are labeled with the caption of the corresponding UI element. Thick lines represent an interaction path simulated by MeMo for the user task knowledge {'German', 'Main Course', 'Lamb Chops'}. Solid arrows are task-oriented interactions that are selected based on the user task knowledge, dashed arrows are device-oriented interactions that MeMo derives from the MBUID models. State 'RecipeList2' corresponds to the screenshot in Figure 3.

For a more comprehensive evaluation of an application, several user tasks are integrated within a dedicated task description file. This file needs to be prepared manually and allows to structure the tasks hierarchically. Omitting device-oriented steps (e.g., page navigation) from the task description file is not only done for convenience, it is also an important prerequisite for evaluating the plasticity of an adaptable multi-target application. As the target-specific FUIs usually differ, e.g., in how the steps within a task sequence are spread over several pages of the interface, so would the task descriptions if they would contain all user actions. Therefore, task descriptions that can be applied across different FUIs of an application need to abstract from the characteristics of a specific FUI.

EXPERIMENTS

Before we show how the integrated architecture can be extended for error prediction, we need to make sure that our theoretical assumptions are appropriate approximations of omission errors. For this reason, we have conducted two user studies that shed light on the relationship between device-orientation and omissions.

We selected a MASP-based kitchen assistance system for the experiments. It aims at helping with the preparation of meals by suggesting recipes, calculating ingredient amounts and maintaining shopping lists. A screenshot of the recipe search screen of the kitchen assistant is displayed in Figure 3. Both studies have been analyzed with a different focus, before [19, 20]. We are therefore just giving an overview over the methods and combined results that were used for the training of the integrated system.

Previous analyses have shown that the concept of device-orientation is not sufficient to explain omission errors [19, 20]. A common strategy to facilitate device-oriented tasks is to make them obligatory, e.g., most current teller machines only hand out cash (the overall goal) *after* the card has been taken (a device-oriented step that is omission-prone otherwise). We call this property *task necessity* and analyze its impact on omissions below.

Methods

Participants

The first experiment was conducted in July and August 2014, the second one in January 2015. A total of 44 members of the Technische Universität Berlin paid participant pool took part. There were 14 men and 30 women, aged between 18 and 59 (M=30.7, SD=10.6). As the instructions were given in German, only fluent German speakers were allowed. Informed consent was obtained from all participants.

Materials

A personal computer with 27" (68.6 cm) touch screen and a 10" (25.7 cm) tablet were used to display the interface of the MASP-based kitchen assistant during experiment 1. In the follow-up experiment, we chose a 23" (58.4 cm) monitor with optical sensor 'touch' technology instead of the personal computer of experiment 1. While the large screens operated in landscape mode, portrait mode was used for the tablet. We created two variations of the UI of the kitchen assistant that were targeted at the large screen and the tablet, respectively. All user actions were recorded by the computer system. The subjects' performance was additionally recorded on videotape for subsequent error identification. While experiment 1 was

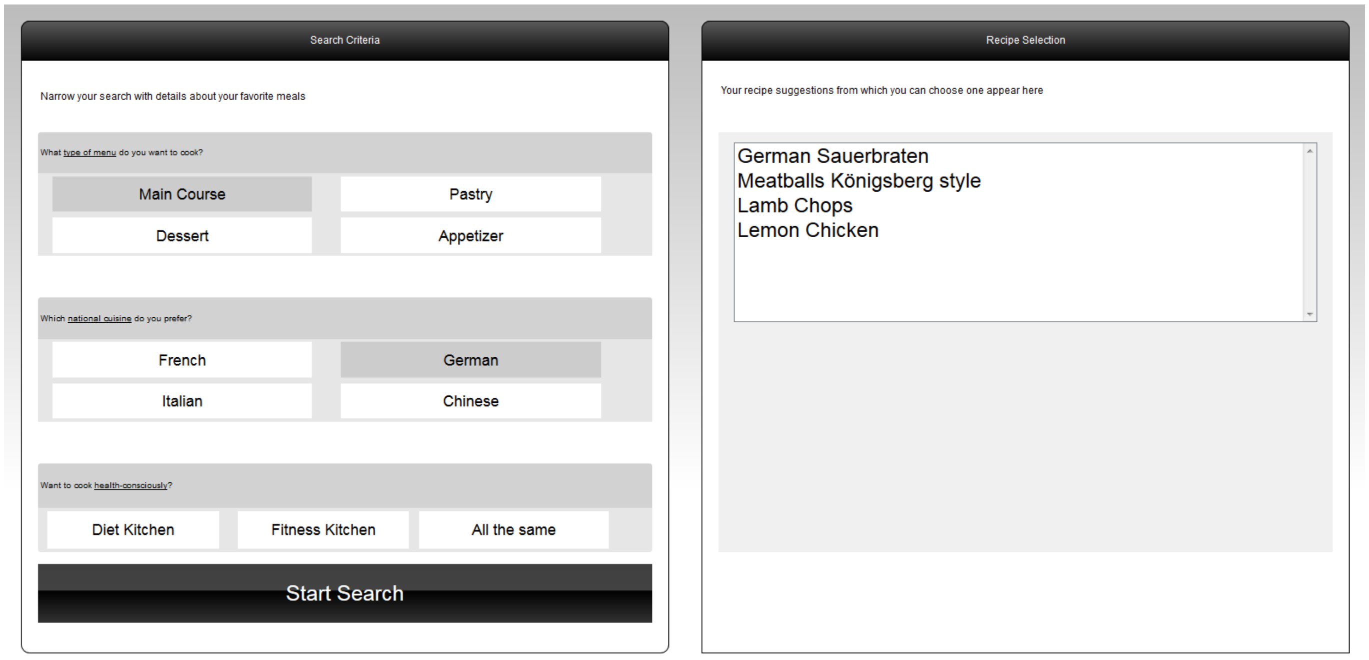

Search Criteria
Narrow your search with details about your favorite meals
What type of menu do you want to cook?
Main Course
Pastry
Dessert
Appetizer
Which national cuisine do you prefer?
French
German
Italian
Chinese
Want to cook health-consciously?
Diet Kitchen
Fitness Kitchen
All the same
Start Search
Recipe Selection
Your recipe suggestions from which you can choose one appear here
German Sauerbraten
Meatballs Königsberg style
Lamb Chops
Lemon Chicken

Figure 3. Screenshot of the English version kitchen assistant used for experiments 1 and 2.

conducted in a real kitchen in our lab space, the additional application of eye-tracking during experiment 2 required fixed lighting conditions that could only be achieved in a neutral environment.

