Analysis of a New Simulation Approach to Dialog System Evaluation

Klaus-Peter Engelbrecht, Michael Quade, Sebastian Möller

▶ **To cite this version:**

Klaus-Peter Engelbrecht, Michael Quade, Sebastian Möller. Analysis of a New Simulation Approach to Dialog System Evaluation. Speech Communication, 2009, 51 (12), pp.1234. 10.1016/j.specom.2009.06.007 . hal-00575232

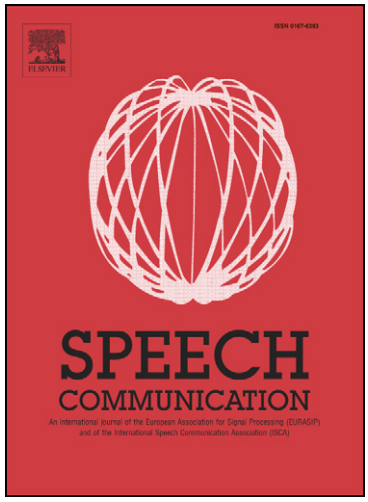
HAL Id: hal-00575232
https://hal.science/hal-00575232

Submitted on 10 Mar 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Accepted Manuscript



Analysis of a New Simulation Approach to Dialog System Evaluation

Klaus-Peter Engelbrecht, Michael Quade, Sebastian Möller

PII: S0167-6393(09)00101-0
DOI: 10.1016/j.specom.2009.06.007
Reference: SPECOM 1820

To appear in: *Speech Communication*

Received Date: 17 December 2008
Revised Date: 18 June 2009
Accepted Date: 24 June 2009

Please cite this article as: Engelbrecht, K-P., Quade, M., Möller, S., Analysis of a New Simulation Approach to Dialog System Evaluation, *Speech Communication* (2009), doi: 10.1016/j.specom.2009.06.007

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

ACCEPTED MANUSCRIPT

Analysis of a New Simulation Approach to Dialog System Evaluation

Klaus-Peter Engelbrecht[1], *Michael Quade*[2], *Sebastian Möller*[1]

[1]Quality and Usability Lab
Deutsche Telekom Laboratories

[2]DAI-Labor

TU Berlin
Ernst-Reuter-Platz 7
D-10587 Berlin
Germany
E-Mails: klaus-peter.engelbrecht@telekom.de; sebastian.moeller@telekom.de;
michael.quade@dai-labor.de

Corresponding author:

Klaus-Peter Engelbrecht
Quality and Usability Lab
Deutsche Telekom Laboratories
TU Berlin
Ernst-Reuter-Platz 7
D-10587 Berlin
Germany
Tel.: +49 30 8353 58486
Fax: +49 30 8353 58409
E-Mail: klaus-peter.engelbrecht@telekom.de

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Abstract

The evaluation of spoken dialog systems still relies on subjective interaction experiments for quantifying interaction behavior and user-perceived quality. In this paper, we present a simulation approach replacing subjective tests in early system design and evaluation phases. The simulation is based on a model of the system, and a probabilistic model of user behavior. Probabilities for the next user action vary in dependence of system features and user characteristics, as defined by rules. This way, simulations can be conducted before data have been acquired. In order to evaluate the simulation approach, characteristics of simulated interactions are compared to interaction corpora obtained in subjective experiments. As was previously proposed in the literature, we compare interaction parameters for both corpora and calculate recall and precision of user utterances. The results are compared to those from a comparison of real user corpora. While the real corpora are not equal, they are more similar than the simulation is to the real data. However, the simulations can predict differences between system versions and user groups quite well on a relative level. In order to derive further requirements for the model, we conclude with a detailed analysis of utterances missing in the simulated corpus and consider the believability of entire dialogs.

Keywords

Evaluation, user simulation, spoken dialog system, usability, prediction model, optimization.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

1. Introduction

Spoken dialog systems (SDSs) are expected to replace simple menu-based interactive voice response systems in the short or medium term. They allow for a natural speech-based human-machine interaction in well-defined domains such as tourist information or control of domestic devices. However, the increasing complexity of such systems also demands for a detailed and economic assessment of system components, and an early evaluation of user-perceived quality already during the system design process. So far, mainly expert evaluation procedures such as Cognitive Walkthrough (e.g. Nielsen, 1993) or Wizard-of-Oz experiments (Fraser and Gilbert, 1991) are applied prior to building a spoken dialog system. Such procedures require usability experts and test users to be available, and result in a specific set of concrete steps to improve the design. In the next iteration of the design, a new evaluation has to be performed, if possible by a second expert who is unbiased by the decisions from the previous iteration.

As Kieras (2003) pointed out, the design workflow can be largely improved by formalizing the requirements of a design in a model of the user, which can be used to test the system design for conformity with the model. The design can then be improved until it satisfies all requirements of the user model, and user- or expert-based testing could be kept for the pre-tested design. In contrast to experts or users, who should not be primed by prior versions of the system or design decisions and therefore have to be exchanged during the iterative design process, a user model can be kept unchanged over several design iterations.

Obviously, the main advantage of automating usability tests is the reduction of effort, money and time involved in evaluations of interactive systems, which eventually leads to a praxis of conducting more studies in their development. According to Nielsen (1993), usability evaluation at early development stages clearly reduces the effort in building a successful interface, as design errors can be eliminated before their effortful implementation. Unfortunately, the effort of such evaluations increases with the complexity of the addressed systems. Tests have to consider more tasks, more different ways to carry out these tasks, and possibly also more user groups. The sheer impossibility of controlling all relevant issues with reasonable resources can lead to deficiencies in user or expert testing, and consequently to unsatisfying interfaces. Automatic usability testing can reduce such effort significantly, and will finally render interfaces more user-friendly and acceptable.

The psychological and the human-computer-interaction (HCI) literature has spawned a number of approaches to model the interaction between users and dialog systems on the level of actions or on the level of cognitive processes involved in operating the interface, e.g. ACT-R (Anderson et al., 2004), SOAR (Newell, 1990), and GOMS (Card et al., 1983). These methods allow HCI experts to specify task features (e.g. user knowledge, interaction steps), while the modeling method itself provides general knowledge about human cognition and behavior. Results from analyses with SOAR, ACT-R or GOMS are typically focused on either execution times or cognitive load, and do not address the generation of higher-level interaction behavior and user-perceived quality.

Though recommended by HCI experts, cognitive models are rarely found in practical application to usability evaluation. Still, a large number of other automated usability methods exists. Ivory and Hearst (2000) categorized 132 such methods according to a taxonomy and showed that these approaches typically automate only some part of the evaluation process (capture, analysis or critique), and are often specific to a narrow class of systems (such as web pages). Speech-based interfaces are not mentioned in their review, however, in recent years, several publications have appeared concerning user simulation techniques for SDSs.

ACCEPTED MANUSCRIPT

1.1 User Simulation for Spoken Dialog Systems

To our knowledge, this topic was introduced by Araki and Doshita (1997), who propose that instead of testing isolated subsystems (e.g. language understanding), the behavior of the entire system should be evaluated. While this could be done with real user tests, they claim that cheaper, quicker, and more objective methods are needed especially during the early development states. In the same year, Eckert et al. (1997) proposed a statistical approach to model the interaction between users and a spoken dialog system. In their model, user and system actions are modeled on the intentional level, and user actions are dependent on the previous system action and the current user state. Possible user actions and their probability are learned from real user data. In addition, parameters describing the users' initiative or patience can be set. Stereotypic user groups can thus be specified by providing distributions of each parameter for the group. This way, various behaviors could be generated, revealing (relatively simple) design errors.

Interactions generated this way could easily be inconsistent, as the user could change its goal during the dialog. Therefore, Scheffler and Young (2001) enhanced this approach by introducing a user goal, specified in the form of attribute-value-pairs (AVPs) and their status (e.g. "specified"). As in the model by Eckert et al., utterances are modeled on the intentional level, however, the selection of an intention is based on a lattice derived from the system grammar. As a consequence, the model is task-dependent.

Pietquin (2006; 2009) describes a new modeling approach which can be formalized as a Bayesian Network. As in Scheffler and Young's model, consistency is ensured by a description of the user's goal as a list of AVPs. Pietquin's concept of user knowledge extends their status description by counting the number of times the system asked for an AVP. In addition, the priority of each AVP can be specified in the goal model. The user actions depend probabilistically on the user goal (incl. priorities), user knowledge, previous system utterance, system speech act, the current state and a model of environmental noise. Thus, the users' behavior can be modeled independent of the task. The model parameters can be learned from a data set or specified by hand by an expert.

