# Evaluating User Interface Adaptations at Runtime by Simulating User Interaction

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

Marco Blumendorf
DAI-Labor, Technische Universität Berlin
Ernst-Reuter-Platz 7, 10587 Berlin, Germany
*marco.blumendorf@dai-labor.de*

Grzegorz Lehmann
DAI-Labor, Technische Universität Berlin
Ernst-Reuter-Platz 7, 10587 Berlin, Germany
*grzegorz.lehmann@dai-labor.de*

Dirk Roscher
DAI-Labor, Technische Universität Berlin
Ernst-Reuter-Platz 7, 10587 Berlin, Germany
*dirk.roscher@dai-labor.de*

Sahin Albayrak
DAI-Labor, Technische Universität Berlin
Ernst-Reuter-Platz 7, 10587 Berlin, Germany
*sahin.albayrak@dai-labor.de*

**Adaptive user interfaces are commonly used for providing different layouts and information according to the current context-of-use. However, the complexity and heterogeneity of potential users, platforms and environments lead to a combinatorial explosion of variants, making it almost impossible to foresee all potential results of adaptations at design time. In this paper we present our work on the automatic evaluation of usability aspects of adaptive user interfaces at runtime which is supposed to be used complementary to design time usability evaluation. We show how a user interface model, providing different adaptation alternatives, can be combined with a model of the current user to simulate interaction and evaluate the feasibility of different adaptations.**

*Automated usability evaluation, Human-computer interaction, User modeling, Adaptive Applications.*

## 1. INTRODUCTION

With the increasing usage of computer systems in all areas of life, the computer evolves from a working tool into a ubiquitous companion. However, this development also entails the usage of computer systems in many different situations throughout our daily lives, where users, platforms and environments (usually summed up as context-of-use) may vary between different interaction cycles. This poses new demands on hard- and software and is often addressed by adaptive applications (Jameson 2008) which aim at providing user interfaces (UI) that are able to adapt to the user (and further context aspects) instead of requiring the user to adapt to the application. The increasing complexity of sensor data and consequently the context information available at runtime allows for even more sophisticated adaptations here, but poses the problem of overviewing and evaluating all possible situations at design time and their impact on the usability and accessibility of the UI. Because users are individuals and different environments and platforms have different characteristics, it becomes almost impossible for the designer to evaluate all (side-) effects of dynamic adaptations to the UI of interactive applications. In summary, this results in an emerging need for dynamic evaluation methods for usability at runtime, complementary to usability evaluation carried out at design time.

In this paper, we present an approach to automatically evaluate different proposed UI adaptations at runtime. Using a running example, we describe required system and user models for an evaluation based on simulated user interaction.

## 2. RELATED WORK

For evaluating adaptive UIs during runtime, an automated usability evaluation (AUE) process is required because a human expert will not be present anymore. Usually, AUE is carried out with computer-aided tools and models of the intended user and the application to be evaluated during an early stage

