Rule-based Approach for Simulating Age-Related Usability Problems

Aaron Ruß[1], Michael Quade[2], Michael Kruppa[1], Mathias Runge[2]

[1]DFKI (German Research Center for Artificial Intelligence) GmbH
{aaron.russ, michael.kruppa}@dfki.de
[2]Technische Universität Berlin, DAI-Labor
{michael.quade, mathias.runge}@dai-labor.de

Abstract. *Ambient Assisted Living* requires easy to use interfaces, making *usability* a critical feature. Because usability evaluations are resource and time consuming, several automation efforts have been made, one of which is the simulation of users interacting with UIs. In this article, we present ongoing work of a tool for automated usability simulations that allows simulating age-related deficits. The tool is specifically intended to be used by IT practitioners, i.e. in difference to cognitive architectures that allow similar simulations, this tool does not require extensive knowledge in cognitive science. A core component of the simulation tool is its rule-based *User Model (UM)*. During a simulated interaction, the UM selects actions causing a model of the UI to change states until a specified task goal is satisfied or the UM "gives up". Interactions of the UM are calculated from probabilities which are informed by rules drawing on user and UI attributes. Using a Monte Carlo approach, the simulation is iterated, resulting in a set of task solutions where non-optimal solutions may indicate usability problems. By analyzing which rules led the UM to interact non-optimally, our approach can offer hints on how to improve the UI. While our approach cannot render user-based evaluations unnecessary, our aim is to substantially reduce the effort involved in usability testing of UIs as well as to provide an automated tool that can be used early on in the development process.

1 Introduction

Ambient Assisted Living (AAL) solutions are often highly distributed and aim at integration into the "natural" environment of their users. As a result, they exhibit a wide variety of *User Interfaces (UI),* e.g. showing information on video screens and ambient displays, or providing control through touch displays and *Speech Dialogue Systems (SDS)*. Easy to use UIs are a central aspect for AAL solutions which makes *usability* a critical feature. But conducting usability evaluations is a time and resource consuming process. This is especially true, if the system under evaluation is aimed to be used by elderly people as in the case of the majority of AAL systems. Not only is the group of elderly users more diverse than younger users – showing significant differences in their sensory, motor and cognitive abilities as well as in their

Ruß, Quade, Kruppa, Runge (2012) *Rule-Based Approach for Simulating Age-Related Usability Problems*. In Wichert, Eberhardt, (Eds.) *Ambient Assisted Living: 5. AAL-Kongress 2012 Berlin*, Germany, January 24-25, Springer Berlin Heidelberg, 149-166
http://dx.doi.org/10.1007/978-3-642-27491-6_11

knowledge and attitudes towards technology – but due to their deficits, lifestyles, and motivations they are also harder to recruit for usability tests (see e.g. [7, 13, 22]).

One way to reduce both costs and effort for evaluations is automated usability testing by model-based evaluations that use simulations [14]. This article describes ongoing work of a rule-based *User Model (UM)* for simulating user interactions that are affected by deficits e.g. *impaired vision* – and general age-related characteristics (e.g. *vision, hearing, tremor, affinity for technology, computer anxiety, domain expertise, cognitive skill*; see [9] for a detailed list) – which can bear significant impact on interactions with UIs. An implementation of our UM is used in the MeMo workbench [8], which is a prototype for rule-based simulations of user interactions for conducting automated usability evaluations. The areas of application include evaluation of classic window-based UIs[1] , web-based UIs and voice-based UIs[2]. By these means, the approach is also intended to be generally feasible to model the use of different interaction devices. Moreover, the workbench is aimed to enable usage by practitioners that are involved in the creation of UIs. Therefore, controlling simulations for (age-related) impairments is intended to be mostly intuitive (e.g. by specifying *"the user has good/bad visual perception"*) and does not require the practitioner to have expert knowledge in cognitive science. Our UM is designed to be as task-independent as possible in order to reduce the cost and effort for its application: to find usability problems in different UIs and with different tasks. Similarly, our model is intended to facilitate the analysis of simulations, e.g. finding user-specific usability issues or even provide critique [14], i.e. give suggestions for improving the UI by highlighting interaction problems.

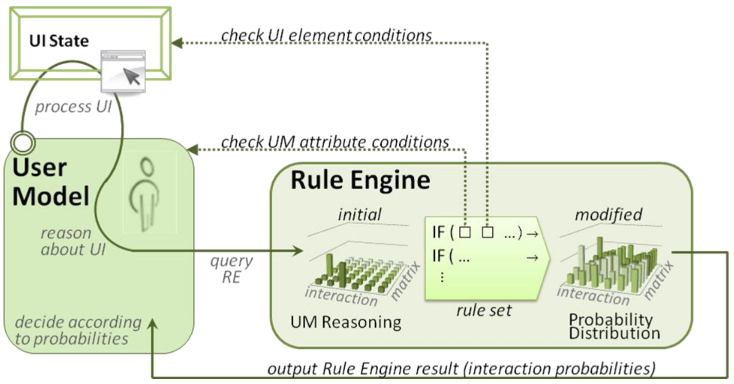


Fig. 1. The general process for employing the rule engine during interaction simulations: The rule engine manipulates the probabilities for potential UM interactions. The UM decides its next step probabilistically according to the distribution. Note that conceptually, the *rule set* is a component of the UM, since the caused probability manipulations are an integral part of the UM's decision making process.

[1] Windows, Icons, Menus, and Pointing devices (WIMP)
[2] Speech Dialog Systems (SDS)

Rule-based usability simulations in MeMo follow a Monte Carlo approach: repeatedly, user interaction is simulated with a given goal of solving a defined user task. Each single iteration of this interaction simulation starts in a specified state of the UI and continues until the task is solved or the UM "gives up". Until then, in each step, probability distributions are calculated in order to decide the next action of the UM and thus causing the UI model to change to the next state. The probability distributions are calculated using a set of rules, in which for example, a rule concerning the *Graphical UI (GUI)* may state that "if button X has a high contrast, then increase its probability-to-be-used". Rules are specified with the help of an *Extensible Markup Language (XML)* syntax, which allows the set of rules to be easily extended and modified.