Design

We used a four-factor within-subject design, the factors being the physical device used, the UI variant, task necessity (obligatory vs. non-obligatory), and whether it was device-oriented as opposed to task-oriented. We collected all user errors and task completion times, but will only present an analysis of omission errors, here. The participants completed a total of 46 tasks grouped into 4 blocks. Physical device, UI variant, and block sequence were varied randomly, but counterbalanced across each experiment. Experiment 2 contained some additional tasks that were introduced to shed more light on non-obligatory device-oriented tasks. We are only reporting the sub-set of tasks that were shared between both experiments, here.

Procedure

After having played a simple game on each of the two devices to get accustomed with the respective touch technology, the participants received training on the kitchen assistant. The training covered all parts of the application that were used during the actual experiment. Each block of tasks began with relatively simple tasks like “Search for German main dishes and select lamb chops”. Afterwards, the ingredients to a recipe were collected (e.g., “How much meat is needed for four servings”) and some of them were added to a shopping list that is part of the kitchen assistant (“How many items are on your shopping list”). The complete procedure lasted less than 60 minutes for both experiments.[2]

Results

We collected a total of 11124 clicks, 92 (0.83%) thereof were omissions. A mixed logit model with subject and block as random factor [4] yielded no effects of physical device, UI version, or experiment (all $p > .4$). There is a significant main effect of device-orientation, but it points into the opposite direction with device-oriented subtasks showing lower error rates ($z = 2.91, p = .004$). Obligatory subtasks were less prone to omissions ($z = -2.11, p = .035$), and there is a significant interaction between task necessity and device-orientation ($z = -3.57, p < .001$). Omission rates along with confidence intervals are presented in Figure 4.

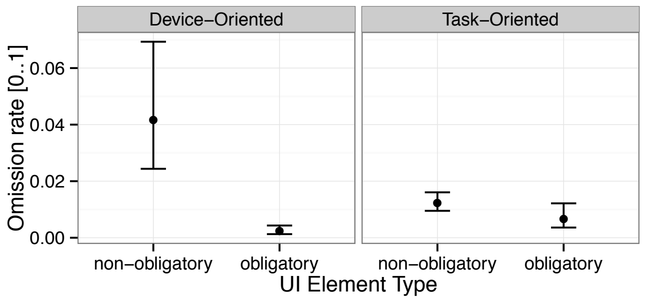


Figure 4. Omission rate for different types of tasks for the aggregated data of the first and second experiment. Error bars denote 95% confidence intervals of the empirical omission rates based on the Agresti-Coull method.

[2]The full instructions are available for download at http://www.tu-berlin.de/?id=135088

Discussion
Several results are noteworthy on the background of the theoretical assumptions laid out before. First of all, the direction of the device-orientation effect contradicts our expectations based on the *task priming* assumption. Why is this the case? The general inconvenience of device-oriented tasks is often counteracted by making them obligatory. In case of the kitchen assistant, 86% of the device-oriented steps were also obligatory, while 74% of the task-oriented steps were non-obligatory. Together with the interaction between device-orientation and task necessity being significant, this does not mean that the main effect of device-orientation contradicts the task priming assumption, but that both main effects should be interpreted with care. Device-orientation cannot be used in isolation; task necessity is at least of equal importance. Only if the interaction of both factors is taken into account, we can explain why some steps are more often forgotten than others. This result feeds back to our theory by demanding that a predictive model of user error must incorporate knowledge about the control flow of the application as well (operationalized here as task necessity). The importance of the application logic is often underestimated in error research because such research is mainly conducted in specific laboratory settings that favor high error rates over generalizability to the world outside the laboratory (e.g., [50, 2, 27]).

With regard to the model-based design of plastic user interfaces, it is worth noting that the property whether a user task is obligatory or not is only represented on the least abstract FUI level. As a result, valid error predictions for a MBUID system are not possible if the FUI level is not being considered. We will return to this point in the General Discussion below.

Finally, the data questions our assumptions in a second way. If the task priming assumption is correct, the activation of tasks like moving to the next screen should be relatively low, i.e., they should be prone to retrieval errors. But if the participants cannot remember them, how do they manage to complete them at all? Obligatory device-oriented tasks showed the lowest omission rate in our experiments. The knowledge-in-the-world assumption provides an answer to this question: While searching the UI for 'inviting' elements, the currently attended element receives additional *visual* priming. This increases the activation of the corresponding subgoal so that the user is able to retrieve it.

INTEGRATED SYSTEM FOR ERROR PREDICTION
Our theoretical assumptions have been modeled within the cognitive architecture ACT-R [3] and have been compared to the data presented above to ensure their validity [19, 20]. While this approach is helpful to advance psychological theory, it does not scale well to applied scenarios. We therefore chose to replicate the core components of the ACT-R model within the MeMo system that had already been integrated with the MASP framework that has been used for the experiments [38].