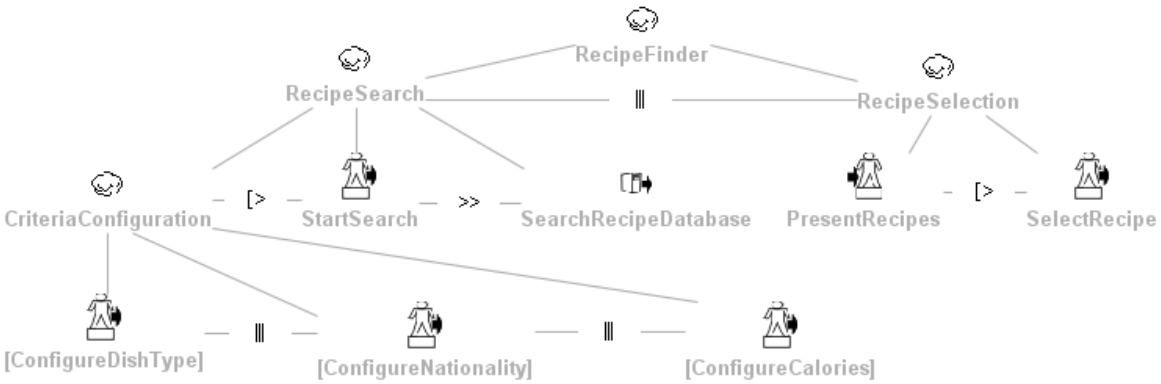


***Figure 1:*** *CTT notation of recipe finder task*

of design time. An overview of AUE methods is given by Ivory and Hearst (2001). Typical results of these AUE methods range from predicting execution times and completion rates of specified tasks, e.g. by using GOMS based methods (John and Kieras 1996), up to estimations of the user's cognitive load, e.g. by using cognitive architectures like ACT-R (Anderson et al. 2004) and EPIC (Kieras and Meyer 1997). Specifically supportive tools which assist the designer in conducting AUE have been proven helpful during design time evaluation. For example CogTool (John and Jastrzembski 2010) creates ACT-R sequences and interaction time predictions from a demonstration of expert user interaction on a mock-up of the UI. However, creating all these different adaptation versions and evaluating all of them in a specific context-of-use, e.g. for different users, is a time consuming task.

Runtime UI adaptation can be achieved by using different methods. Because the approach described below makes use of models of the UI and underlying tasks, we are focusing on model-based development approaches. Sottet et al. (2007) propose UI model mappings for plastic user interfaces (Calvary et al. 2002) augmented with information about their impact on usability properties specified at design time, e.g. according to criteria specified in (Bastien and Scapin 1993). These mappings help the designer to select appropriate UI transformations at design time. During runtime these mappings can be used to negotiate possible adaptations with the user, e.g. via a meta-UI which is a concept used by different systems to provide additional information and interactions. UIDE (Sukaviriya et al. 1993) is a knowledge-based approach for adaptive UIs. The authors propose to trace the user's actions and to redesign menus and add new commands to the UI according to shortest predicted execution times. The SUPPLE system, outlined in (Gajos and Weld 2004), refers to UI adaptation as an optimization problem. SUPPLE focuses on an algorithm for minimizing execution times for users with different motor impairments by building up a user model for motor abilities based on training data.

In (Feuerstack et al. 2008) an AUE is connected to executable models of an application at design time. By executing these models the final UI is simulated and evaluated for a specific user model. The results of the evaluation need to be manually reintegrated into the design process.

## 3. ADAPTATION EXAMPLE

For the purpose of this paper, please consider the example of a senior using a cooking assistant application in a smart home environment. The first step of the cooking assistant, the recipe finder shown as a single screen on top of Figure 2, allows configuring search criteria (left side), a list of search results (right side) and the possibility to confirm a recipe to cook (bottom right). After that the user is guided through the preparation process. The process has been modelled as a task tree compatible with the CTT notation (Paterno et al. 1997) which can be seen in Figure 1.

Initially, the criteria configuration consists of three optional subtasks which are all shown in parallel (see bottom of Figure 1):

1. *ConfigureDishType*: Selection of the desired dish type (e.g. main meal or dessert).
2. *ConfigureNationality*: Selection of the national origin of the recipe.
3. *ConfigureCalories*: Selection of the calories' amount.

In the recipe selection task, the recipe finder then presents the recipes matching the criteria and enables a selection of one recipe. To support the user during interaction more effectively, we consider