The simulation results give multiple task-solutions which potentially reveal a broader spectrum of usability problems as compared to a single task-solution, e.g. expert interaction. In addition, the frequency of specific task-solutions is a further indicator for their importance. Finally, by analyzing the applied rules in each iteration step, semantically relevant reasons for the UM's choice can be provided. This allows deriving hints on how to improve the UI.

In [15] former work on the MeMo workbench is described based on a correct solution path for specified user tasks, which had to be provided by the practitioner. In this approach, errors are simulated by automatically introducing deviations from the correct solution path and error corrections are simulated by returning to this correct path. This strategy of producing usability problems shares similarities with approaches described in [20] and [17]. However, it is limited in that only foreseen error types at certain key points in the task solution can be simulated. In difference, the current approach described in this article, computes correct solution paths automatically but only uses them as post hoc measure for the correctness (i.e. determining deviations) and the successful completion of simulated solutions (i.e. reaching the goal condition).

The current concept and architecture of the MeMo workbench were introduced in [8] including a brief description of the rule mechanism and giving a short summary for using the workbench to replicate an experiment in which a SDS was tested. The set of rules was derived from user experiments and literature research. The article [9] describes the modeling of a SDS using the workbench. With addition of a statistical model, experimental data is replicated in simulations. The article also features a short high-level description for the rule mechanism.

In the following article we discuss this rule mechanism of the MeMo Workbench in more detail. First, we will give a short summary of the simulation process (for a more detailed discussion we refer to [9]). Then we describe the rule mechanism illustrated by an example and followed by a discussion.

2 Related Work

Automated usability evaluation is usually carried out with computer-aided tools and models of the intended user and the system to be evaluated. In most approaches the evaluation process consists of a simulation of user interactions while performing specified tasks. With GOMS-based methods [16], evaluation results typically provide *execution time* predictions and *learning rates*. Cognitive architectures [25], e.g. ACT-R [2] and EPIC [18], allow creating more detailed and consistent models of human information processing, e.g. for uncovering *cognitive load*. However, these simulations are highly task dependent and the creation of models usually requires extensive knowledge in the domain of cognitive modeling.

CogTool [25] mitigates the effort involved in developing specialized ACT-R models. A model representation for the user interaction is derived from input sequences which a practitioner demonstrates on a mock-up of the UI. Then CogTool compiles and executes an ACT-R model for expert interaction, i.e. an ideal interaction path for the task is simulated. This primarily allows evaluating the efficiency (*"how long does it take?"*) for the demonstrated task solution. Deviations from the demonstrated interaction path are not considered. This is addressed by an extension to CogTool, the CogTool-explorer [27]. Building on the work of SNIF-ACT [11], CogTool-explorer implements a model for *information seeking behavior* that uses a label-following strategy [21] driven by semantic similarity measures.

Similarly, MeMo [8] addresses the simulation of exploratory user behavior, i.e. simulations of users finding different task-solutions. This allows investigating efficacy *("is it possible to fulfill the task?")* as well as efficiency *("how many steps are needed?")*. UI models are created by the practitioner, which may be based on real systems or early prototypes. MeMo also provides an import-feature for web pages which allows creating the required UI elements automatically, while the interaction logic needs to be added afterwards. Due to the nature of exploratory behavior, time prediction is not an integral feature of MeMo and thus other tools as, for example, CogTool, currently are more appropriate for predicting execution time of tasks.

In [3] a simulator is presented that aims specifically at the evaluation of *assistive* user interfaces by predicting likely interaction patterns for disabled and able-bodied users. The simulation is based on mappings between descriptions of the device space and "knowledge" from the specified user space, which both need to be supplied by the evaluator. Tool support for learning of "first time users" is provided by an interactive simulation process.

Similar to ACT-R, the UM of the programmable modeling approach described in [4], is based on SOAR production rules [19] and follows the assumption that human users interact rationally.

While the simulation of explorative behavior with cognitive architectures aims at highly verifiable models for user strategies, this also demands highly task-dependent models. In difference to these approaches, the work described here is intended to simulate explorative behavior with as little task-dependence as possible. As a result, the simulated user strategy may not match as precisely, but may only approximate the actual user strategy. In addition, parameters of our UM (e.g. for controlling deficits) are more abstract than that of most UMs in cognitive architectures and GOMS-based approaches. Yet, these differences to cognitive architectures allow for a broader area of application and, most of all, further the main goal of our approach: the specific application of uncovering usability problems in UIs by IT practitioners.

3 Rule-based User Simulations

The next sections describe our approach for *interaction simulations*. First, we describe the interaction between UM and UI model and the concepts to achieve a goal driven interaction. Then, we elaborate on the role of *rules* in the simulation: how they are used to model variations in the simulation process based on characteristics of the *user*, *interaction history* and the *UI* under evaluation.

3.1 Simulation process

The interaction process between user model and UI model is primarily driven by concepts of *speech act* theory [24]. Specifically, this means that interactions between both models are characterized by *information exchange,* i.e. information is exchanged in a question-answer structure by speech acts of *request* and *inform*. Accordingly, the user model initially requires user task knowledge for this information exchange, which needs to be specified by the practitioner in advance of each simulation. This user task knowledge consists of information particles for a successful accomplishment of the specific user task. Each information particle contains an attribute which specifies the domain of the information and a value which defines a specific assignment to that attribute, as for example the <attribute, value> pair: <*action*, *turn_on*>.

For exchanging user task knowledge, UI elements – e.g. buttons and text fields – are specified in the context of UI states as part of the UI model. These UI elements offer *input* and *output interactions,* e.g. the label of a button represents an *output interaction* and the click on that button is an *input interaction*. By means of these specific interactions, the user model is able to receive information from the UI model and to manipulate the UI model according to its own task knowledge. This procedure is similar to the concept of *label following* [21] which serves as a basis in similar approaches, e.g. [27]. As an illustration for such an interaction process, take a security guard who asks (*request*) for a name and password and letting someone enter only after validating the information that was provided (*inform*). A *login screen* of an application serves an equivalent function by *requesting* users to enter their name and password.

In the modeled interaction process, the UI facilitates the transfer of the requested information by providing necessary UI elements. Accordingly, the UM attempts to exchange user task knowledge with the UI model by employing input interactions which best fit the completion of the task goal in each UI state.