In order to make this integrated evaluation system capable of errors, its until then optimally behaving (path search based) user simulation had to be modified in a reasonable manner. This was done based on the MFG theory as introduced above. While path search on the state graph of the application is still used to find an optimal sequence of interactions to attain a goal, the individual interactions (i.e., arrows in Figures 2 and 5) in the resulting sequence are now subject to memory activation.

Goal Activation Computation
Each element of the task description that the model is following (i.e., the goals) was extended by a numerical activation value. For reasons of simplicity, these activation values are not computed using ACT-R's sophisticated activation formulae, but are drawn from a standard Gaussian random variable instead (see Figure 6). Omissions occur when the activation of a goal (i.e., element of the task description) happens to fall below a fixed retrieval threshold rt. Visual priming while searching the screen when using the knowledge-in-the-world strategy is achieved by adding a fixed visual priming constant vp.

Task Priming is modeled as a positive numerical value that is added to the goal's activation, thereby reducing the probability of retrieval failure. It represents activation spreading from the overall goal (e.g., looking up the 'lamb chops' recipe) to a subgoal leading to it (e.g., using a 'main dish' search attribute). We are using LTMC [45], a dedicated long term memory module with sophisticated activation spreading to compute the amount of task priming. LMTC's main purpose is to represent knowledge about the world as semantic networks [18]. The integrated system as presented here uses it to represent the user knowledge about the current system, only. This was achieved by adding all recipes contained within the kitchen assistant to LTMC together with their attribute mapping, i.e., the semantic network was built up from facts like 'Panna Cotta is a dessert' or 'Sauerbraten is a main dish'.

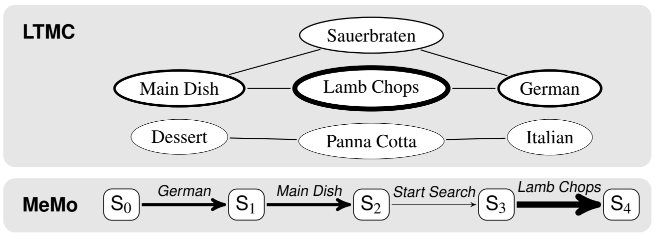


Figure 5. Knowledge representation and application of spreading activation within LTMC and application to MeMo simulation. Activation is visualized as node border thickness in LTMC and as arrow thickness in MeMo. By highly activation the overall goal of the sequence ('Lamb Chops'), semantically connected nodes ('German', 'Main Dish') receive priming. The resulting activation values are applied during MeMo's interaction simulation. Note that the device-oriented subgoal 'Start Search' does not receive any priming at all.

Within the network, a specific dish (e.g., 'Lamb Chops') spreads activation to its attributes ('German' and 'main dish') but not to the other nationalities or types of dish. Search attributes accordingly spread activation to recipes, but as more recipes belong to a single search attribute than attributes to recipes, the amount of spreading is smaller in this direction. For each action of the user model, the LTMC module is run to compute the amount of priming that the goal that corresponds to the step receives. The result of the computation within LTMC is applied after rescaling it by the maximum task priming constant tp. The process is visualized in Figure 5.

Fitting Procedure

We are using the combined results of both experiment 1 and 2 as empirical basis of the user model. The model has three free parameters, the value of each can be directly estimated from the data.

- The retrieval threshold *rt* below which a task is forgotten is estimated from the data using the omission probability of device-oriented non-obligatory tasks ($p = .037, z = -1.78$, see Figure 6). This task category does neither receive task priming, nor is it enforced by the application logic. As the users must completely rely on their memory in this case, the empirical omission rate should be a good estimator of the retrieval threshold.
- The amount of task priming *tp* is estimated as the difference in omission rates between task-oriented and device-oriented non-obligatory steps ($\Delta z = 0.48$).
- Visual priming *vp* is estimated using the difference between obligatory and non-obligatory device-oriented tasks ($\Delta z = 1.02$).

We assessed the fit of the resulting model by performing 100 model runs with the task set used for experiment 1. The fit to the training data is very good, R^2=.99, RMSE=.0038. The Maximum Likely Scaled Difference (MLSD; [48]), that takes the variability of the data into account, is 1.94. Being so close to its theoretical minimum of 1 means that the model should not be refined without the danger of overfitting.

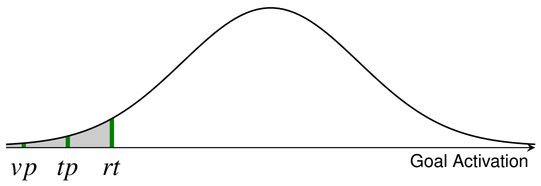


Figure 6. Mapping of empirical omission rates to the parameters of the cognitive user model. Activation is assumed to be a normally distributed random variable. The area under the curve left of one of the lines indicates the omission probability of a corresponding goal.

VALIDATION EXPERIMENT

In order to test the generalizability of the user model, we designed a new experiment using a different application from a similar domain and new types of user tasks. The new application is a MASP-based health assistant that has been developed as part of a health information system for migrants [36].

The health assistant contains a recipe search similar to the kitchen assistant used in the previous experiments. In contrast to the kitchen assistant, the health assistant's UI is built around the health conditions of its users. It also features a finer grained shopping list generator that can handle several personalized lists at a time.

Methods

Participants

The experiment was conducted in July and August 2015. 30 participants, 15 men and 15 women, aged between 19 and 56 (M=33.7, SD=8.7), were recruited from the paid participant pool of Technische Universität Berlin.

Materials

The health assistant was displayed on a 23" (58.4 cm) monitor with optical sensor 'touch' technology similar to the one used during experiment 2 and on the 10" (25.7 cm) tablet used before. All devices operated in landscape mode. User actions were again recorded by the computer system and additionally videotaped for error classification.