While Pietquin's model requires many parameters to be set, Schatzmann (2007a) introduces a simpler model, in which the behavior is elicited according to a stack-like agenda. In the beginning, the agenda is populated with actions derived from the user's goal. The goal (and the agenda) may comprise inform actions and request actions, in which the semantic content is described in the form of AVPs. At each user turn, a number N of actions is popped from the top of the agenda, while a system turn might cause the insertion of a new action, e.g. if a misunderstood AVP is confirmed by the system. The agenda allows the user model to act with initiative, e.g. if the system does not ask for a particular constraint. Schatzmann also proposes a method to learn the model parameters from a set of real user data (Schatzmann 2007b). Schatzman (2007b) compared interaction parameters (dialog length and task success) of the model with either handcrafted parameters or parameters learned from data, and could show that both models lead to similar results, the learned parameters performing slightly better than the handcrafted ones. A similar model was used by Ai and Weng (2008) to evaluate an SDS in the restaurant selection domain.

While these approaches are named "statistical", there are also a few rule-based approaches which have been used for the purpose of evaluating dialog systems. López-Cózar et al. (2003) develop a user model for a dialog system in the domain of fastfood ordering. Actions are selected by a number of rules describing expected behavior for each system question, given a goal and the correctness of the information confirmed by the system. While the aforementioned approaches simulate user behavior on the intention level, López-Cózar collects a corpus of possible utterances for each semantic representation, which is input to the

ACCEPTED MANUSCRIPT

automatic speech recognition (ASR) during the simulation. This way, evaluation of the integrated system (ASR, language understanding, dialog manager) is possible. Two recognition front-ends and two confirmation strategies are compared in a simulated user test, which informed further decisions in the system design process.

Chung (2004) utilizes user simulation to instantiate a system from web content. The simulator mainly serves the purpose of debugging state transitions, enhancing the system's speech and language understanding capabilities, and debugging the automatically generated prompts. To do this, the system makes use of the Genesis speech generator (Seneff, 2002), and an optional text-to-speech (TTS) synthesis. User intentions are simulated by starting with a user query and choosing the next user action at random from the system reply frame, which contains the possible inputs to this prompt and their frequencies. A dialog history is kept to avoid inconsistent behavior. In addition, the user simulation asks for help or repetition with a predefined probability. This way, many different dialogs can be generated. However, the approach focuses on the task of narrowing down search results. The initiative of the user model relies on the information contained in the system reply frame.

As a step towards a general system, Ito et al. (2006) propose a user simulator based on VoiceXML. However, their simulation mechanism only works for a restricted set of systems, where one slot is filled at a time and each slot is explicitly confirmed. Intentions are simulated by choosing a goal at random, considering the VoiceXML description of the system. The user model replies with the AVP which the system asks for, and checks for correct understanding when the slot is confirmed. Corresponding utterances are sampled from the grammar associated with each slot and can be output via TTS. Ito et al. asses the recognizer performance for in-grammar utterances, as well as the stability of the system in case of many parallel calls.

In our previous work (Engelbrecht et al., 2008; Möller et al., 2006), we introduced the MeMo workbench for semi-automatic usability testing. It aims at the development and implementation of a general user simulation framework, applicable to different kinds and classes of systems, such as SDSs and Graphical User Interfaces (GUIs). The aim of the simulation is an analysis of a system's usability from early prototypes to the stage where the system is fully implemented. Thus, like CogTool (John and Salvucci, 2005), it is embedded in a workbench which allows building a model of the interface which is then tested for usability. To our knowledge, in SDS research, such an approach has not been taken so far.

1.2 Requirements to User Simulations

As the evaluation should start before test users are confronted with the system for the first time, the simulation should be independent of interaction data as far as possible. Instead the user model should be derived from the system description, as well as from general knowledge about the users and the task. The user model should produce behavior based on a task description which might differ from the task description of the system, reflecting the notion of a so-called "Mental Model" the user develops when interacting with a system (Norman, 1983). Finally, the behavior of the user model should be interpretable to guarantee the control and understanding of its behavior. Only then the designer can derive the steps to be taken for the improvement of the system.

A general user model, automatically generated and applicable to all kinds of systems, is far out of reach. The system designer needs to provide information about the users' knowledge, their goals, and about their interaction behavior. While intuitively this procedure seems to be logically circular (the designer will always build the system to fit the way she thinks about the users), Kieras (2003) pointed out that a model of the interaction summarizes the design and

ACCEPTED MANUSCRIPT

therefore can be inspected to understand how the design supports the user in performing the task. Furthermore, today's systems' interaction complexity easily exceed the tracing capabilities of designers. This is especially true for SDSs, where the system has to cope with quasi-random speech recognition or understanding errors. In addition, not all requirements for a design can be fulfilled in all cases, and the designer needs to carefully choose which guidelines to prioritize. In such a scenario, sample dialogs generated by a user simulator can provide valuable hints for design decisions. At a later stage, as first user tests have been carried out, simulation can be useful to re-use corpora in the fine-tuning of the system. Still, simulation will not replace user tests completely, but serve as a tool to get the most out of the information available at each design stage.

Still, the more complete the user model is, the more accurate the prediction of usability problems will be from the first draft to later development cycles. Therefore, as a step towards an improved simulator, a major aim of this paper is to describe the weaknesses of our current simulator by analyzing in detail which behavioral traits our user model is capable of, and which are still missing. In previous work on user simulation, as cited above, approaches have been evaluated focusing on the believability of the simulation (in particular, see Schatzmann, 2005; Rieser and Lemon, 2006; Ai and Litman, 2008). In other words, human-likeness of the simulation was assessed. This paper proposes a different path to evaluation, in that the type of behavior observed in the real user experiment is classified and compared to the simulated behavior. In order to draw on rich benchmarking data, we compare simulation results to a corpus of human-machine dialogs, in which the machine – a TV controller in a smart home system – provokes a wide range of user behavior. The task is to choose a TV show, using natural language input, and this is approached by the users in many different ways.

1.3 Outline of the Paper

The paper is structured as follows: In Section 2, we report our progress in building the MeMo workbench, and present the approach for modeling the system and the user. In Section 3, we show how a system can be represented as a model in the workbench, and in Section 4 we present results of our simulation in comparison to those from a real user test with the system. In Section 5, we discuss the results in the light of variance found in real user corpora and identify the features of the simulation which still need improvement. Finally, in Section 6 conclusions are drawn for the further development of the workbench.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

2. User Simulation in the MeMo Workbench

The simulation approach followed in MeMo is similar to the agenda-based approach proposed by Schatzmann et al. (2007a), in that utterances are generated from the description of the task from the user's perspective. Like in the previous approaches to dialog modeling described above, it assumes that the interaction (with any user interface) can be described as a sequence of states in which the system presents an interface and the user uses this interface to perform an action to proceed towards the task goal. However, unlike previous approaches, we assume that the user behavior might be altered by usability problems. If the interface is usable, the user will proceed through the state sequence on a direct path, as intended by the designer. However, if there is a usability problem, the user might choose a wrong action with some probability. Additionally, probabilities for alternative or incorrect actions might be influenced by specific user characteristics.

Deviations from the direct path are sometimes called "user error" despite a clear ambition to improve the interface and not the user to minimize these errors (Bohus and Rudnicky, 2005; Möller et al., 2006; Norman, 1981). Such behavior is taken into account by the simulation in terms of rules. These vary the probabilities for each possible action in a state depending on interface and user features. As another type of error which is special to SDSs, the system may incorrectly understand the users' commands due to restricted capabilities of the automatic speech recognition (ASR) unit and the language understanding unit of the system.

Interactions are simulated with a model of the system, which can be specified by the system designer before the actual implementation of the system using a graphical user interface provided by the MeMo workbench. To enable realistic interactions, the system model has to be annotated with the features which are relevant for user behavior. These features are assigned to the elements of the interface, which we call *User Interface (UI) objects*. This could be a prompt of an SDS or a widget of a GUI.