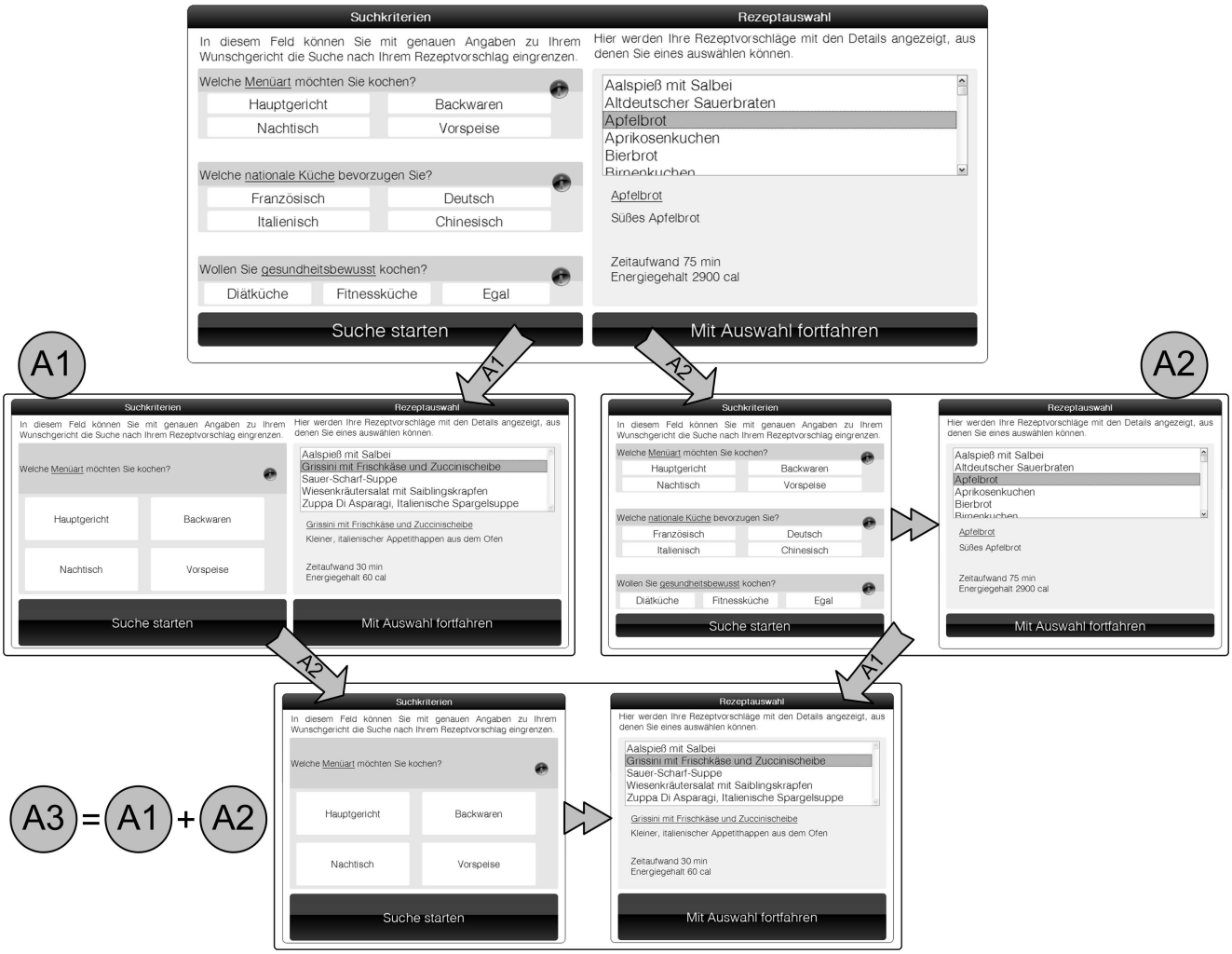


***Figure 2:*** *Recipe finder adaptations A1, A2 and A3*

three predefined adaptations to be performed on the UI. The three adaptations, depicted in Figure 2, can be summed up to:

- A1: reducing the *CriteriaConfiguration* task by removing the optional subtasks. Only the *ConfigureDishType* task is left to simplify the UI by reducing the number of tasks and UI elements, which also gives more space to the remaining elements.
- A2: changing the temporal relation between *RecipeSearch* task and *RecipeSelection* task from interleaving to enabling. This way *RecipeSearch* and *RecipeSelection* become subsequent rather than parallel tasks and are displayed on subsequent screens.
- A3: a combination of A1 and A2. The *CriteriaConfiguration* task is simplified and *RecipeSelection* follows after *RecipeSearch* has been configured. This significantly reduces the complexity of the overall task to be performed by the user.

Applying any (or none) of the above adaptations may be reasonable, depending on the current user (as a subset of the context-of-use). The usability effects of the adaptations may differ e.g. depending on users' cognitive, viewing and motor abilities as well as on their current goals and long-time preferences which all together can usually only be known during runtime.

## 4. MODEL-BASED RUNTIME USABILITY EVALUATION

In this section we describe our approach of combining runtime UI models with user models in order to select the adaptations that fit best based on the results of an automatic evaluation.

The utilization of UI models at runtime provides access to design information of the application and allows an abstract and formal view of the actual UIs implementation. The models can therefore on the one hand be used to control and adapt the UI and on the other hand, which is the focus of this

paper, to evaluate the adapted UI. The models used for this purpose provide comprehensive information about the UI, i.e. details of all visible UI elements and tasks to be performed. In this way, interaction concepts from design time are made available for an AUE at runtime. To support the simulation, the formal description of the UI is complemented by a formal description of the current user. Combining application and user model then facilitates the consequent evaluation of the interaction with the application by simulating interaction steps. In a final step, the utilization of appropriate evaluation criteria allows judging the quality of an adaptation e.g. in terms of task execution times or mean task completion rates. Below, details of the models and the AUE process are described in more detail.