In order to realize this goal-oriented process, the UM enters a *reasoning phase* (see Fig. 1) during each step of the interaction process. In the reasoning phase the UI state's output and input interactions are processed, resulting in a selection of input interactions that the UM determines to further the *task completion*. Then the UM decides which action to take next, based on *probability distributions* (e.g. executing an *input interaction*, *"giving up"*, or *"asking for help"*). The decision is highly influenced by numerous dependencies between attributes of the UM and features of the UI model. These dependencies are handled by employing a rule engine in each interaction step. The rule engine *triggers* specific rules if their conditions are met.

The process of selecting input interactions is iterated until the task has been fulfilled or the UM "gives up", e.g. if the UM is not able to find relevant input interactions that correspond to the user task knowledge in the current UI state of the simulation.

In the next section we describe the structure of these rules and their effect on the interaction process in more detail.

3.2 Rule definition

The *rule engine* calculates probability distributions that represent plausible behavioral choices of the simulated user. The basic idea is to capture typical behaviors of specific user groups in specific situations in a set of rule definitions. These rules are applied to the interaction process and modify the probability that the simulated user behaves one way or the other.

The rules follow a typical IF-THEN schema (see Listing 1): A single rule is defined by a description of a specific situation with regard to a specific user and a running system (*condition*) and a description of the typical reaction of the user when confronted with the situation (*consequence*).

During simulation, the rule engine applies the rules according to their conditions (see Fig. 1). In this process, the UM queries the rule engine which then determines the probabilities to be modified – and also how they are modified – based on the current state: the rule engine checks the rule conditions against the current properties of UI elements and attributes of the UM and applies the probability modifications defined by the rules accordingly.

More specifically, in order to query the rule engine for the current simulation step, the UM provides initial probabilities for the interactions that are applicable in the current UI state (see Fig. 1). These initial probabilities may be, for instance, equally distributed for all interactions, or the UM may have increased the probabilities for some interactions that are considered to match the current task goal. We will not go into further detail about the initial probabilities, since they are part of the UM's reasoning process and not the rule mechanism (see e.g. [9, 26]).

After applying the rules, the rule engine returns a *probability distribution matrix* for all possible user interactions at the given system state of the UI model. This probability distribution is used by the UM to probabilistically select an interaction. Afterwards, the interaction is executed on the UI model and the simulation advances to the next simulation step.

In the remainder of this section, we will describe the structure and definition of rules in more detail. On an abstract level, we differentiate between three types of rules:

1. *Interaction rules* modify the *interaction probability distribution matrix* during simulations. These rules influence the interaction selection of the UM – most rules that are currently used in the MeMo workbench are interaction rules.
2. *History rules* are triggered by events – or sequences of events – during the simulation and are strictly speaking an extension of *interaction rules*. Here, events refer to property values of previous UI states, so rules of this category modify probabilities depending on previous UI states of the simulation.

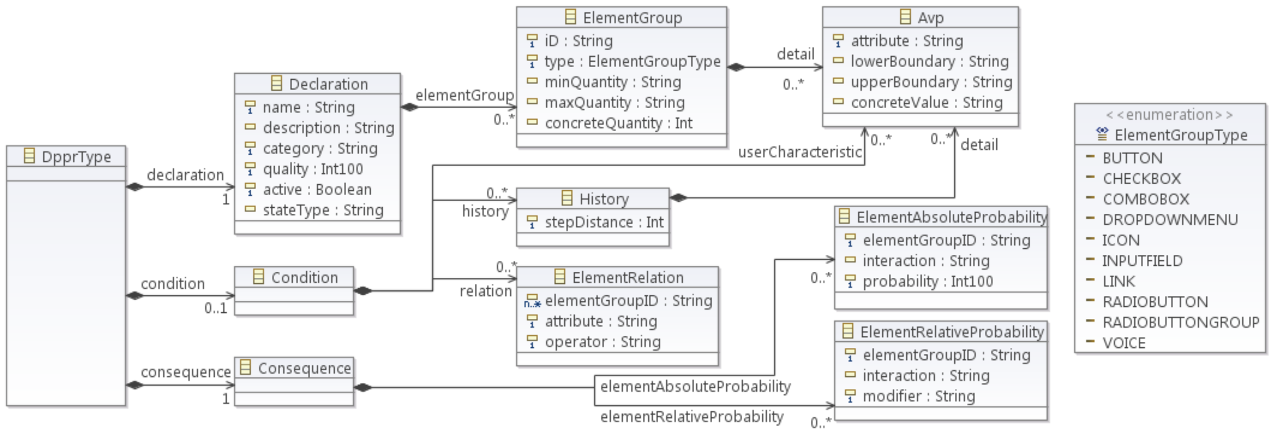
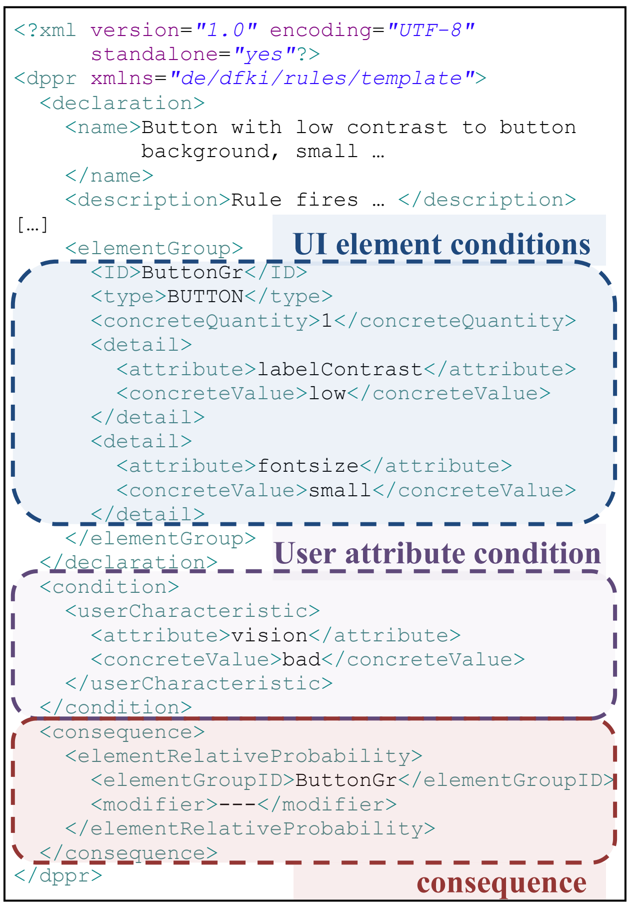