Features of *UI objects* can be distinguished according to their role in the interaction between the user and the system. On the one hand, *input interactions* determine which information the user model can provide through a *UI object*, e.g. a slot to be filled in reply to a prompt. On the other hand, *output interactions* describe the meaning of labels or texts associated with the *UI object*, e.g. the meaning of a system prompt. *Output interactions* can further be distinguished into *requests* and *informs*. While *requests* describe what the system requests through this *UI object*, *informs* provide the user model with knowledge it might need for the task. For example, they can be used to annotate confirmation prompts with the slots that are confirmed. This way, *output interactions* can determine non-optimal behavior of the user model due to semantic misinterpretations. Finally, features describing the appearance of the *UI object* can be defined. Some of these are calculated automatically, e.g. the length of the system prompt is calculated from a text string. Further features can be freely defined and annotated by hand. These features can determine user behavior due to form and arrangement of the *UI objects*, if there are respective rules.

Performing an *input interaction* on a *UI object* leads to a new system state. This is modeled by a transition from the *UI object* to the successive state. The entirety of states (represented by an interface with several *UI objects*) and transitions is called *system interaction model*. Apart from this, a *system task model* needs to be defined, which specifies the tasks supported by the system.

Such a system task is determined by the information which is conveyed from the user to the system in order to accomplish the task goal. As in previous work on user simulation and dialog management (e.g. Schatzmann, 2007b), the information is specified as a set of

ACCEPTED MANUSCRIPT

semantic concepts, which have the form of an attribute and one of several values that can be assigned to it (attribute-value-pairs, AVPs). This description is not identical with the task goal, as the user might be able to circumvent some input (e.g. by using a shortcut or macro) or neglect a constraint. Therefore, the task goal is specified as a further set of AVPs which are obligatory to be communicated to the system for task fulfillment.

A set of attributes and values consistent with those of the *system task model* is used to describe the *user task model*. That is, if the user has the correct task knowledge, the user task model equals the *system task model*. However, the user's task knowledge can differ from the one of the system (Möller et al., 2007a), which can result in usability problems and should therefore be resolved by the interface.

During the interaction, the AVPs conveyed to the system are tracked in the system interaction model. The user interaction model can specify AVPs through the input interactions of the UI objects of the current state. For example, if the user replies to a system prompt with a semantic concept understandable to the system, the respective AVP will be stored in the system interaction model.

In order to reduce the number of state transitions necessary to describe the system behavior, these can be annotated with conditions and consequences. Both concern the state of AVPs stored in the system. For example, a transition could be conditional on the value for a specific attribute, or as a consequence of a transition a value could be assigned to a particular attribute. A mixed-initiative dialog can then be modeled by conditioning the target state of a transition on the AVPs the user has filled so far. This way, fewer transitions are needed than in a plain state-machine modeling approach.

User Group Editor

Simulations can be run for different groups of users, as was previously done by Eckert et al. (1997). These authors define four very distinct user groups with extreme behavior patterns (e.g. "patient" user hanging up the phone after 99 turns), each group reflecting a single psychological trait. Janarthanam and Lemon (2008) simulate the behavior of users with different domain knowledge in a troubleshooting system. The groups are basically distinguished by asking for extended help messages more or less often.

The user group editor of MeMo tries to go beyond this by allowing the definition of more realistic user groups. Each user group is described by a set of characteristics which are considered to be relevant for user interaction behavior. The user characteristics considered in MeMo have been derived from previous work on the classification of users (Hermann et al., 2007). The cited paper aimed at a user classification scheme which is based on interaction behavior, thus allowing to recruit a comprehensive set of users for usability tests. To do this, the most differentiating characteristics of users were collected at first. Examples are affinity to technology, anxiety, problem solving strategy or domain expertise. In the user group editor, also the user's age and deficits like hearing impairment can be specified.

A user group is defined by assigning a value to each characteristic given in a GUI editor (Figure 1). All characteristics can have a neutral value (or range), which means that this characteristic is not considered in the calculation of the probabilities for the user actions. Like this, the knowledge about users can be formalized in a user group, while not all characteristics have to be known to the person defining the user group.

To generate differentiated user behavior, in addition to the user groups rules need to be defined which describe the relation between the user characteristics and the user behavior. Note that rules are not specified for the groups, but for the characteristics and their values. E.g., a rule might state that users with high anxiety more often fail to utter their request in

ACCEPTED MANUSCRIPT

time. Such relations are much easier to estimate than the behavior of a complex user with many characteristics.

To instantiate a user for an iteration of the simulation, it is sampled from the user group by assigning corresponding values for each characteristic. If a range of values is specified for a group, a concrete value out of that range is selected randomly.

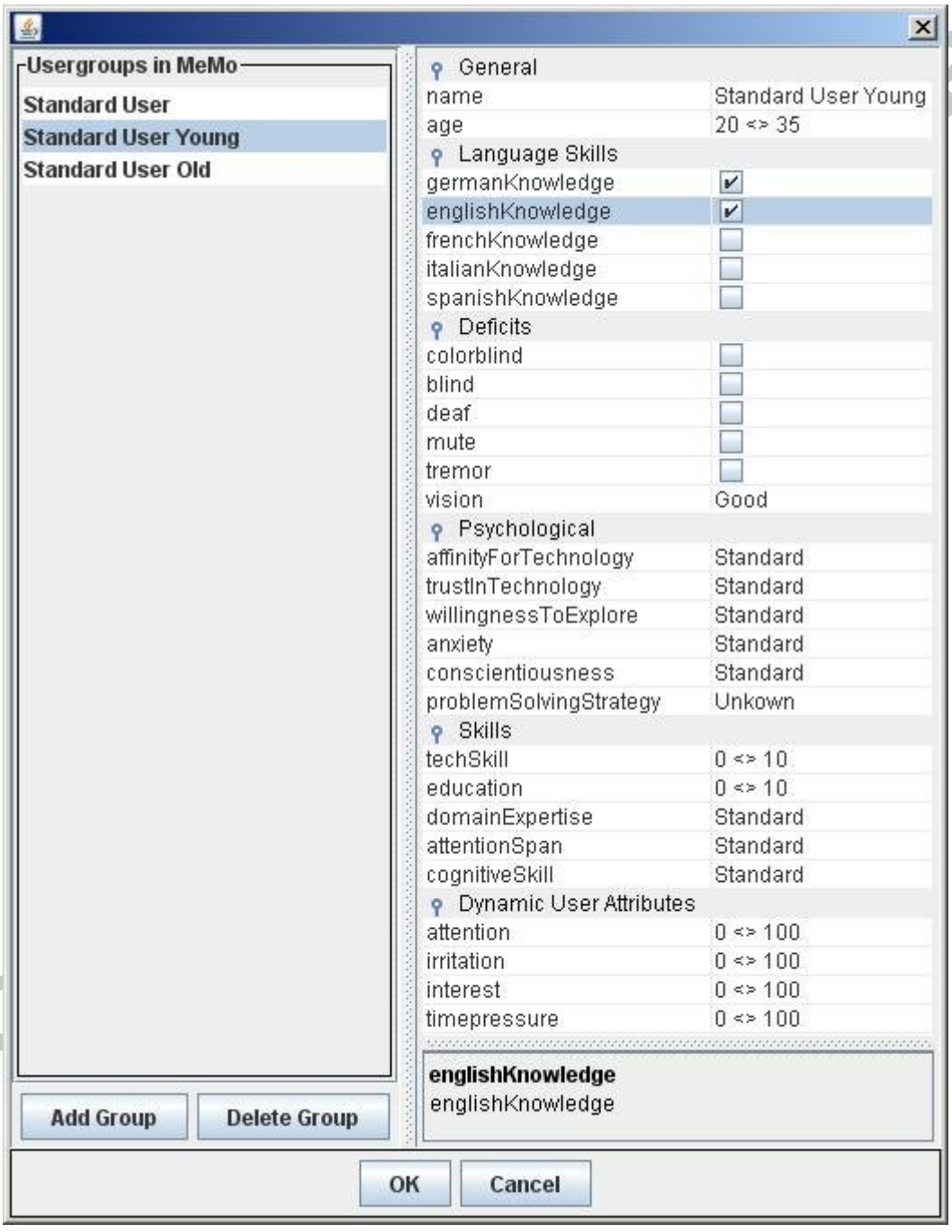


Figure 1. User group editor of the Memo workbench.

Rule Engine

The rule engine evaluates the features of the current system state and the user characteristics, selects the rules which apply, and calculates the resulting probabilities for each possible *input interaction*. Changes in probabilities are specified only approximately, as rules should be as general as possible. A rule can increase or decrease the probability for a specific *input interaction* to one of five fixed levels. Concrete values for each level are specified in a

ACCEPTED MANUSCRIPT

configuration file. This way, by changing the configuration file, the effect of all rules can be adjusted to reflect user behavior as close as possible.