### 4.1. Model-based UI Runtime Architecture

To provide the required models, we utilize a model-based runtime system for ubiquitous UIs, called the Multi-Access Service Platform (MASP, available at http://masp.dai-labor.de) (Blumendorf et al. 2009). MASP UIs are defined by the designer as sets of models compatible with the Cameleon reference framework (Calvary et al. 2002), spanning task and domain models, as well as abstract and concrete UI models. The models are held at runtime to dynamically derive the final UI based on definitions of the designer as well as the runtime state encapsulated within the models. While comprising additional features, we focused on the evaluation based on three models: the task model, the abstract UI model and the concrete UI model.

The tree-like task model basically defines the workflow of an application by describing the tasks that a user (interaction task) and the system (application task) need to perform during their interaction. Interaction and application tasks are leafs of the task tree and grouped by abstract tasks. They are related to each other by temporal operators to specify their order, which is depicted in the example of Figure 1. The MASP interprets this model at runtime to determine the currently active UI elements on a high level of abstraction. Thus every interaction task is refined on the abstract UI (AUI) level with device and modality-independent AUI interactors. Every AUI interactor is further refined in two concrete UI (CUI) models, as it is depicted on the left side of Figure 3. An output model describes the presentation of an AUI interactor, e.g. by graphical interactors like buttons, labels and text fields. The model is enhanced with layout information, describing the spatial relationships between the graphical interactors. A complementing input model describes possible inputs of the user, e.g. keyboard and mouse input, gestures and natural language input.

For our approach the described system models are combined with a user model to support the usability evaluation at runtime. Our user model is based on the concept of the Model Human Processor (Newell 1990) and prior work described in more detail in (Feuerstack et al. 2008) and (Engelbrecht et al. 2009). For building up the user model, it can be initialized with a set of attributes from the current user affecting the simulated interaction. These include static elements like general skills and abilities, psychological factors and dynamic elements like task knowledge. Simulating interaction based on this user model and the different proposed adapted versions of the UI allows e.g. estimating task completion rates for different UI variations and therefore is used to select the most feasible adaptation.

### 4.2. Runtime Adaptation and Evaluation

The adaptation and evaluation process comprehends three steps. At first, the MASP creates copies of the current UI models and then applies different adaptations to each. Afterwards, an AUE process simulates user interaction on each adapted version of the UI. This process is depicted in Figure 3. For this paper, task completion rates are estimated for each adapted UI version of the *RecipeFinder* task. The completion rates are later used as a measure for the criteria of effectiveness. Accordingly, the adaptation with the best completion rate is applied and the resulting UI is delivered to the user in a third step. An outlook on further evaluation criteria is given in section 5.

Following the maximum rationality hypothesis of Newell (1990), the AUE starts with selecting the goals for the simulation run which directly influence the actions to be performed by the user model. In our example the overall goal of every simulation run is the accomplishment of the adapted part(s) of the UI. All needed tasks to fulfil this goal are described in the task model of the UI. Thus the AUE analyses the task model and determines all interaction tasks as they must be accomplished by the user (model) to accomplish the adapted part of the UI. Every interaction task results in one sub goal of the overall goal and is used by the user model to drive the interaction.

In the next step, interaction is started by a simulated perception process. By using the CUI output model, the user model is able to perceive information from the UI elements, e.g. text from labels and layout information from all graphical UI elements. Likewise, the user model is also able to perceive possible interactions with the UI from the CUI input model. The perceived information is used in the subsequent information processing step, when this information