Fig. 2 Schema for XML rule definitions. *DpprType* is the main XML element that contains the rule's *Declaration* (required), *Condition* (optional, see sect. 3.2), and *Consequence* (required) sections.


```
<?xml version="1.0" encoding="UTF-8"
      standalone="yes"?>
<dppr xmlns="de/dfki/rules/template">
  <declaration>
    <name>Button with low contrast to button
          background, small …
    </name>
    <description>Rule fires … </description>
[…]
    <elementGroup>
      <ID>ButtonGr</ID>
      <type>BUTTON</type>
      <concreteQuantity>1</concreteQuantity>
      <detail>
        <attribute>labelContrast</attribute>
        <concreteValue>low</concreteValue>
      </detail>
      <detail>
        <attribute>fontsize</attribute>
        <concreteValue>small</concreteValue>
      </detail>
    </elementGroup>
  </declaration>
  <condition>
    <userCharacteristic>
      <attribute>vision</attribute>
      <concreteValue>bad</concreteValue>
    </userCharacteristic>
  </condition>
  <consequence>
    <elementRelativeProbability>
      <elementGroupID>ButtonGr</elementGroupID>
      <modifier>---</modifier>
    </elementRelativeProbability>
  </consequence>
</dppr>
```
UI element conditions
User attribute condition
consequence

Listing 1 Example for a rule definition in XML format (for more details see Table 2).

3. The third category of rules deals with the number of *information particles* that are selected by the UM for information transfer in the current simulation step. This type of rule is most relevant for SDS where more than one *information particle* can be transferred from user to system by a single *speech act* (i.e. "filling input slots of the SDS").

The structure of rule definitions is specified in an *XML Schema Document (XSD, see* Fig. 2). The top level structure divides a rule into three main sections, namely *declaration*, *condition* and *consequence.*

In the declaration part, general information about the rule is stated. It allows setting a name and a description for the rule. In addition, the relevant objects for the rule's application have to be declared: interaction widgets of a simulated UI form *element-groups* that describe the (group of) UI elements in all necessary details. After declaring element-groups, they can be referenced from the *condition* and *consequence* part of the rule (by the element-group ID). The declaration of element-groups is an implicit condition for their existence. In other words, a rule may only be triggered if – in addition to the other *conditions* – there exists a matching element-group in the current UI state.

The *condition* part limits the execution of rules according to the stated constraints. Conditions may relate to

a) attributes of the UM (*user characteristics*):
   The application of the rule depends on specific values, or a range of values of a user attribute, e.g. that the attribute "vision" has the value "bad".
b) previous events in the course of the simulation-run (*history*):
   The application of the rule depends on specific values or a range of values of properties in previous UI states. For each history condition (event), we need to state (i) how many simulation steps back this condition refers to and (ii) the property in question. The property has to be described in two separate *detail*-statements: one stating the property and value (or range of values) this condition refers to and one stating the element-group i.e. the "owner" of the property. For example, that the rule depends on the fact, that in the previous simulation step there was a SDS *prompt* that articulated a long output text, e.g. a condition referring to the UI prompt property "numberOfSyllables" with "minQuantity" 30.
c) dependencies between element-groups (*relation*):
   The application of the rule depends on the relation between declared element-groups, e.g. that the property "size" of element-group A is *greater* than that of element-group B.

It should be mentioned that the current implementation allows no conditions concerning the non-existence of UI elements, e.g. a rule that depends on the non-existence of a button with certain properties. However, this is no principle restriction of the approach and will be addressed in future developments.

Table 1 Excerpt from the available UI properties and user attributes (only entries relevant for the example in sect. 4 are listed here, for more details see e.g. [9]). The GUI property *labelDistance* is used for checkboxes, and *conventional* for icons. The other GUI properties are used for annotating labels (links, icon-, and checkbox-labels).

Name	Value Range
GUI Properties	
contrast	5 levels
fontsize	5 levels
coding	consistent, inconsistent
colorCoding	true, false
underlineCoding	true, false
boldFontCoding	true, false
graphicalCoding	true, false
layoutGroup	[left,right,upper] navigation, content
labelDistance	high, low
conventional	true, false
User Attributes	
vision	good, bad
domainExpertise	high, low

Finally, the *consequences* part of a rule defines the effects of the rule on element-groups. An effect may be either *relative* or *absolute*. Relative effects manipulate the probability of the referenced element-group based on its current value. In contrast, absolute effects overwrite the current probability value of the element-group. Additionally, the effect may be restricted to a specific interaction of the element-group, i.e. the rule may only modify the probability of the interaction “LeftClick”.

Currently the rule mechanism is implemented with the *Java Rule Engine System (JESS)*. In addition to a basic framework of JESS rules, the XSD rule definitions are compiled into the system internal JESS representation. During simulation, the current context description – i.e. the UI state and UM – are converted into JESS facts to allow the rules access to them.

In difference to the JESS rules, the XSD-based syntax allows for more accessible rules with regard to comprehension and manipulation. For example, the XSD format can be used to automatically generate GUI editors so that practitioners who are unfamiliar with XML can also work with the rules (e.g. using the *Eclipse Modeling Framework (EMF)*, see [1]).

4 Example

In the following section we will describe an artificial example to illustrate the rule mechanism. For this example we use a minimal UI model consisting of one GUI dialog. The dialog contains several GUI elements that were chosen to illustrate the effect of rules caused by various different properties and values (see Fig. 3 and Table

1). No user task knowledge is modeled in order to avoid increased or decreased probabilities for GUI elements due to the UM's reasoning phase: the input for the rule engine is equally distributed and differences in the output reflect solely the influence of the rule mechanism on the probabilities.