While one could possibly think of finding general rules applicable to all interfaces, or to all interfaces of one class, a designer can also custom-tailor rules for the current simulation in order to modify the behavior of the modeled users. This way, simulations can be run under specified assumptions, such as expected user errors.

A further module integrated into the MeMo workbench simulates speech recognition and understanding errors to be encountered in real-life systems. A number of approaches to modeling such errors have been proposed in recent years. To cite just a few, a simple model by Pietquin and Renals (2002) estimates the error probabilities from the type of recognition task, while a more complex model might take into account acoustic confusability of potential user utterances, as in Schatzmann et al. (2007c). We chose a simple approach, in which deletions, substitutions and insertions of semantic concepts are generated at random to an amount specified in a configuration file. The amount of such errors may also depend on system and user characteristics, such as prompt openness or domain expertise, and can therefore be adapted according to the current situational context and user type during the interaction. In early development phases, before the ASR and language understanding units are fully developed, such an approach can model the knowledge which the developer has about the system and its potential users reasonably well. For later phases, a more complex error model could be integrated into the simulation process.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

3. Conducting an Experiment with MeMo

In order to evaluate our simulation approach, we conducted an experiment with the MeMo workbench and compared the results to those of the same experiment with real users. Figure 2 gives an overview of the procedure, which will be explained in detail in the following sections.

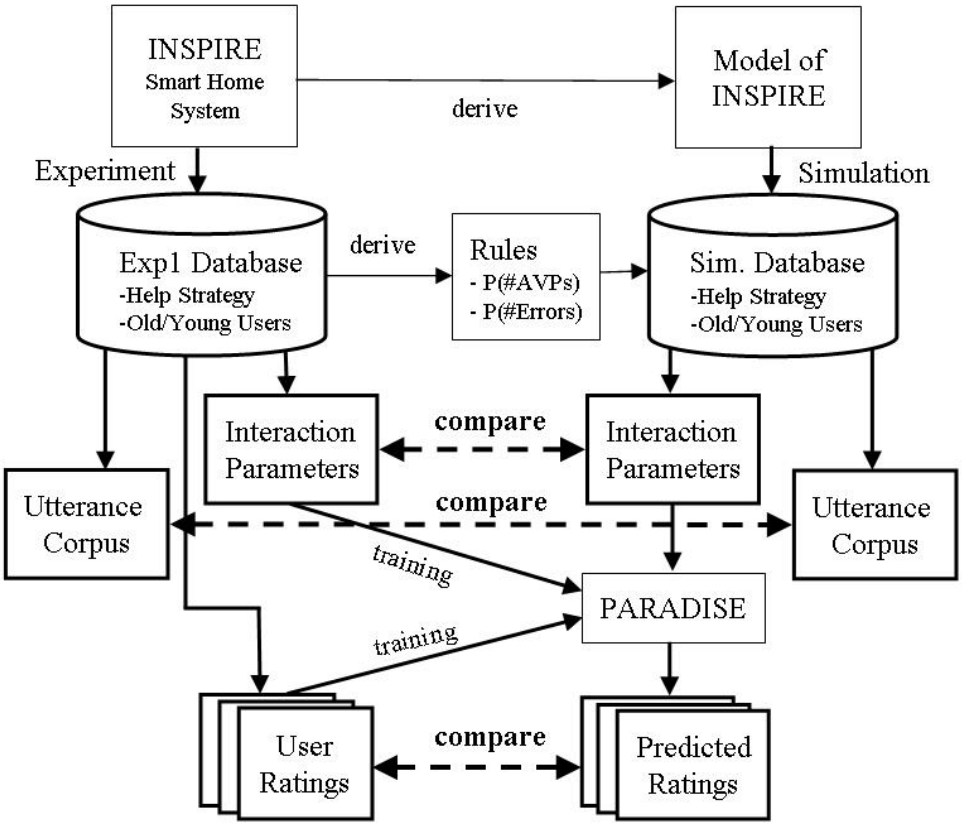


Figure 2. Overview of the Evaluation simulation procedure

3.1 Modeling the INSPIRE Smart-Home System

The smart-home system has been developed in the frame of the EU-funded IST project INSPIRE (INfotainment management with SPeech Interaction via REmote microphones and telephone interfaces; IST 2001-32746). It was designed to control the following domestic devices via speech: 3 lamps, an electronic program guide (EPG), a TV, a video recorder, a fan, and an answering machine.

INSPIRE leads a mixed-initiative dialog with the user to successively fill the necessary slots for a task. The dialog starts with the system question "How may I help you?" The user can reply to this with any set of concepts. In the course of the dialog various errors can occur, which the system can cope with. If it does not understand any concept, a no-match prompt is played back. If only part of the user utterance was understood, the system will re-ask the concerned concept at the appropriate point of the dialog. As a further error, the system can misunderstand a concept, which can result in two values specified for the same concept, e.g., {Device=lamp,TV}. In this case, the system will enter a special state where it informs the user about this conflict, and restricts the reply options to one of these concepts. Also, the system

ACCEPTED MANUSCRIPT

performs an implicit confirmation whenever it has understood a concept. By this, the user knows when she needs to restate a concept.

All these features have been implemented in the system model with the help of states, transitions and their conditions. The original system also features some strategies to cope with the situation that a set of concepts has been acquired for which no solution exists in the database. As the system model does not work on a database, this would have to be hard-coded in the interaction logic of the system. In addition, these system features took effect only in rare cases, so we decided to disregard these features for the model of the system.

We tested our simulation by considering a task with the EPG which our test participants had performed in the experiment. We chose this task because it provoked a wide variety of user behavior including different types of errors. The task was given to the users in the following form:

> *Have a look into the electronic program guide to get an overview of the evening program. Choose from the offers the system proposes on the screen. As soon as you find an interesting movie, ask the INSPIRE system to remind you when it starts.*

The system's internal description of this task contains 8 concepts to choose the device, restrict the movie database and perform an action on the movie.

- TV, program_info, today, evening, movie, *channel_id*, *number*, reminder

Not all of these concepts are necessary for a successful task accomplishment. Once a slot has been specified, the system (model) searches a database of possible task solutions and the corresponding constraints. It then requests the next slot for which more than one value is still possible. If the user for example starts with "remind me on a movie", {reminder} and {movie} will be added as constraints for the database search. As all possible entries contain the constraints {TV} and {program_info}, the system will ask for the day the movie is broadcasted next. The channel concept is left out by the system, because it is redundant, however, when the user names a channel, the system can find a solution for the task.

3.2 Modeling the experiment

In this section we describe the experiment modeled with the workbench. In this test, which was conducted recently with INSPIRE, two conditions - user age and help style - were varied. 32 participants were recruited from an "old" (older than 60 years) and a "young" (20-30 years) population. Both groups interacted with two versions of INSPIRE: The first one provided help only when required (dynamic help), while the second version provided help whenever the user entered a state for the first time (fixed help). By this, the user could always know the reply options and possible wordings, with the cost of longer system prompts. Each user performed two scenarios in this experiment, covering all devices which can be operated with the system. Each scenario consisted of seven tasks. Speech recognition was bypassed with a human transcriber, however, the natural language understanding component of the system was used, so understanding errors could occur.

From the logged dialogs, interaction parameters have been calculated according to ITU-T Supplement 24 (2005). Not all parameters defined in this standard have been used for the evaluation of our simulation. Definitions for the parameters selected by us are given in Table 1. After each scenario, users filled out a questionnaire similar to the one proposed in ITU-T Rec. P.851 (2003), to rate the quality of the system. The questionnaire included a user judgment of the overall impression of the interaction. This judgment was stated on a scale

ACCEPTED MANUSCRIPT

with five points connected by a ruler to suggest equidistance of the points. The points were labeled with words expressing the overall quality from "excellent" to "bad" (Figure 3). This item was collected also for individual tasks during the scenarios, and could thus be used for this study, where only one task of the scenarios was considered.