Design

Dropping the comparison of different UI versions, we could reduce our experiment to a three-factor within-subjects design. The remaining independent variables were physical device (screen vs. tablet), device-orientation and task necessity. We had a new set of tasks generated by a different researcher to reduce implicit bias towards simple paraphrase of the task instructions used during experiment 1 and 2. The resulting 35 user tasks were grouped into four trial blocks consisting of 8 to 10 trials.

Procedure

After a quick warm-up game, the participants received about 10 minutes of training with the system. Four personas, three of them with the health conditions diabetes, pregnancy, and lactose intolerance, were introduced to them. The experimental blocks started with simpler tasks like counting the number of lactose-free recipes available in the system, or comparing health-related nutritional information. The participants were then walked through a background story (e.g., planning dinner for one of the personas) that comprised selecting which recipes to prepare and creating individualized shopping lists. The whole procedure took a little less than one hour.

Results

We recorded 7699 clicks in total, 297 (3.9%) thereof were omissions. There was no effect of device (mixed logit model with subject and trial block as random factors [4], $z = .28, p = .78$), but both task necessity ($z = -3.79, p < .001$) and device-orientation ($z = -2.46, p = .014$) had significant influence on the omission rate. The interaction between both was significant as well ($z = 2.06, p = .039$).

Discussion

The overall error rate is higher than in the previous experiments, but still within the usual range of 5% for procedural error [41]. The biggest difference is the high omission rate for non-obligatory task-oriented goals that even exceeds the omission rate of their device-oriented counterpart. This result is rather unexpected because the previous experiments produced the opposite pattern in line with the task-priming assumption (compare Figures 7 and 4).

The user tasks connected to this category are a) selecting ingredients and b) toggling search attributes. We speculate that the use of rather uncommon ingredients (e.g., soy-drink, quince purée) and search attributes (e.g., lactose intolerance) caused this high omission rate.

Model Fit

In order to assess the generality of the model, we used the model parameters as they had been fit to the previous data (see above and Figure 6). The model completed the health assistant trials 100 times, the fit to the new data is good with R^2=.70. While the rather high RMSE of .021 indicates the model missing the generally higher omission rate in the validation experiment, the MLSD [48] value of 2.6 indicates a reasonable fit given the uncertainty in the empirical data (see Figure 7). Despite not anticipating the overall increased omission rate, we see that the higher omission rate of task-oriented non-obligatory tasks is well captured by the model.

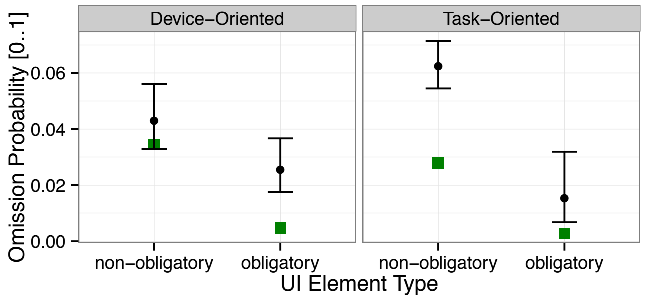


Figure 7. Fit of the cognitive user model to the validation experiment. Error bars denote 95% confidence intervals of the empirical omission rates; 🟩 denote model predictions.

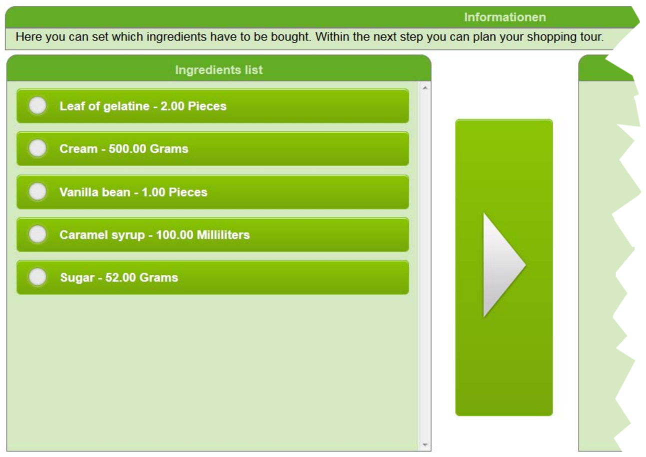
More important than the quantitative fit is the qualitative usefulness of the model: Can it facilitate the UI design process, i.e., does the model identify UI elements that are especially prone to errors? We approached this question by comparing the omission rates for different types of UI elements on a rank basis. As presented in Table 1, most ranks are matched between data and model. The biggest difference occurs for *search attribute* buttons, which were empirically the 2nd most forgotten type while the model predicted them being ranked 5th.

UI element type	device-oriented	rank (data)	rank (model)
Select ingredient	–	1st	1st
Toggle search attribute	–	2nd	5th
Move ingredient to shopping list	yes	3rd	2nd
Select persona	–	4th	4th
Assign ingredient to persona	yes	5th	3rd
Continue on next screen	yes	6th	6th
Select recipe	–	7th	7th

Table 1. Ranks of the empirical and predicted omission rates.

Especially noteworthy is that both model and empirical data show the highest omission rates for the *select ingredients* type of element. An exemplary screenshot of the health assistant's ingredients list is shown in Figure 8. This element falls into the task-oriented category, i.e., it should be rather resilient to omissions following the task priming assumption. How does the model produce this prediction?