As example GUI dialog we use a *Hyper Text Markup Language (HTML)* web page containing various style variations of web links, icons and checkboxes. Fig. 3 shows the automatically imported UI model of the example web page: the type and locations of links, icons and checkboxes are automatically detected and annotated. Other properties of the GUI elements have to be manually annotated. Including some additional modifications, it took us (i.e. skilled users) about 5 minutes to create the final UI model for the example – excluding the time needed to create the HTML web page. Table 1 shows an overview for the GUI properties as well the user attributes that were manipulated when modeling the example. Besides the more obvious modifications and annotations (Fig. 3), we added the text on the right hand side of the 2 checkboxes as their *labels* where the upper checkbox label ("wide distance") has *high labelDistance* and the lower one ("normal distance") a *low labelDistance*. For the icons we marked the one on the left as *conventional (true)* icon and the other icon on the right as *not conventional (false)*.

As UM, we modeled two different user groups – that of stereotypical "young experts" and "older users" – by varying the 2 user attributes *vision* and *domainExpertise*: the group for *Experienced Users* receives *good vision* and *high domainExpertise* while the group for the *Trust Guided Users*, i.e. "old users", get *bad vision* and *low domainExpertise*.

For the rules that are used in this example, we draw on the current set that was developed for the MeMo workbench. The relevant rules that modify the probability distribution in the examples are listed in simplified form in Table 2: If the table row contains effects for both user groups, this implies two rule definitions. Effects that mark a relative *increase* of probabilities are marked with the symbol + and *decreasing* effects are marked with -. The *strength* of the effect is marked using 1 to 3 of the respective symbols for weak, medium, and strong effects. Some attributes in the column *GUI attribute conditions* have two values assigned which is signified by the | separator. In this case, the table row reflects (at least) two rule definitions in which all other attribute-value pairs stay constant for both rules except for the two-valued attribute. Since each row may reflect more than one rule definition, the last table column gives the amount of rule definitions that the corresponding row describes; the table summarizes 35 rules.

The corresponding rule definitions in their XML representation have similar form and complexity as the example rule in Listing 1. The rules described in Table 2 were derived from user experiments for *web site navigating*; they are based on experimental data augmented with expert knowledge (in analogy to [8]). While in principle the rule mechanism is open for MeMo workbench users to modify and create new rules, our goal is to establish a base set of broadly applicable and reusable rules so that users of the workbench do not have to create their own rules (see also sect. 5.2). For this reason, we used preexisting rules and do not report details concerning the time involved in creating the rules.

Table 2 Overview of rules that are used for the example. The columns *Experienced* and *Trust Guided* mark the effect of rules for the respective groups where *vis* + and *exp* + correspond to the user attribute conditions of *vision* = *good* and *domainExprtise* = *high*. Similarly, *vis* - and *exp* - correspond to the conditions *vision* = *bad* and *domainExpertise* = *low*.

Element Type	**GUI Attribute Conditions**		***Experienced*** *vis* + | *exp* +	***Trust Guided*** *vis* - | *exp* -	**Rule Definitions**
Link	colorCoding = true,	layoutGroup = content | left_navigation	+ + +	+	4
	contrast = high,	layoutGroup = content	+ + +	+ +	2
	contrast = low,	layoutGroup = content	-	- - -	2
	contrast = low,	layoutGroup = left_navigation	*NA*	- - -	1
	fontsize = big,	layoutGroup = content | left_navigation	+ + +	+ + +	4
	fontsize = normal,	layoutGroup = content | left_navigation	+ + +	*NA*	2
	fontsize = small,	layoutGroup = content	-	- - -	2
	graphicalCoding = true,	layoutGroup = content | left_navigation	+ + +	+ +	4
	underlineCoding = true,	layoutGroup = left_navigation	*NA*	-	1
	underlineCoding = true,	layoutGroup = content | left_navigation	+ + +	+	4
	contrast = low			- -	1
	contrast = medium			-	1
	fontsize = small		-	- -	2
Icon	conventional = false,	layoutGroup = content	- -	*NA*	1
	conventional = true,	layoutGroup = left_navigation	+ + +	*NA*	1
Check Box	labelDistance = high,	layoutGroup = content	*NA*	- - -	1
	labelDistance = low,	layoutGroup = content	*NA*	- -	1
	layoutGroup = content		- - -	*NA*	1

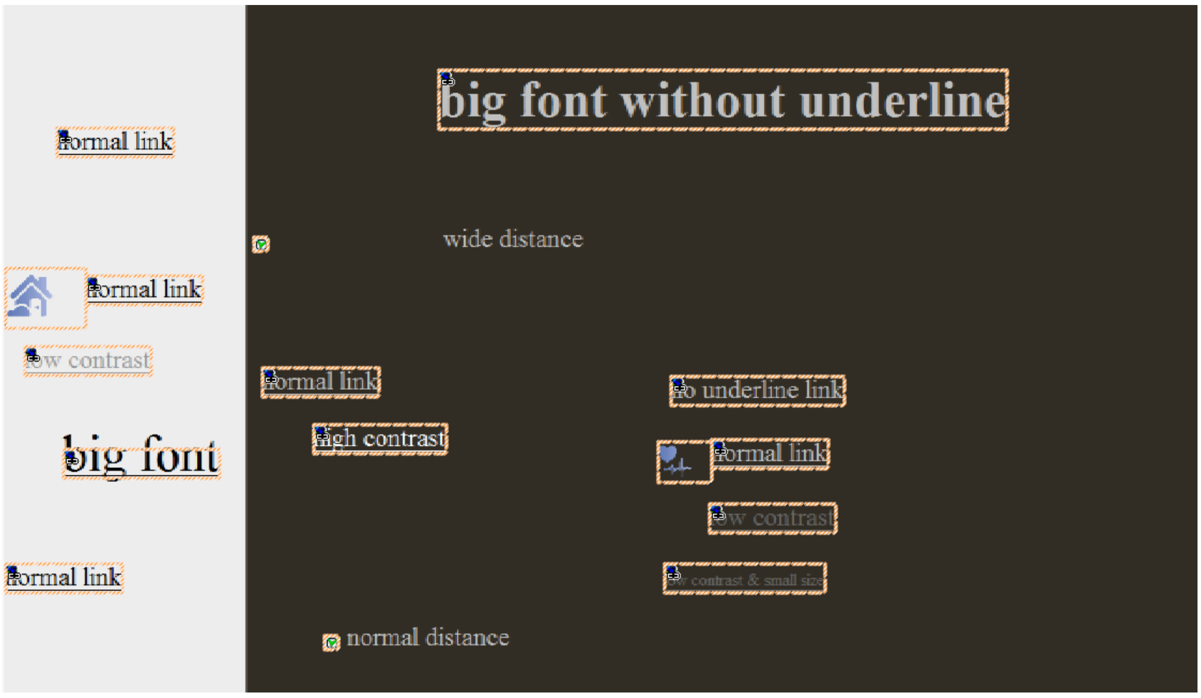
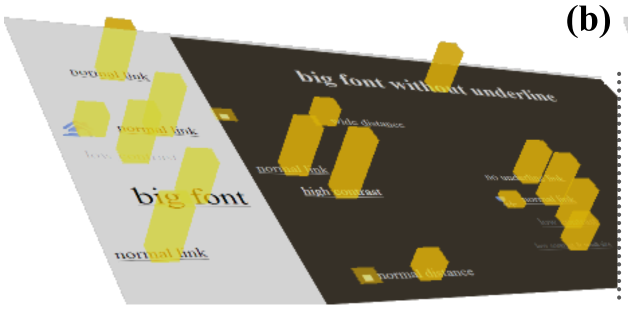