Abbr.	Name	Definition
CER	concept error rate	Percentage of incorrectly understood semantic units, per dialogue. Concepts are defined as attribute-value pairs (AVPs), with n_{AVP} the total number of AVPs, and s_{AVP}, i_{AVP} and d_{AVP} the number of substituted, inserted and deleted AVPs. The concept error rate can then be determined as follows: $CER = \frac{s_{AVP} + i_{AVP} + d_{AVP}}{n_{AVP}}$
#AVPs	Number of attribute value pairs	Average number of semantic concepts (AVPs) per user utterance $\#AVPs = \frac{n_{AVP}}{\#UserTurns}$
WPST	words per system turn	Average number of words per system turn in a dialogue.
#UserTurns	number of user turns	Overall number of user turns uttered in a dialogue.
#NoMatch	number of ASR rejections	Overall number of ASR rejections in a dialogue. An ASR rejection is defined as a system prompt indicating that the system was unable to "hear" or to "understand" the user, i.e. that the system was unable to extract any meaning from a user utterance.

Table 1. Definitions of parameters used in this study according to ITU-T Supplement 24 (2005).

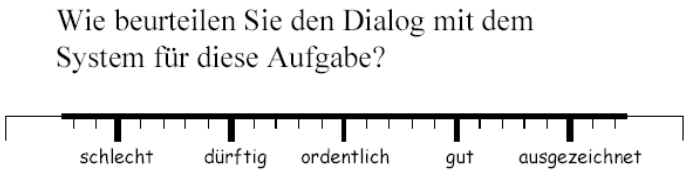


Figure 3. Rating scale for collecting user judgments ("How do you rate the dialog with the system for this task? – bad / poor / fair / good / excellent", see ITU-T Rec. P.851, 2003)

Like Ai and Weng (2008), we trained a PARADISE model (Walker et al., 1997) to predict these user judgments for the simulated dialogs from the interaction parameters. The PARADISE framework assumes that user satisfaction can be modeled as a function of task success and dialog costs in terms of efficiency and dialog quality. Task success and dialog costs can be measured in the form of interaction parameters, as they were described above. The prediction model for the user's satisfaction is obtained by training a linear regression equation on empirical data, using interaction parameters as predictors and subjective judgments of the corresponding dialogs as the dependent variable. We used the task-wise data, which provided 14 vectors of predictors and ratings for each of the 32 users. From the parameters offered to the stepwise inclusion algorithm, three were taken into the model, namely the number of user turns (*#UserTurns*), the average number of words per system turn (*WPST*), and the concept error rate (*CER*).

ACCEPTED MANUSCRIPT

$$Task_rate = -0.37 * \#UserTurns - 0.20 * WPST - 0.16 * CER$$

(R^2_{adj}=0.34; all parameters were normalized to zero mean and unity variance prior to the regression)

Rules for the simulation

Analysis of the experimental data showed that both system version and user group impacted the user behavior. The *CER* was significantly impacted by the help condition (t(60)=-3.387, p=0.001), and to some degree by the users' age (t(60)=-1.829, p=0.072). For the help condition, the effect is very clear as in the fixed-help condition the possible replies were explicitly mentioned each time the user encountered a system question. Principally, these effects should be modeled with the rule engine, however, it currently lacks some features necessary for doing so. Therefore, as a quicker, intermediate-term solution, this functionality was added independent of the rule engine. This allowed us to enter the target number of insertions, deletions and substitutions directly for each combination of user age and the presence of help in the current dialog state, as exemplified in Figure 4. Note, that also in the dynamic help condition help is given sometimes, e.g. embedded in the no-match prompts.

```
# initial error probabilities
DELETE_RATE=0.28
SUBST_RATE=0.010
INSERT_RATE=0.023
# young user group without preceding help prompt
YOUNG_NO_HELP_DELETE_RATE=-0.37
YOUNG_NO_HELP_ SUBST_RATE=-0.37
YOUNG_NO_HELP_INSERT_RATE=-0.37
# old user group after help prompt
OLD_HELP_DELETE_RATE=-0.68
OLD_HELP_ SUBST_RATE=-0.68
OLD_HELP_INSERT_RATE=-0.68
```

Figure 4. Exemplary rules determining the default rates of deletions, substitutions and insertions, and their variation depending on user group and help state.

The other characteristic of the user utterances which shows considerable variance is the number of concepts provided. It significantly depends on the users' age (t(60)=-3.400, p=0.001). We also expected that the number of concepts would vary with the prompt type (free/open/closed). This was quantified by calculating the average number of concepts for each user group and prompt type. The effects could be described with nine MeMo rules. For each prompt type, one rule sets the average likelihood for a specific number of AVPs provided in a user utterance. Two other rules per prompt type modify these probabilities according to the user group.

For the rules to take effect, the system prompts had to be annotated with the respective attributes (prompt-openness and presence of help). The prompts had been specified independent of the dialog states and then been assigned to them. Like this the system configuration with fixed help could be derived rapidly from the original system model by adding the respective help prompt to each state. Then, we custom-tailored two user groups which differed in their age range. In the experiment, age had an effect on the user behavior mainly due to it's correlation with technical affinity and the cognitive abilities. However, we chose to design the simulation according to the experimental design, so we could take advantage of the clear division between the groups with equal participant numbers and bi-polar distribution of the age variable.

ACCEPTED MANUSCRIPT

3.3 Example dialog

In order to illustrate the simulation process we consider an example dialog between the user model and the system model, see Table 2.[1]

The interaction starts with the system asking a free question to the user in step 1. The column "State Features" shows how the prompt is communicated to the user model in the form of features. These comprise the openness of the prompt (free), and an *output interaction* of type *request* (non = the system does not ask for a specific attribute). In addition, the information accepted as a response in this state (all AVPs) is given. As for the feature "prompt_open" the value is "free", a rule with this condition is triggered. The rule engine determines that it is most likely for the user to provide 2 or 3 AVPs in the next utterance, instead of one single AVP which might be the typical answer for a prompt asking for one particular attribute.

Now a random number is generated to determine the number of AVPs to be 3. As no particular AVP is asked for by the system, the three AVPs are randomly selected from the user task knowledge.

In step 2, the system first gives an implicit confirmation, modeled as a set of AVPs in an inform-type output interaction. The number of AVPs in the next utterance is determined to be 1, which is likely because the prompt is now open and not completely free. As the system asks for a specific attribute, the user replies with the value for this attribute. If a second AVP would be specified by the user, this would be selected randomly from the remaining AVPs in the task knowledge which the user still needs to convey. After the utterance has been selected, a speech understanding error is generated by the respective module, resulting in the system understanding "afternoon" instead of "evening".

In the third step, the system again gives an implicit confirmation of what it understood in the previous turn. Thus, the user model can recognize a mismatch between its task knowledge and the presented information. As a consequence, repetition of the misunderstood AVP is set as a priority action. In this turn, the random number generator and the rule engine decided for two AVPs to be uttered by the user. Therefore, the attribute the system has asked for in that turn (TVShowType) is provided by the user as well.

In the subsequent step the system presents the user a program listing for the next evening and requests the number of the entry the user wants to choose. The user replies with the number 4 and asks the system to remind him of this particular show. In this state, the grammar is restricted to recognize only numbers, consequently the AVP {TVAction=reminder} is not recognized.

Finally, in step 5 the user is asked which action she wants to perform on the chosen show, where she transfers {TVAction=reminder} once more because the system explicitly asked for this attribute. After successfully transmitting this AVP, the simulator notices that the system task goal conditions have been met and terminates this simulation run.

[1] The entire simulated dialog corpus can be downloaded from <web address>

ACCEPTED MANUSCRIPT

#	Surface Form	State Features/ User Utterance AVPs	Rules/Remarks
S1:	Welcome to the INSPIRE smart home system. How may I help you?	**Prompt_open**=free **OutputInfo**=none **InputInfo**= All AVPs	If prompt is completely free, the user provides 2 or 3 AVPs rather than 1 or more than 3.
U1:	I'd like to see the TV program for tomorrow.	{device=TV} {action= program_info} {day=tomorrow}	
S2:	I understood TV, program info and tomorrow. For which time would you like to search?	**Prompt_open**=open **OutputInfo**= Request(Time) **OutputInfo**= Inform({device=TV}, {action=program_info}, {day=tomorrow}) **InputInfo**=all AVPs	If prompt is open, the user provides 1 or 2 AVPs rather than 3.
U2:	In the evening.	{Time=afternoon}	Substitution of concept due to a speech recognition error.
S3:	I understood afternoon. What type of program are you interested in?	**Prompt_open**=open **OutputInfo**=request (TVShowType) **OutputInfo**=inform ({time=afternoon}) **InputInfo**=all AVPs	If prompt is open, the user provides 1 or 2 AVPs rather than 3. Mismatch between inform and user task knowledge leads to the time being repeated.
U3:	Please show me shows, but in the evening, please.	{time=evening} {TVShowType=Show}	
S4:	I understood evening and show. Please select a show from the list by saying a number.	**Prompt_open**=closed **OutputInfo**=request (TVShowName) **OutputInfo**=inform ({time=evening}, {TVShowType=Show}) **InputInfo**=TVShowName	If prompt is closed, the user provides 1 AVP rather than 2, and never three or more. Note: Only the TVShowName is a possible input at this state.
U4:	Remind me on number 4.	{TVShowName=4} {TVAction=reminder}	
S5:	I understood number 4. What do you want to do with the show?	**Prompt_open**=open **OutputInfo**=request (TVAction) **OutputInfo**=inform ({TVShowName=4}) **InputInfo**=all AVPs	If prompt is open, the user provides 1 or 2 AVPs rather than 3.
U5:	Set a reminder.	{TVAction=reminder}	Goal conditions met. Simulation stopped.