Three factors contribute to the high omission rate of ingredients: First, the ingredients appeared unusual to some of the participants and were only loosely connected to the recipes (e.g., tofu and teriyaki sauce for a pepper skewers recipe). The amount of task priming was therefore practically zero. Second, the trials concerning ingredients were rather long and often contained several ingredients. With every ingredient added to the list, the probability that (at least) one ingredient in a trial is forgotten becomes much higher. And finally, once a goal retrieval fails, the visual search for suitable elements as proposed in the *knowledge-in-the-world assumption* is a (partially) unsystematic process. In case of long ingredients lists, a distracting list entry can take over control and the forgotten goal never gets a chance to come into action.[3]



Here you can set which ingredients have to be bought. Within the next step you can plan your shopping tour.

Ingredients list
Leaf of gelatine - 2.00 Pieces
Cream - 500.00 Grams
Vanilla bean - 1.00 Pieces
Caramel syrup - 100.00 Milliliters
Sugar - 52.00 Grams

Figure 8. Ingredients list of the health assistant.

GENERAL DISCUSSION

In the following, we are discussing our integrated system from three perspectives: the validity of the user model, the benefits and limitations compared to comparable existing approaches, and the applicability of the integrated system during the model-based development of plastic UIs.

Validity of the Cognitive User Model

The generalizability of the model was fostered on several ways. First, our studies have been designed using real world applications and usage scenarios and have drawn participants mainly from non-student populations. The generality of the model was examined in a validation experiment using a different application and new set of tasks. Although this led to an unexpected error pattern, the model fits the new data very well. Second, by basing the user model on psychological theory, we can build our model on the results of many other researchers instead just our own studies.

The model extends on the activation-based Memory for Goals theory [1] by highlighting the importance of external cues during sequential action. Internal cues are divided into task-oriented, i.e., steps that directly contribute to the users' goals, and device-oriented ones [2]. Together with the assumption that only task-oriented steps receive additional priming from the user's overall goal, this allows not only to explain our data, but also provides an acceptable explanation of how device-orientation actually affects sequential behavior.

[3] Note that the first and the last factor are of general nature, only the co-occurrence of all three has led to the specific error pattern of the validation experiment.

The original MFG model is based on the ACT-R theory [3] which provides a psychologically plausible framework for the computation of activation values. In this paper, we traded the cognitive plausibility provided by ACT-R for the practical applicability of the model to design questions in HCI. By reducing a previously validated ACT-R model [19] to a simple probabilistic model with few parameters, it was possible to integrate the model into the MASP system, a runtime architecture for model-based applications. Thereby, no mock-up of the application is necessary to perform the evaluation, and the automatic identification of device-oriented (i.e., potentially error-prone) task steps is possible through inspection of UI meta-information (i.e., AUI and CTT model) provided by the MASP

The error model has several limitations. First, the model only covers expert behavior. The initial formation of the task sequence by novice human users is beyond its capabilities. The model also does not have any general knowledge and therefore cannot account for errors caused by the UI design violating general expectations of its users towards computer systems (e.g., about the functionality of a 'home' or a 'back' option). Second, the connection to a specific MBUID system limits the applicability of the model to applications that are developed within that system. Third, the domain knowledge of the user model within LTMC is currently restricted to the information that is used and provided by the application. Thereby, it is not possible to spot inconsistencies with user expectations.[4]

Comparison to other Approaches

While the system presented here is the first to combine MBUID with psychologically plausible user models, the prevention of user error based on UI meta-information has been proposed before.

Paternò and Santoro have proposed an inspection-based evaluation of safety-critical systems based on the task model of such a system which also forms the central part of the MBUID process [35]. This early approach still lacks both the automation and the predictive value.

In a follow-up paper, Mori and colleagues presented the TERESA tool that uses the task model to deduce problematic states like dead ends or unreachable states within a UI [30]. As this approach is not grounded in psychological theory, it fails to incorporate the human aspect of error and can not account for typical user errors as discussed in the literature (e.g., [9, 22, 17]) or the specific error patterns presented in this paper. TERESA rather targets errors during the formation of the task model as part of the requirements engineering process than errors actual users would make after the release of the product.

Palanque and Basnyat [32] build upon the task model of an application to predict the tolerance of a system towards user errors. But similarly to Paternò and Santoro [35], their approach consists only of a systematic inspection of the task model while adding a multitude of possible types of errors based on the work of Reason [41], Rasmussen [40] and Hollnagel [23]. The inclusion of error theories represents an improvement over [35], but the approach still lacks automation and prediction. Also, our data show that the information on the level of the task model is not sufficient for the prediction of omission errors. Many device-oriented tasks, e.g., navigation to a subsequent page, are not even represented on the AUI model level but introduced later when the actual FUI is developed.

In recent years, formal verification has been proposed to ensure error tolerant systems [7, 42]. Due to the formal nature of the approach, its application requires highly specific knowledge during the modeling of the system under evaluation. The software used for the verification task is computationally heavy and does not provide the automation that is possible by the integration with an MBUID system. The approach may nonetheless be worth the effort in safety-critical scenarios, e.g., as part of the general approval process in the medical domain. In contrast, our approach is meant to provide early predictions of usability issues during ongoing development processes that are applied regularly, e.g., as part of a continuous integration system. It is therefore designed to be easy to use and meant to provide first hints about which part of an UI should be redesigned.

Common to all approaches discussed above is that they can not account for the problem of *plasticity* [12] of adapted UIs as they are needed for ubiquitous systems. The sheer number of possible devices and form factors renders inspection-based or verification-based approaches impossible if all UI variations are meant to be evaluated. The automation provided by integrated systems like the one presented in this paper solves this problem.

Applicability in the Development Process

The usefulness of the integrated system can be assessed by comparing the time and money spent on the validation experiment to the time that the simulation needed. Both approaches share the initial task of planning the evaluation (3 days). Running the validation experiment with 30 participants took six days, manually annotating the videos took another five days, and the statistical analysis another two days. The work was evenly shared between a researcher and a student worker, which leads to a conservatively estimated daily rate of 150€. The participants of the study were paid a total of 300€. Neglecting additional costs for equipment and room rent, this sums up to a total of 2700€ for the experiment, compared to 450€ for the simulation. Furthermore, simulating 100 users took two days on a standard consumer laptop, which is much faster than the 11 days of empirical data collection and video annotation.