Fig. 3 Automatically imported UI model of a web page with annotated locations for GUI elements (hyper links, icons, checkboxes).

(a)



(b)

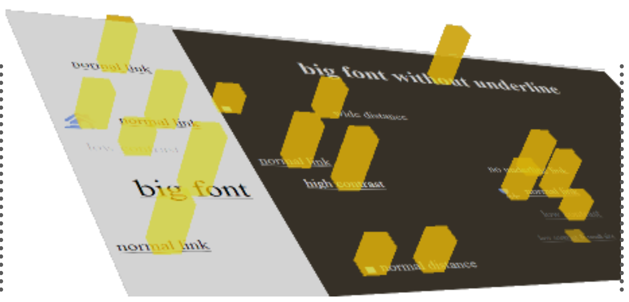


Fig. 4 Probabilities modified by the rule engine superposing a screenshot of the example web page: (a) *Experienced Users*, (b) *Trust Guided Users*. Note that the probabilities for the checkboxes and their labels – as well as for the icons and their labels – are shown separately; the according total probability is the sum of the probabilities of the element and its label.

Fig. 4 shows the probabilities returned by the rule engine displayed over a screenshot of the example web page. The figure shows the probabilities for the icon-links and their labels, as well the checkboxes and their labels separately, i.e. the total probability for the icon-links and the checkboxes is the sum of their own probability and the probability of the corresponding label. In Fig. 4 we can see distinct differences for the probabilities in the lower right group of GUI elements (with low contrast, small font size properties) between the 2 modeled user groups as well as for the 2 checkboxes. The diagram in Fig. 5 highlights the differences between the two modeled user groups with the highest difference on the left. The diagram confirms the impression from Fig. 4, that the modeled “older users” are more likely to leave GUI elements with low contrast and small font size unused and “profit” more from large font sizes, while the “young experts” tend to ignore checkboxes for the navigation task. Also, in difference to the “old users”, the “young experts” make a clear distinction between *conventional* and *unconventional* icons.

The resulting probabilities for *high contrast* links in the *content* area show that “old users” are *less* likely to use them. This difference is caused by the second rule in Table 2. The rule definition is based on the analysis of a web navigation experiment (analogous to [8]) that states that *Experienced Users* are more likely than *Trust Guided Users* to use a high contrast link in the content area. A possible explanation could be that, on the one hand, the visually impaired *Trust Guided Users* may still profit more than the unimpaired group from the high contrast in the sense of perceptual improvement. But, that on the other hand, the expert users, due to their experience, interpret high contrast links as especially important and consequently are more likely to use them.

In summary, the example illustrates that the MeMo workbench supports rapid creation of UI models and UMs reflecting different user groups. Additionally, the rule mechanism applies complex modifications to the interaction probabilities depending on the UI element properties and the UM attributes for the modeled groups.

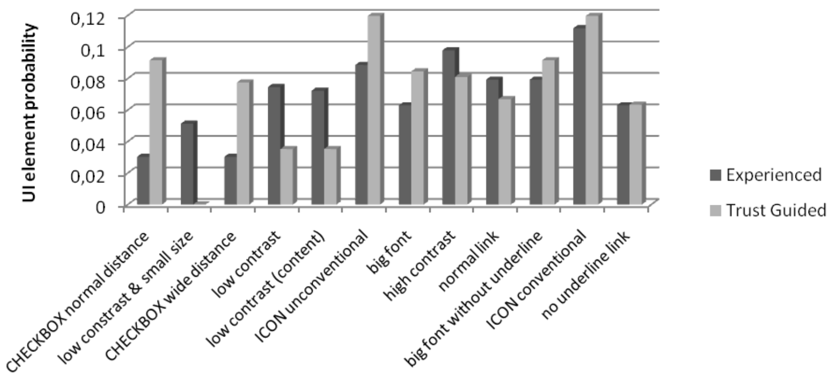


Fig. 5 The resulting probabilities of the two modeled user groups of *Experienced* and *Trust Guided Users* for GUI elements in the example. The GUI element with the highest difference between the groups is to the left, and the rest is ordered correspondingly. When several GUI elements of the same type (e.g. "normal link") have the same probabilities, only one entry is shown in this diagram.

5 Discussion and Future Work

The main intention of the approach described in this paper is to enable IT practitioners to test their applications in an early stage of the development process. Therefore, we describe mechanisms to define and execute rules on simulated user interaction for the purpose of automated usability evaluation. Conditions of these rules take user attributes and UI element properties from the current simulated system state or from previous states into account. Consequences of the rules are used to modify the probability-to-be-used of UI elements.

In [9] the modeling of a software system and replication of observed experimental data gives support for the general feasibility of the described rule mechanism for modeling and predicting usability problems. However, a number of questions concerning the application of the rule mechanism and its advantages need further investigation and are therefore discussed in more detail in the following.

5.1 Description Level of Rules

The description level of the rules is rather abstract: rules operate on (abstract) user attributes and (observable) UI element properties. In terms of the "real world", the rules take *behavioristic observable* properties into account in order to determine their effect.