Table 2. Example dialog illustrating the simulation process in the MeMo workbench.

ACCEPTED MANUSCRIPT

4 Analysis of Simulated Corpora

4.1 Evaluation Criteria

In our analysis of the quality of the simulation, we adhere to Schatzmann et al., (2005), who assessed simulated corpora by comparing them to real user corpora.

On the one hand, Schatzmann et al. analyze "high-level-features", i.e. interaction parameters such as *#Turns*, the turn length quantified as the number of actions per turn, and the user activity quantified as the ratio of system to user actions per dialog. In addition, the dialog style, quantified as the number of speech acts, and the task success are regarded. In Schatzmann's analysis, these measures show a constant decrease with the sophistication of the four user models tested, i.e. they are generally applicable to the assessment of simulations. However, a good fit of these measures does not automatically imply that the simulation was realistic. Like Schatzmann et al., we compare the distributions of the parameters, i.e. apart from mean values the standard deviation (Std), maximum (Max) and minimum (Min), to obtain more detailed results.

The interaction parameters used by us are those presented in Table 1. Our parameters *#UserTurns* and *#AVPs* are basically the same as Schatzmann's #Turns and turn length. In addition, we calculated the parameters needed for the user judgment prediction model, *WPST* and *CER*. *#NoMatch* was calculated because it has often shown a high correlation with user judgments (e.g. Walker et al., 2000). We did not determine the task success, as it is done by Schatzmann et al., because we assume that it cannot be reliably measured in usability experiments, i.e. the "realistic" task success rate is unknown.

Schatzmann et al. furthermore analyze *Recall* and *Precision* of utterances generated by the simulator. This is done on a turn-by-turn basis, feeding each system turn into the simulator, together with the dialog history, and collecting the response of the simulated user. Utterances are modeled in an abstract way as actions compound of a speech act and the information conveyed. When compared to real user utterances of a corpus, *Recall* and *Precision* can be calculated as

$$P = 100 * \frac{Correctly_predicted_actions}{All_actions_in_simulated_response}$$

$$R = 100 * \frac{Correctly_predicted_actions}{All_actions_in_real_response}$$

Again, the simulations can be distinguished by these measures. However, no value range is presented that indicates that a corpus is realistic.

We adopted these two procedures for our evaluation. In reporting our results, we start with the comparison of interaction parameters for complete corpora. We then proceed with the comparison of individual utterances in the corpus. As the workbench does not allow to feed in system states together with their history, we compared the utterances in the simulated corpus with those in the real user corpus. This can be done under consideration of different amount of context of the utterance, where the context is defined by the features of the current system state and the dialog history. If no context is considered, utterances might be judged as correct although they were uttered in a situation where they were inappropriate or did not make sense. On the other hand, if too much context is considered, a great many experimental data are needed to cover a significant number of replies in that situation.

ACCEPTED MANUSCRIPT

We chose to not consider context at first, i.e. we compared the utterances irrespective of the situation they were uttered in ("context-free"). Then, we considered the utterances in the context of the system prompt, i.e. an utterance was considered as equal to another one if the same concepts were replied to the same system prompt ("state-dependent"). No information about the dialog history was included in the context features.

Unlike Schatzmann's approach, utterances in MeMo are represented as concepts, without the addition of the speech act. We abandoned the speech acts as in our data they carry additional information only in very few cases, when the user has to undo a concept. In all other cases, the speech act is to provide information, or if the concept is "yes/no", the speech act is to confirm information.

4.2 Results

High-Level Features

We first discuss *CER* and *#AVPs* (Table 3), which we adapted for the simulation from the experiments. The remaining parameters, discussed below, were not adjusted according to the experimental data, but were derived from *#AVPs* and *CER* by means of our simulation. Therefore, they are not just dependent on these two parameters, but also on characteristics peculiar to our simulation approach, i.e. the hard-coded aspects of the user task and interaction model. Note that *CER* and *#AVPs* were also produced by running multiple iterations of the simulation with the rules describing their statistical distribution across the groups, conditions and system attributes. Consequently, their consideration gives insight in how well given facts can be modeled with the current workbench.

For *CER*, the modeling seems to be adequate. All four configurations were replicated relatively accurately. Also, for *#AVPs* the values for each sub-group were replicated well enough to clearly show the differences between the age groups as well as the equality of the help conditions. Table 5 shows the significance of the effects for the experiment and the simulation. While in the simulation the effect of help condition is significant, the effect size (Partial η^2) is negligible, as it is in the experiment. We want to note here that tweaking this parameter is not as easy as just specifying the average numbers. The number of concepts as output by the rule engine might be chopped if fewer concepts are required for the task. To cope with this, we tuned the user to utter slightly more concepts than required, part of which would be chopped in the dialogs. To model this more accurately, the time dependence of the user behavior needs to be included in the rules.

We then turn to the parameters inferred by the simulation (Table 4). We look here at those parameters which showed to have a significant relation to the perceived quality of the INSPIRE system in the experiments described above. A quick look at the table shows that the values achieved with the simulation differ considerably from those acquired in the experiment, but the relation between the groups is replicated quite well.

We first have a look at the parameter *#UserTurns*. The mean turn number was much lower in the simulation, because some errors with very problematic consequences did not occur. Such errors concern the user's conception of the task rather than utterance wording problems which are modeled with the speech understanding error generator. For example, some users did not understand that they had to provide a time for the system to find an appropriate program. Instead they expected the system to present a schedule of programs including starting times to choose from. Some of these users reset the dialog several times when the system asked for the time. On the other hand, the shortest dialogs we simulated underestimated the number of steps

ACCEPTED MANUSCRIPT

it took even the quickest users to finish the task. However, the differences found between the help conditions are replicated well by the simulation, while in both cases differences between age groups are not significant (Table 5). Thus the simulation would lead to the correct conclusions given this evaluation question.

The next parameter, *WPST*, mainly depends on the system prompts as they are specified in the model. While generally the true system prompts were used, the system features some dynamic prompt generation, which was not explicitly modeled in the workbench. In particular, the system implicitly confirms the concepts it understood from the previous user utterance, by inserting them into a template sentence. In the model, just the template was copied in, so that the word number is always equal to the case of two concepts confirmed. In addition, in the experiment the task occurred sometimes in the middle of the scenario, where the system uses a shortened starting prompt ("What else?" instead of "Welcome to the INSPIRE smart home system. How may I help you?"). Overall, the system model seems to be accurate enough, as in the dynamic help condition the resulting word counts for both user groups are very similar to those in the experiment. However, for the fixed help condition, the word counts are far too high. This leads to a clearly higher effect size of the help condition in the simulation, whereas the age groups do not differ significantly in both the experiment and the simulation.

Parameter		Age group	Help condition	N	Min	Max	Mean	Std
CER	Exp	Young	Fix	16	0.00	0.25	0.11	0.10
			Dyn	16	0.00	0.61	0.23	0.19
		Old	Fix	15	0.00	0.33	0.17	0.09
			Dyn	15	0.11	0.65	0.32	0.18
	Sim	Young	Fix	1000	0.00	1.50	0.10	0.13
			Dyn	1000	0.00	0.73	0.21	0.15
		Old	Fix	1000	0.00	1.50	0.16	0.15
			Dyn	1000	0.00	2.00	0.32	0.16
#AVPs	Exp	Young	Fix	16	1.00	1.67	1.20	0.18
			Dyn	16	1.00	1.80	1.23	0.27
		Old	Fix	15	1.00	2.00	1.44	0.31
			Dyn	15	1.00	2.05	1.46	0.31
	Sim	Young	Fix	1000	0.89	2.67	1.22	0.22
			Dyn	1000	0.89	2.25	1.22	0.21
		Old	Fix	1000	0.92	2.67	1.39	0.29
			Dyn	1000	0.90	2.50	1.42	0.26

Table 3. Comparison of parameters controlled by rules.