When it comes to evaluating the plasticity of adaptable UIs, the *scalability* of the empirical and the simulation approach becomes of highest importance. In the empirical case, money and time costs for conducting experiments and annotating videos multiply with the number of UI adaptations that need to be covered. The simulation on the other hand only needs more computational processing time for each new version of the FUI, making it possible to evaluate the usability of arbitrary

[4]Example: The kitchen assistant contains a recipe for Ratatouille (French vegetable stew) and files it under main dish. During the usability studies, two participants voiced objections because they regarded Ratatouille as being a side dish, only.

numbers of different UIs at stable costs as long as the AUI and CTT models remain unchanged.

Finally, the UI of the health assistant needed to receive some polishing before the user tests could start which led to extra costs and time delay. The automated system on the other hand does not get distracted by broken images or skewed layouts that quickly grab the participants' attention during user studies. The last point is especially important during early design stages when no presentable UI is available.

CONCLUSIONS AND FUTURE WORK

We have presented a UI development framework for ambient applications integrated with a user modeling system. This combination can provide usability predictions during early development stages, with the intention to improve the quality of the application while reducing costs that would arise from late redesigns of the UI (due to usability issues). Major benefits of the integrated system are the improved error predictions due to the exploitation of meta-information provided by the MBUID system and the high grade of automation. The usability predictions of the integrated system are drawn from a model of human error grounded in cognitive science theory. A validation experiment using a new application and new set of user tasks confirms both the assumptions of the model and the general suitability of the approach.

Future work will explore the applicability of the integrated system in other domains and/or interaction paradigms (e.g., speech). The current limitations of the error model will be approached by incorporating information on how users construct the application logic and content from ontologies that are external to the application. This way, usability problems like ambiguous captions of UI elements could be spotted by the integrated evaluation system as well.

ACKNOWLEDGMENTS

This work was supported by DFG grant MO 1038/18-1 ("Automatische Usability-Evaluierung modellbasierter Interaktionssysteme für Ambient Assisted Living").

REFERENCES

1. Erik M Altmann and J Gregory Trafton. 2002. Memory for goals: An activation-based model. *Cognitive science* 26, 1 (2002), 39–83. DOI: http://dx.doi.org/10.1207/s15516709cog2601_2
2. Maartje GA Ament, Anna L Cox, Ann Blandford, and Duncan P Brumby. 2013. Making a task difficult: Evidence that device-oriented steps are effortful and error-prone. *Journal of experimental psychology: applied* 19, 3 (2013), 195. DOI: http://dx.doi.org/10.1037/a0034397
3. John R Anderson, Daniel Bothell, Michael D Byrne, Scott Douglass, Christian Lebiere, and Yulin Qin. 2004. An integrated theory of the mind. *Psychological review* 111, 4 (2004), 1036–1060. DOI: http://dx.doi.org/10.1037/0033-295X.111.4.1036
4. Douglas Bates, Martin Maechler, Ben Bolker, and Steven Walker. 2013. *lme4: Linear mixed-effects models using Eigen and S4*. R package version 1.0-5.
5. Arthur N Beare and RE Dorris. 1983. A simulator-based study of human errors in nuclear power plant control room tasks. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, Vol. 27. Sage Publications, 170–174. DOI: http://dx.doi.org/10.1177/154193128302700213
6. Marco Blumendorf, Grzegorz Lehmann, and Sahin Albayrak. 2010. Bridging Models and Systems at Runtime to Build Adaptive User Interfaces. In *Proceedings of the 2Nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '10)*. ACM, New York, NY, USA, 9–18. DOI: http://dx.doi.org/10.1145/1822018.1822022
7. Matthew L Bolton, Ellen J Bass, and Radu I Siminiceanu. 2012. Generating phenotypical erroneous human behavior to evaluate human–automation interaction using model checking. *International journal of human-computer studies* 70, 11 (2012), 888–906. DOI: http://dx.doi.org/10.1016/j.ijhcs.2012.05.010
8. Michael D Byrne. 2001. ACT-R/PM and menu selection: Applying a cognitive architecture to HCI. *International Journal of Human-Computer Studies* 55, 1 (2001), 41–84. DOI:http://dx.doi.org/10.1006/ijhc.2001.0469
9. Michael D Byrne and Elizabeth M Davis. 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 (2006), 627–638. DOI: http://dx.doi.org/10.1518/001872006779166398
10. Gaëlle Calvary, Joëlle Coutaz, David Thevenin, Quentin Limbourg, Laurent Bouillon, and Jean Vanderdonckt. 2003. A Unifying Reference Framework for Multi-Target User Interfaces. *Interacting with Computers* 15, 3 (2003), 289–308. DOI: http://dx.doi.org/10.1016/S0953-5438(03)00010-9
11. Tim Clerckx, Kris Luyten, and Karin Coninx. 2004. DynaMo-AID: A Design Process and a Runtime Architecture for Dynamic Model-Based User Interface Development.. In *Engineering Human Computer Interaction and Interactive Systems*. 77–95. DOI: http://dx.doi.org/10.1007/11431879_5
12. Joëlle Coutaz and Gaëlle Calvary. 2012. HCI and software engineering for user interface plasticity. In *Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications* (3rd ed.), Julie A Jacko (Ed.). CRC Press, Chapter 52, 1195–1220.
13. Klaus-Peter Engelbrecht, Michael Kruppa, Sebastian Möller, and Michael Quade. 2008. MeMo workbench for semi-automated usability testing. In *INTERSPEECH*. 1662–1665.
14. Racim Fahssi, Célia Martinie, and Philippe Palanque. 2015. Enhanced Task Modelling for Systematic Identification and Explicit Representation of Human Errors. In *Human-Computer Interaction – INTERACT*