Still there is some variance for the level of abstraction that directly affects the validity of simulations as well as the kinds of usability questions that can be answered by simulations: rules can draw on low-level user attributes (e.g. the user's *luminance contrast perception*) as well as highly abstract attributes (e.g. the user's *affinity for technology*). Accordingly, analyses of rules that fired during simulation allow only inferences – regarding usability problems and solutions – on the same abstraction level as the rules that were used for the simulation.

Rule mechanisms for similar applications, i.e. for the simulation of exploratory behavior, are mostly employed in the context of *cognitive architectures*. In general, rule mechanisms in cognitive architectures are less abstract. They are mostly concerned with inner cognitive processes, i.e. they operate on and influence inner cognitive properties and variables that are not directly observable. As a result, applying cognitive architectures in the context of usability simulations usually requires very specific and task dependent rules. These specific rules function as hypothesis for user strategies and – if sufficiently verified against experimental data – can provide explanations for user behavior and for the cause of non-optimal user decisions that may signify usability problems.

In contrast, the more abstract rules from our approach relate usability problems to a set of user attributes and UI properties. Here we argue that in the context of uncovering usability problems, this is usually sufficient information to investigate and fix problems with the UI and underlying tasks. Especially under the premise that a high abstraction level makes rules more readily intelligible to non-experts of cognitive science and can thus provide sufficient information for IT practitioners to further investigate highlighted usability issues. This can be achieved by analyzing *aberrant* task solutions (with regard to the optimal solution) and examining the rules that caused aberrations which then provide information for possible usability problems on the same description level as they were specified.

Currently, we also explore, if and how inner cognitive attributes of the UM can be integrated into our rule mechanism. In an experimental implementation, rules can access and manipulate *intentions* and *Dynamic User Attributes (DUA)*. In difference to "normal" static user attributes (e.g. *visual acuity, tremor, education*), DUA (e.g. *attention, irritation, time pressure*) can change their value in the course of the interaction simulation. For instance, a history rule may state that the DUA *irritation* will rise, when in a certain sequence of interactions, the UM fails to find a particular *information particle*. Or, during a simulation, the UM may first have the intention to *accomplish the given task*, but then temporarily change the intention to *ask for help*, due to a rule that checks, if the irritation of the UM surpasses a certain threshold.

5.2 Creating Rule Sets

A basic assumption in our approach is that the complex influences of properties and attributes on the probability distribution can be modeled using a large number of rules which themselves are comparatively simple (see example in Listing 1). The creation of a rule set that reflects complex influences can be managed by iteratively extending the set and adding individual comprehensible rules.

However, with increasing number of rules, their combined effects become harder to judge by practitioners when extending the rule set: given a set of attribute and property values, it becomes harder to foresee the effect of the rules on the probability distribution. For small rule sets or simulations that are intended to approximate user behavior only roughly, this manual process may still be feasible. In order to reach sufficiently faithful and plausible simulations, considerable effort to validate the effect strength of rule definitions is necessary, e.g. by using machine learning for deriving the strength from experimental data. Although we were able to apply the GUI-related rule set and make predictions for 3 UI models in the context of the SmartSenior[3] project, so far we have not been able to validate the GUI-related rule set against experimental results due to lack of usable experimental data from user tests.

A further problem for large rule sets is that modeling the dependencies between properties and between attributes quickly becomes cumbersome. In principle, each possible value of the dependent attribute – or property – requires its own rule definition, potentially resulting in an exponential number of rules for representing the dependency.

There is no tool support yet, but for the most part this could be overcome by allowing the practitioner to specify a condensed definition of the dependency and then use this to compile the necessary rule definitions.

Despite these issues, we propose that the high abstraction level of the rules makes them good candidates to be used and reused in simulations for different UIs: They exhibit a comprehensible syntax and work with generic definitions of UI elements and user attributes, which offer the possibility to adapt them to new areas of application. However, specific criteria that allow deciding if a specific rule can be reused in other simulations still need further investigation.

Due to these considerations, we are planning to establish a base set of validated and reusable rules that are applicable to a wide variety of UIs and that provide reasonable results for uncovering usability problems with these UIs.

5.3 Interpretation and Use of Distributions

In terms of the *Model Human Processor* (*MHP*, see [5]), the use of the rule engine is applied during the *cognitive processing phase*. The modified probability distribution is then directly used to compute the UM's interaction decision during that phase.

Several studies exploring the effect of age in web browsing tasks suggest that the difference between younger and older users is less pronounced in task success but more in completion time and necessary steps [6, 10, 12, 28].

Consequently it seems more likely, that e.g. GUI elements with low contrast have not per se a lower probability to be used, but are less likely to be looked at and therefore evaluated. For instance, such UI elements might not be perceived due to "overlooking" or prematurely selecting another interaction before inspecting all available UI elements. In terms of the probabilistic simulation process, this implies that first a probability distribution for the *perception* is calculated and then one for the *cognitive processing*.

As a work in progress, we extended the MeMo workbench to incorporate three processing phases following the MHP for *perception, cognition, and motion* [23, 26]. As a result, probability distributions calculated by the rule engine employing appropriate sets of rules are used separately in the different phases. This allows modeling sequential dependencies more naturally than using a single probability distribution for calculating the UM's decision making.

[3] http://www.smart-senior.de/enEN/

6 Conclusion

In this article, we presented our ongoing work on model-based automated usability evaluations with the help of the MeMo workbench. We focused on the description of user simulations that are affected by deficits characteristic for old age. The main goal of these simulations is to find usability problems related to these specific needs. Therefore, we have incorporated a rule-based approach which employs user attributes and UI properties in rules for calculating probability distributions. These probabilities are then used to determine user interactions of exploratory behavior. In difference to existing approaches, e.g. cognitive architectures, the rules capture more general aspects of usability knowledge. Accordingly, our approach is less task-dependent and can be transferred to other tasks and even UIs more easily while maintaining reasonable precise predictions about usability problems.

We conclude by asserting that our approach, as well as other existing approaches, cannot replace user testing. Instead, our approach aims to considerably reduce time and effort by enabling early simulations and provide early usability feedback for practitioners during the UI development.