To explain the overestimated word counts in the fixed help condition, we refer to the lower number of turns in the simulated dialogs, which were attributed to a smoother dialog. The straight path through the dialog also increased the *WPST*, because the users did not pass a state more than once (as it happened in the experiment), so that all prompts are "seen for the first time". Therefore, all prompts are coupled with a help prompt, which clearly makes them longer on average.

The last parameter examined, *#NoMatch*, is closely related to the *CER*, however, not all concept errors result in a no-match. Also, a no-match normally has a different consequence than a partial non-understanding or a substitution or insertion of a concept. Therefore, the number of times where a concept error leads to each of these consequences would be

ACCEPTED MANUSCRIPT

interesting to know from the simulation. In our simulation, however, the number of no-matches is underestimated, which becomes apparent in the mean and maximum values for each configuration. As the average number of AVPs in the simulated utterances is as high as in the real experiment, we conclude that the understanding errors occurred in clusters in reality. In other words, in reality some utterances were associated with many errors, while some where associated with fewer errors, while in the simulation the errors were equally distributed among the utterances. However, also for this parameter the significant impact of the help condition could be predicted with the simulation. For the user groups, no significant difference was found in the experiment, while in the simulation the difference between age groups was significant. In both cases, the size of the effect is underestimated by the simulation, meaning that the higher significance of simulated results is due to the higher number of cases mainly.

In summary, while the absolute values of the tested parameters differ between the experimental data and the simulation, the relative results are replicated well. Thus, a comparison between age groups or system variants via simulation, (e.g. with the aim of deciding which system to employ for each group) would result in the same decision as the real user experiment.

Parameter		Age group	Help condition	N	Min	Max	Mean	Std
#UserTurns	Exp	Young	Fix	16	5	18	9.5	3.8
			Dyn	16	5	30	12.8	7.7
		Old	Fix	15	5	21	11.1	5.3
			Dyn	15	6	35	15.7	8.2
	Sim	Young	Fix	1000	2	12	5.6	1.2
			Dyn	1000	3	13	6.5	1.6
		Old	Fix	1000	2	13	5.6	1.4
			Dyn	1000	1	15	6.7	1.8
WPST	Exp	Young	Fix	16	12.2	30.1	19.2	5.2
			Dyn	16	10.1	17.6	13.7	1.9
		Old	Fix	15	12.2	27.0	18.3	4.5
			Dyn	15	10.1	19.1	13.6	2.7
	Sim	Young	Fix	1000	20.8	40.0	32.1	2.2
			Dyn	1000	9.6	23.2	15.0	1.7
		Old	Fix	1000	19.0	40.5	31.8	2.8
			Dyn	1000	9.0	25.5	15.3	2.1
#NoMatch	Exp	Young	Fix	16	0	4	0.8	1.1
			Dyn	16	0	7	2.1	2.3
		Old	Fix	15	0	3	1.1	1.1
			Dyn	15	0	10	3.7	2.8
	Sim	Young	Fix	1000	0	5	0.3	0.6
			Dyn	1000	0	6	1.0	1.0
		Old	Fix	1000	0	6	0.5	0.8
			Dyn	1000	0	8	1.4	1.3

Table 4. Interaction parameters found by simulation compared to the same parameters in the real experiment.

ACCEPTED MANUSCRIPT

		Exp			Sim		
	Effect	F(1)	p	Partial η^2	F(1)	p	Partial η^2
CER	Help condition	11.9	0.001	0.170	818.2	0.000	0.170
	Age group	3.9	0.053	0.063	316.6	0.000	0.073
	Help condition* Age group	0.1	0.729	0.002	34.4	0.000	0.009
#AVPs	Help condition	0.2	0.672	0.003	6.7	0.009	0.002
	Age group	11.2	0.001	0.162	557.4	0.000	0.122
	Help condition* Age group	0.01	0.914	0.000	3.7	0.056	0.001
#UserTurns	Help condition	5.7	0.021	0.089	419.1	0.000	0.095
	Age group	1.9	0.168	0.032	3.3	0.070	0.001
	Help condition* Age group	0.2	0.676	0.003	9.6	0.002	0.002
WPST	Help condition	28.1	0.000	0.326	57371.3	0.000	0.935
	Age group	0.2	0.636	0.004	0.2	0.627	0.000
	Help condition* Age group	0.1	0.708	0.002	15.0	0.000	0.004
#NoMatch	Help condition	14.9	0.000	0.204	664.0	0.000	0.142
	Age group	4.0	0.051	0.064	97.8	0.000	0.024
	Help condition* Age group	1.5	0.226	0.025	14.9	0.000	0.004
Ratings	Help condition	0.4	0.512	0.007	3.8	0.052	0.001
	Age group	7.7	0.008	0.117	54.0	0.000	0.013
	Help condition* Age group	1.4	0.239	0.024	23.6	0.000	0.006

Table 5. Statistical significance of experimental results, compared to the significance of simulation results.

User Judgment Prediction

We then used the prediction model introduced in Section 3.2 to predict user judgments from the simulated data for each configuration of the experiment. The results can be seen in Figure 5. The ratings have been normalized to zero mean and unity variance, as the model was trained on normalized data. Displayed are mean and standard deviation for both user groups (line-styles) and help conditions. Displayed on the left-hand side, young users judge the system more positively than old users ($p>0.01$, see Table 5). In addition, old users judged the dialogs with dynamic help worse than those with fixed help. The standard deviations overlap largely in all conditions, and neither the effect of help condition, nor the interaction between the two effects is significant, as can be seen in Table 5. On the right-hand side, the predictions are displayed. As in the real data, young users are estimated to rate the dialogs higher than old users ($p>0.01$), but the effect size is much lower. The difference between the help conditions is underestimated for the old users, and not significant, as in the experiment.

ACCEPTED MANUSCRIPT

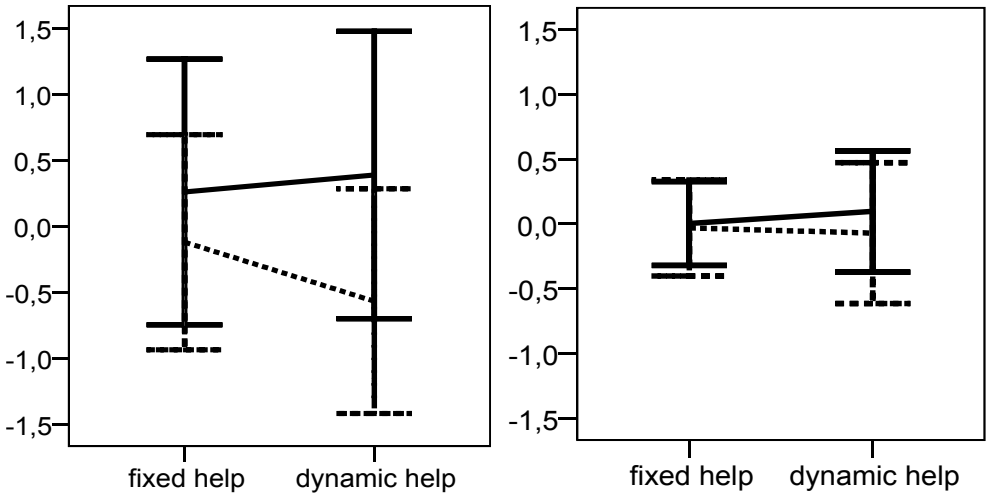


Figure 5. Comparison of user judgments as observed in the experiment or predicted from the simulation. Compared are results for both system configurations, lines representing old (dotted) and young (solid) users.

	Context-free	State-dependent
# correctly predicted turns	25	44
# unique sim. replies	172	554
# unique exp. replies	77	179
Precision	14.5	7.9
Recall	32.5	24.6

Table 6. Overlap between utterances in simulated corpus and the experimental corpus. Context-free means that utterances are compared irrespective of the context they were uttered in. State-dependent means that the previous system question is considered in the comparison.