2015, Julio Abascal, Simone Barbosa, Mirko Fetter, Tom Gross, Philippe Palanque, and Marco Winckler (Eds.). Springer, 192–212. DOI: http://dx.doi.org/10.1007/978-3-319-22723-8_16

15. Juan Manuel González-Calleros, Jan Patrick Osterloh, Rene Feil, and Andreas Lüdtke. 2014. Automated UI evaluation based on a cognitive architecture and UsiXML. *Science of Computer Programming Journal* 86 (2014), 43–57. DOI: http://dx.doi.org/10.1016/j.scico.2013.04.004

16. Christian Gram and Gilbert Cockton (Eds.). 1997. *Design principles for interactive software*. Chapman & Hall, Ltd., London, UK.

17. Wayne D. Gray. 2000. The Nature and Processing of Errors in Interactive Behavior. *Cognitive Science* 24, 2 (2000), 205–248. DOI: http://dx.doi.org/10.1207/s15516709cog2402_2

18. Marc Halbrügge, Michael Quade, and Klaus-Peter Engelbrecht. 2015a. How can Cognitive Modeling Benefit from Ontologies? Evidence from the HCI Domain. In *AGI 2015*, Jordi Bieger, Ben Goertzel, and Alexey Potapov (Eds.). LNAI, Vol. 9205. Springer, Berlin, 261–271. DOI: http://dx.doi.org/10.1007/978-3-319-21365-1_27

19. Marc Halbrügge, Michael Quade, and Klaus-Peter Engelbrecht. 2015b. A Predictive Model of Human Error based on User Interface Development Models and a Cognitive Architecture. In *Proceedings of the 13th International Conference on Cognitive Modeling*, Niels A Taatgen, Marieke K van Vugt, Jelmer P Borst, and Katja Mehlhorn (Eds.). University of Groningen, Groningen, the Netherlands, 238–243.

20. Marc Halbrügge, Michael Quade, and Klaus-Peter Engelbrecht. 2016. Cognitive Strategies in HCI and Their Implications on User Error. In *Proceedings of the 38th Annual Meeting of the Cognitive Science Society*. in press.

21. Laura M Hiatt and J Gregory Trafton. 2015. An Activation-Based Model of Routine Sequence Errors. In *Proceedings of the 13th International Conference on Cognitive Modeling*, Niels A Taatgen, Marieke K van Vugt, Jelmer P Borst, and Katja Mehlhorn (Eds.). University of Groningen, Groningen, the Netherlands, 244–249.

22. Kimberley Hiltz, Jonathan Back, and Ann Blandford. 2010. The roles of conceptual device models and user goals in avoiding device initialization errors. *Interacting with Computers* 22, 5 (2010), 363–374. DOI: http://dx.doi.org/10.1016/j.intcom.2010.01.001

23. Erik Hollnagel. 1998. *Cognitive reliability and error analysis method (CREAM)*. Elsevier, Oxford, UK.

24. Bonnie E John, Evan W Patton, Wayne D Gray, and Donald F Morrison. 2012. Tools for Predicting the Duration and Variability of Skilled Performance without Skilled Performers. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, Vol. 56. SAGE Publications, 985–989. DOI: http://dx.doi.org/10.1177/1071181312561206

25. David Kieras. 1997. A Guide to GOMS Model Usability Evaluation using NGOMSL. In *Handbook of Human-Computer Interaction* (2nd ed.), Marting G Helander, Thomas K Landauer, and Prasad V Prabhu (Eds.). North-Holland, Amsterdam, Chapter 31, 733 – 766. DOI: http://dx.doi.org/10.1016/B978-044481862-1.50097-2

26. Barry Kirwan. 1997. Validation of human reliability assessment techniques: Part 2 – Validation results. *Safety Science* 27, 1 (1997), 43–75. DOI: http://dx.doi.org/10.1016/S0925-7535(97)00050-7

27. Simon YW Li, Ann Blandford, Paul Cairns, and Richard M Young. 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 (2008), 314. DOI:http://dx.doi.org/10.1037/a0014397

28. Quentin Limbourg, Jean Vanderdonckt, Benjamin Michotte, Laurent Bouillon, and Victor Lopez-Jaquero. 2005. USIXML: A Language Supporting Multi-path Development of User Interfaces. In *Engineering Human Computer Interaction and Interactive Systems*, Remi Bastide, Philippe Palanque, and Joerg Roth (Eds.). LNCS, Vol. 3425. Springer, Berlin, 200–220. DOI: http://dx.doi.org/10.1007/11431879_12

29. Joaquin Miller and Jishnu Mukerji. 2001. *Model Driven Architecture (MDA)*. Technical Report ormsc/2001-07-01. Object Management Group, Architecture Board ORMSC. http://www.omg.org/cgi-bin/doc?ormsc/01-07-01.pdf

30. Giulio Mori, Fabio Paternò, and Carmen Santoro. 2004. Design and Development of Multidevice User Interfaces through Multiple Logical Descriptions. *IEEE Trans. Softw. Eng.* 30, 8 (2004), 507–520. DOI: http://dx.doi.org/10.1109/TSE.2004.40

31. Donald A Norman. 2002. *The design of everyday things*. Basic books.

32. Philippe Palanque and Sandra Basnyat. 2004. Task patterns for taking into account in an efficient and systematic way both standard and erroneous user behaviours. In *Human Error, Safety and Systems Development*. Springer, 109–130. DOI: http://dx.doi.org/10.1007/1-4020-8153-7_8

33. Fabio Paternò. 2003. ConcurTaskTrees: An Engineered Notation for Task Models. In *The Handbook of Task Analysis for Human-Computer Interaction*, Dan Diaper and Neville Stanton (Eds.). Lawrence Erlbaum Associates, Mahwah, NJ, 483–501.