Acknowledgments. This paper presents work of the SmartSenior project funded by the German Federal Ministry of Education and Research (BMBF, FKZ 16KT0902). The authors gratefully acknowledge the implementations and inventions contributed to the MeMo workbench by Klaus-Peter Engelbrecht, Jens Haupert, Britta Hofmann, Marc Hümmer, Maximilian Kern, Dorothea Kugelmeier, Wai-Lung Lee, Stefan Schaffer, Stefan Schmidt, Matthias Schulz, Peter Steinnökel, and Carsten Wirth.

References

[1] Generating an EMF Model using XML Schema (XSD). http://help.eclipse.org/-ganymede/topic/org.eclipse.emf.doc/tutorials/xlibmod/xlibmod.html (May 2008), accessed 26-August-2011

[2] Anderson, J.R., Bothell, D., Byrne, M.D., Douglass, S., Lebiere, C., Qin, Y.: An integrated theory of the mind. Psychological Review 111(4), 1036–1060 (October 2004)

[3] Biswas, P., Robinson, P.: Automatic evaluation of assistive interfaces. In: Proc. 13th International IUI. pp. 247–256. ACM (2008)

[4] Blandford, A., Butterworth, R., Curzon, P.: Models of interactive systems: a case study on programmable user modelling. Int. J. Hum.-Comput. Stud. 60(2), 149–200 (2004)

[5] Card, S., Moran, T., Newell, A.: The psychology of human-computer interaction. L. Erlbaum Associates, Hillsdale, NJ (1983)

[6] Chin, J., Fu, W.T., Kannampallil, T.: Adaptive information search: age-dependent interactions between cognitive profiles and strategies. In: Proc. of the 27th international CHI. pp. 1683–1692. ACM Press, New York, (2009)

[7] Dickinson, A., Arnott, J., Prior, S.: Methods for human-computer interaction research with older people. Behaviour & Information Technology 26(4), 343–352 (2007)

[8] Engelbrecht, K.P., Kruppa, M., Möller, S., Quade, M.: MeMo workbench for semi-automated usability testing. In: Proc. 9th Interspeech. pp. 1662–1665. Australia (2008)
[9] Engelbrecht, K.P., Quade, M., Möller, S.: Analysis of a new simulation approach to dialog system evaluation. Speech Communication 51(12), 1234–1252 (2009)
[10] Fairweather, P.G.: Influences of age and experience on web-based problem solving strategies. In: Stephanidis, C. (ed.) Universal Access in Human-Computer Interaction. Addressing Diversity, LNCS, vol. 5614, pp. 220–229. Springer (2009)
[11] Fu, W., Pirolli, P.: Snif-act: A cognitive model of user navigation on the world wide web. Human-Computer Interaction 22(4), 355–412 (2007)
[12] Grahame, M., Laberge, J., Scialfa, C.: Age differences in search of web pages: The effects of link size, link number, and clutter. Human Factors: The Journal of the Human Factors and Ergonomics Society 46(3), 385 (2004)
[13] Gregor, P., Newell, A.: Designing for dynamic diversity: making accessible interfaces for older people. In: Workshop on Universal Accessibility of Ubiquitous Computing: Proc. of the 2001 EC/NSF workshop on Universal accessibility of ubiquitous computing: providing for the elderly. Association for Computing Machinery, New York, USA, (2001)
[14] Ivory, M., Hearst, M.: The state of the art in automating usability evaluation of user interfaces. ACM Computing Surveys (CSUR) 33(4), 470–516 (2001)
[15] Jameson, A., Mahr, A., Kruppa, M., Rieger, A., Schleicher, R.: Looking for unexpected consequences of interface design decisions: The memo workbench. In: Winckler, M., Johnson, H., Palanque, P. (eds.) Task Models and Diagrams for User Interface Design, LNCS, vol. 4849, pp. 279–286. Springer (2007)
[16] John, B.E., Kieras, D.E.: The goms family of user interface analysis techniques: comparison and contrast. ACM Trans. Comput.-Hum. Interact. 3(4), 320–351 (1996)
[17] Kasik, D., George, H.: Toward automatic generation of novice user test scripts. In: Proc. of the SIGCHI conference. pp. 244–251. ACM (1996)
[18] Kieras, D.E., Meyer, D.E.: An overview of the epic architecture for cognition and performance with application to human-computer interaction. Hum.-Comput. Interact. 12(4), 391–438 (1997)
[19] Newell, A.: Unified theories of cognition. Harvard University Press, Cambridge (1990)
[20] Palanque, P., Basnyat, S.: Task patterns for taking into account in an efficient and systematic way both standard and erroneous user behaviours. Human Error, Safety and Systems Development pp. 109–130 (2004)
[21] Rieman, J., Young, R.M., Howes, A.: A dual-space model of iteratively deepening exploratory learning. International Journal of Human-Computer Studies 44(6), 743–775 (1996)
[22] Rogers, W., Fisk, A.: Toward a psychological science of advanced technology design for older adults. The Journals of Gerontology Series B: Psychological Sciences and Social Sciences 65(6), 645 (2010)
[23] Ruß, A.: Modeling visual attention for rule-based usability simulations of elderly citizen. In: Harris, D. (ed.) Engineering Psychology and Cognitive Ergonomics, LNCS, vol. 6781, pp. 72–81. Springer (2011)
[24] Searle, J.R.: Speech acts: An essay in the philosophy of language. Cambridge University Press (1969)
[25] Sears, A., Jacko, J.A.: The Human-Computer Interaction Handbook. Human Factors and Ergonomics, L. Erlbaum Associates, 2nd edn. (2007)

[26] Steinnökel, P., Scheel, C., Quade, M., Albayrak, S.: Towards an enhanced semantic approach for automatic usability evaluation. In: Proc. of the Computational Linguistics-Applications Conference 2011 (October 2011), accepted

[27] Teo, L., John, B.E.: Cogtool-explorer: towards a tool for predicting user interaction. In: CHI '08 extended abstracts on Human factors in computing systems. pp. 2793–2798. CHI EA '08, ACM, New York, NY, USA (2008)

[28] Wagner, N., Hassanein, K., Head, M.: Review: Computer use by older adults: A multi-disciplinary review. Comput. Hum. Behav. 26, 870–882 (September 2010)