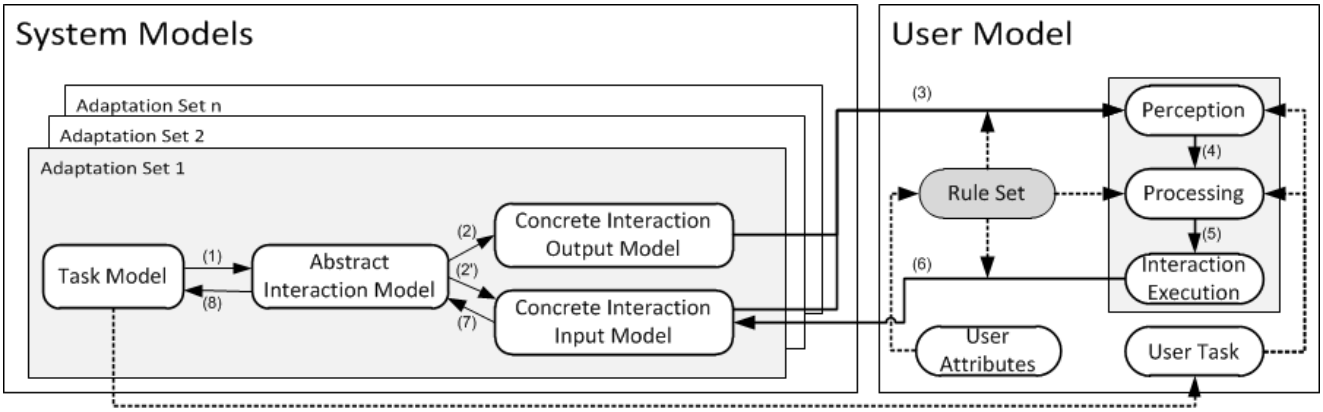


***Figure 3:*** *Instances of the required system models with different predefined adaptations are provided by a model-based runtime infrastructure. The UI models are interconnected with a user model during runtime interaction simulation. A set of rules influences the perception, information processing and interaction execution of the modeled user. Starting from (1) to (8), continuous lines display the chain of actions, while dotted lines display influences between the models.*

is compared to the current goal of the user model according to the principle of label following (Rieman et al. 1996). The processing model also follows the maximum rationality hypothesis and therefore interactions are chosen depending on whether or not they move the interaction process towards the current goal. In the last step, the interaction execution is simulated, e.g. pressing a specific button. In this way interaction between the user model and the CUI input model is performed and both models are updated accordingly.

To simulate the user attributes affecting the interaction process and therefore to generate imperfect interactions, contrary to predefined expert interaction paths as e.g. in CogTool (John and Jastrzembski 2010), we make use of a probabilistic approach based on rules. These rules are applied at all three phases of interaction: perception, information processing and interaction execution. The conditional part of these rules describes trigger points for rules to fire, e.g. the user has a tremor and there is a small button on a touch display. The consequences part describes the effects of the conditions on the simulated interaction process, e.g. reducing the probability for interaction execution by a specific value. After all rules are applied, the probabilities of all input interactions of the current UI are calculated according to the effects of the triggered rules. In a final step, the probabilities of all interactions needed to fulfil the current user goal on the proposed adapted UI versions, are combined to the task completion rate. By simulating semantically equivalent interaction paths on each adapted version of the UI, i.e. the same task, all task completion rates are compared and the best set of adaptations is provided to the user.

General feasibility of our approach for modeling novice users and simulating interaction with user interfaces based on an initial set of rules is explained in (Engelbrecht et al. 2009). In the example depicted in Figure 2 different adaptation strategies are proposed. Given a senior user whose vision is reduced and who has a slight tremor which influences interaction with small UI elements, e.g. buttons on a touch display, simulations with adaptations A2 and A3 outperform the initial UI and version A1. Furthermore, by splitting up the interaction into two separate UIs, less information is transported at a time which allows rendering the remaining UI elements significantly larger. As stated above, interaction simulations conducted with the goal of choosing any recipe from the database have the highest task completion rate in A3 followed by A2. However, when the user's goal is to select a certain recipe, e.g. from Italian cuisine, and the database contains several Italian recipes, A2 suits the user's needs best because A2 allows filtering the database. Further challenges of the evaluation process are discussed in the following section.

## 5. CONCLUSION & OUTLOOK

This paper describes an approach for the dynamic evaluation of UIs at runtime. With the help of UI models and a model of the current user an automated usability evaluation based on simulated interaction determines the suitability of different adaptations at runtime. In this way, several independent adaptations can be proposed and evaluated according to current available information of the context-of-use, which is usually not possible at design time due to its possible complexity. In this paper, we focused on a narrow set of user attributes which affect estimated task completion rates to demonstrate the interconnection of the required models in a runtime evaluation.