34. Fabio Paternò. 2005. Model-based Tools for Pervasive Usability. *Interacting with Computers* 17, 3 (2005), 291–315. http://dx.doi.org/10.1016/j.intcom.2004.06.017

35. Fabio Paternò and Carmen Santoro. 2002. Preventing user errors by systematic analysis of deviations from the system task model. *International Journal of Human-Computer Studies* 56, 2 (2002), 225–245. DOI: http://dx.doi.org/10.1006/ijhc.2001.0523

36. Till Plumbaum, Sascha Narr, Elif Eryilmaz, Frank Hopfgartner, Funda Klein-Ellinghaus, Anna Reese, and Sahin Albayrak. 2014. Providing Multilingual Access to Health-Related Content. In *eHealth – For Continuity of Care: Proceedings of MIE2014*. IOS Press, Amsterdam, NL, 393–397. DOI: http://dx.doi.org/10.3233/978-1-61499-432-9-393

37. Michael Quade. 2015. *Automation in Model-based Usability Evaluation of Adaptive User Interfaces by Simulating User Interaction*. Ph.D. Dissertation. Fakultät IV, Technische Universität Berlin. DOI: http://dx.doi.org/10.14279/depositonce-4918

38. Michael Quade, Marc Halbrügge, Klaus-Peter Engelbrecht, Sahin Albayrak, and Sebastian Möller. 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 (EICS '14)*. ACM, New York, NY, USA, 139–148. DOI: http://dx.doi.org/10.1145/2607023.2607033

39. Michael Quade, Grzegorz Lehmann, Klaus-Peter Engelbrecht, Dirk Roscher, and Sahin Albayrak. 2013. Automated Usability Evaluation of Model-Based Adaptive User Interfaces for Users with Special and Specific Needs by Simulating User Interaction. In *User Modeling and Adaptation for Daily Routines*, Estefanıa Martın, Pablo A Haya, and Rosa M Carro (Eds.). Springer, 219–247. DOI: http://dx.doi.org/10.1007/978-1-4471-4778-7_9

40. Jens Rasmussen. 1983. Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models. *Systems, Man and Cybernetics, IEEE Transactions on* 13 (1983), 257–266. Issue 3. DOI: http://dx.doi.org/10.1109/TSMC.1983.6313160

41. James Reason. 1990. *Human Error*. Cambridge University Press, New York, NY.

42. Rimvydas Rukšėnas, Paul Curzon, Ann Blandford, and Jonathan Back. 2014. Combining human error verification and timing analysis: a case study on an infusion pump. *Formal Aspects of Computing* (2014). DOI:http://dx.doi.org/10.1007/s00165-013-0288-1

43. Dario D Salvucci. 2010. On reconstruction of task context after interruption. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 89–92. DOI: http://dx.doi.org/10.1145/1753326.1753341

44. Mario Sanchez, Ivan Barrero, Jorge Villalobos, and Dirk Deridder. 2008. An Execution Platform for Extensible Runtime Models. In *3rd Int. Workshop on Models at Runtime at MoDELS'08*.

45. Holger Schultheis, Thomas Barkowsky, and Sven Bertel. 2006. LTM C – an improved long-term memory for cognitive architectures. In *Proceedings of the Seventh International Conference on Cognitive Modeling*. 274–279.

46. Jean-Sebastien Sottet, Gaelle Calvary, Joelle Coutaz, and Jean-Marie Favre. 2008. A Model-Driven Engineering Approach for the Usability of Plastic User Interfaces. In *Engineering Interactive Systems*, Jan Gulliksen, MortonBorup Harning, Philippe Palanque, GerritC. van der Veer, and Janet Wesson (Eds.). LNCS, Vol. 4940. Springer, Berlin, 140–157. DOI: http://dx.doi.org/10.1007/978-3-540-92698-6_9

47. Neville A. Stanton. 2003. The Human-computer Interaction Handbook. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, Chapter Human Error Identification in Human-computer Interaction, 371–383. http://dl.acm.org/citation.cfm?id=772072.772097

48. Terrence C Stewart and Robert L West. 2010. Testing for equivalence: a methodology for computational cognitive modelling. *Journal of Artificial General Intelligence* 2, 2 (2010), 69–87. DOI: http://dx.doi.org/10.2478/v10229-011-0010-8

49. Leonghwee Teo and Bonnie E John. 2008. Towards a tool for predicting goal-directed exploratory behavior. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, Vol. 52. SAGE Publications, 950–954. DOI: http://dx.doi.org/10.1177/154193120805201311

50. J Gregory Trafton, Erik M Altmann, and Raj M Ratwani. 2011. A memory for goals model of sequence errors. *Cognitive Systems Research* 12 (2011), 134–143. DOI: http://dx.doi.org/10.1016/j.cogsys.2010.07.010

51. J Gregory Trafton and Raj M Ratwani. 2014. The law of unintended consequences: The case of external subgoal support. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*. ACM, 1767–1776. DOI: http://dx.doi.org/10.1145/2556288.2557422

52. Jean Vanderdonckt. 2008. Model-Driven Engineering of User Interfaces: Promises, Successes, Failures, and Challenges. In *Proc. ROCHI 2008*.

53. Scott D. Wood and David E. Kieras. 2002. Modeling Human Error For Experimentation, Training, And Error-Tolerant Design. In *In Proceedings of the Interservice/Industry Training, Simulation and Education Conference*.