There are several directions for future work. At first, we want to evaluate the described approach further and conduct case studies with real participants. We also need to examine the feasibility with a larger set of adaptations, e.g. regarding the performance of the simulation approach with very deep task tree models while keeping the system responsiveness high. A further central aspect is to give human users control over the adaptation process and keep them informed about reasons for specific adaptations. We are aware of the fact that slightly changed environment parameters or changes in the user model might lead to delivering different UIs which might confuse users due to missing consistency between different interaction cycles. To address this issue, we are planning to enhance the meta-UI, already available in the MASP, for giving the user the chance to define and override layout adaptations, consistency parameters and even interaction goals.

Finally, we want to extend the user model with more (dynamic) attributes that have an impact on the simulated interaction and allow deducing different goals for the simulated interaction during the evaluation process, e.g. reducing interaction steps and minimizing cognitive load. For this point, we especially want to analyse different competing evaluation goals and their impact on the UI. By adding further parameters from the user's environment, e.g. noise and lighting parameters, we want to refine the adaptation and evaluation process further.

## 6. REFERENCES

Anderson, J.R., Bothell, D., Byrne, D., Douglass, S.,Lebiere, C. and Qin, Y. (2004) An integrated theory of the mind. Psychological Review 111, pp. 1036-1060.

Bastien, J. and Scapin, D. (1993) Ergonomic Criteria for the Evaluation of Human-Computer Interfaces. Technical report INRIA, No 156.

Blumendorf, M., Lehmann, G., Roscher, D. and Albayrak, S. (2009) Ubiquitous User Interfaces: Multimodal Adaptive Interaction for Smart Environments. In; Multimodality in Mobile Computing and Mobile Devices: Methods for Adaptable Usability, pp. 24-52.

Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Souchon, N., Bouillon, L, Florins, M. and Vanderdonckt, J. (2002) Plasticity of user interfaces: A revised reference framework. In: Proceedings of the First International Workshop on Task Models and Diagrams for User Interface Design, pp. 127-134.

Engelbrecht, K.-P., Quade, M., Möller, S., (2009). Analysis of a New Simulation Approach to Dialogue System Evaluation. Speech Communication, 51, pp. 1234-1252.

Feuerstack, S., Blumendorf, M., Kern, M., Kruppa, M., Quade, M., Runge, M. and Albayrak, S. (2008) Automated usability evaluation during model-based interactive system development. In: Proceedings of the 2nd Conference on Human-Centered Software Engineering and 7th International Workshop on Task Models and Diagrams, pp. 134-141, Springer.

Gajos, K. and Weld, D. (2004) SUPPLE: Automatically Generating User Interfaces. IUI '04: Proceedings of the 9th international conference on Intelligent user interface, pp. 93-100.

Ivory, M. and Hearst M. (2001). The state of the art in automating usability evaluation of user interfaces. ACM Comput. Surv., pp. 470-516.

Jameson, A. (2008). Adaptive interfaces and agents. The Human Computer Interaction Handbook, pp. 433-458.

John, B. and Jastrzembski, T.(2010) Exploration of costs and benefits of predictive human performance modeling for design. In Proc. of International Conference on Cognitive Modeling 2010.

John, B. and Kieras, D. (1996) The GOMS family of user interface analysis techniques: comparison and contrast. ACM Trans. Comp.-Hum. Interact., pp. 320-351.

Kieras, D. and Meyer, D. (1997) An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. ACM Trans. Hum.-Comp. Interact., pp. 391-438.

Newell, A. (1990) Unified theories of cognition. Harvard University Press, Cambridge, MA, USA.

Paterno, F., Mancini, C. and Meniconi, S. (1997) Concurtasktrees: A diagrammatic notation for specifying task models. In Proceedings of Interact'97, Chapman and Hall.

Rieman, J., Young, R. M. and Howes, A. (1996) A dual-space model of iteratively deepening exploratory learning. International Journal of Human-Computer Studies, 44, pp. 743-775, Academic Press.

Sottet, J.-S., Calvary, G., Coutaz, J. and Favre, J.-M. (2007) A model-driven engineering approach for the usability of plastic user interfaces. In Proceedings of Engineering Interactive Systems 2007, Springer.

Sukaviriya, P., Foley J. and Griffith, T. (1993) A second generation user interface design environment: the model and the runtime architecture. In Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 375-382.