Precision and Recall

We also compared the utterances in our simulation to those observed from real users (Table 6). We observe a very low *Precision* for both context-free utterances and state-dependent replies. This means that the simulation produces many utterances which we did not observe in the experiments. This is to some degree due to the higher number of unique utterances in the simulation than in the experiment. An analysis of these utterances shows that the biggest part of them is very plausible for the EPG task. However, they seem to be less likely than other utterances, as they did not appear in the experiment.

The *Recall* of utterances from the experiment is also very low, considering the much higher number of different utterances in the simulation than in the experiment. 69.1% of the utterances found with real participants are not produced by the simulation and are thus not available for the improvement of the system.

As expected, the results for utterances in the context of the state they were uttered in stay behind those for context-free analysis. This can be explained by the higher number of possible pairs of utterances and states.

ACCEPTED MANUSCRIPT

5 Discussion

Ai and Litman (2006) analyzed how results like those above are related to the realism of the simulated corpora by comparing not only simulations to real corpora but applying the same analysis also to different real corpora. By comparing interaction parameters and success metrics, they showed that two real corpora (with only slight system changes) differ in the evaluation parameters as well. On the other hand, only very poor simulations with completely random user behavior showed remarkably lower values for the same parameters than did the real corpora. Therefore, they concluded, the measures cannot be used to describe the "reality-level" of simulations.

The task and interactions simulated here are complex enough to trigger a wide variety of different user behavior. Therefore, in this section we benchmark the measures applied above by comparing two real user corpora, the one cited above (called exp. 1 in the following), and a new corpus described below (exp. 2). We then proceed to analyze the simulation in more detail. Utterances which were not produced with the simulation are classified according to the complexity that would have to be added to the simulation in order to produce them. Finally, we look at entire dialogs to assess utterances in this context. By analyzing two examples manually, we avoid the statistical problems arising from the small data set.

5.1. Results for the comparison of real corpora

Experiment 2 data

The second real user corpus stems from an experiment conducted in 2004 with the INSPIRE system at Ruhr-Universität Bochum, Germany. This usability study was designed to explore three different interface metaphors for the INSPIRE system. The experiment was conducted with 24 users (10 f, 14 m) aged between 19 and 29 years (mean: 23.7 years), and recruited from the university environment. Each user interacted with all three metaphors according to three different task scenarios. Two of the interaction scenarios contained exactly the same tasks with the EPG as the experiment described above. However, as a result of the experiment the prompts had been shortened and some vocabulary had been entered into the natural language understanding component. As no significant differences between the interface metaphors were found, data could be merged for our analysis. More details on the experiment are described in Möller et al. (2007b).

High-level features

We first compared the interaction parameters for the two experiments. Only those cases of exp.1 have been considered which were comparable to those in exp. 2, i.e. young users interacting with the dynamic help version of the system. Table 7 shows that the distributions of the parameters differ considerably, although not all differences are significant, which however might be due to the low number of remaining cases from exp. 2. In case of *WPST* ($t(58)=-6{,}31$; $p<0.001$), this is due to the shortening of the prompts after exp. 2 had been conducted. However, no such simple explanation can be given for the differences in *#UserTurns* and *#NoMatch* (both not statistically significant). As we tried to improve the NLU component, we would expect less no-matches in the more recent exp. 1. After all, changes in the system vocabulary were minimal and cannot be the only reason for the mean no-match rate to increase by nearly 50%.

The higher average number of turns in exp.1 interactions could be explained with the (still unclear) increase in no-matches. However, looking at the maxima, an increase in dialog length by ten turns cannot be due to the less than eight no-matches.

ACCEPTED MANUSCRIPT

Parameter		N	Min	Max	Mean	Std
WPST	Exp.1	16	10.1	17.6	13.7	1.9
	Exp.2	44	12.9	26.4	17.8	2.4
#UserTurns	Exp.1	16	5	30	12.8	7.7
	Exp.2	44	4	20	9.4	4.2
#NoMatch	Exp.1	16	0	7	2.1	2.3
	Exp.2	42	0	8	1.4	1.8

Table 7. Comparison of interaction parameters for two experiments with real users.

Precision and Recall

We also analyzed *Precision* and *Recall* of utterances in exp. 2 compared to exp. 1. As for the simulations, all analyses were done once without the utterance context and once with the context of the current state. We used all data of exp. 1 including both user groups and help configurations, as we did not expect a considerable impact on what the users say (we only expected an impact on how they said it). On the other hand, a corpus shrank to 16 dialogs could not be expected to cover a sufficient amount of different utterances to serve as a benchmark.

Table 8 shows that the *Precision* of utterances is considerably higher for a real user experiment than for the simulation. This is mainly due to the lower number of unique utterances in the real user corpus, as the number of utterances which are common between the experimental corpora is not higher. That is, the number of recalled uterances in the simulation is of the same size as in a real user experiment

	Context-free	State-dependent
# correctly predicted turns	25	41
# unique exp.2 replies	64	143
# unique exp.1 replies	79	182
Precision [%]	40	29
Recall [%]	32	23

Table 8. Precision and recall of utterances between two real user corpora.

This result shows a further potential advantage of user simulation, as it exemplifies that user tests cannot capture all relevant user behavior within reasonable effort. Opposed to that, simulation has the potential to generate a comprehensive set of dialogs and utterances inexpensively. A good simulation can therefore not just save time and costs but even improve the reliability of usability tests. However, as a precondition to this, the simulation has to be able to generate all types of user behavior which might come up (cf. also Rieser and Lemon, 2006). We therefore analyzed in more depth the nature of the user behavior which is missing in the simulation. This was done by grouping the utterances of real users which were not generated in the simulation by the reasons underlying their absence. The classification therefore hints at the improvements which are necessary to cover the full range of user behavior observed in exp.1.

ACCEPTED MANUSCRIPT

5.2. Analysis of individual utterances

The following groups resulted from the classification of user utterances not generated in the simulation.

"Mental Model" problem, incl. generation of new AVPs (Table 9)

This group comprises utterances which are caused by errors in the user's understanding of the task, i.e. her Mental Model. 32 % of uncovered utterances belong to this class. Such errors have been discussed in HCI research, which shows their relevance for usability studies and experiments. Therefore, these utterances are highly interesting. They comprise unnecessary actions, mismatch of user intention and system task structure, and incorrect resolution of ambiguities in the task description given to the users. Currently, such errors cannot be simulated with the workbench. Mental Models are constantly developing, not always consistent and difficult to track in experiments. Therefore, their representation is not supported by the current architecture. A future challenge will be to derive knowledge about the user's Mental Model from the system model and general knowledge about the user.

System Turn	User Reply (Concepts)	Example wording	Description
What else can I do for you?	{switch_on}, {TV}	Switch the TV on.	Unnecessary action
What else can I do for you?	{*film_title*}	Show James Bond.	Reference to titles only by list-numbers
What else can I do for you?	Not describable	Switch off the TV after the film is finished.	Unnecessary action
What do you want to do with the program?	{program}, {show}	Please show it.	Correct action "remind" available as separate command
What type of program are you searching for?	{documentary}	A documentary film.	User interpreted "film" as super-category of "movie" and "documentary"

Table 9. Examples for mental model problems not covered by our simulation.

Flexibility in user behavior (Table 10)

In 28% of the utterances not recalled by the simulation, the user deviated from the task, but without the action being erroneous. One variant of this is to loosen a task constraint (e.g. {movie} → {ANY}). In another case, the user would ask for help or repetition of the prompt. The utterance {MORE} may occur when not all possible shows to select from can be displayed on one screen. These utterances will be possible to predict with slight enhancements of our user model.

ACCEPTED MANUSCRIPT

System turn	User reply (concepts)	Example wording	Description
What type of program are you searching for?	{ANY}	Show me all.	No generalization or alternation of concepts
I could not understand you. How may I help you?	{HELP}	What can I do?	Help requests not simulated
Please select a title from the list by saying the number.	{REPEAT}	Please repeat that.	Repetition requests not simulated
Please select a title from the list by saying the number.	{MORE}	Next programs.	Film searched for always on first page of list

Table 10. Examples where users were more flexible than simulated users can be.

Lack of variety in simulated user behavior

These utterances are in accordance with the dialog model utilized in MeMo and are therefore principally possible in the simulation. However, the simulated users did not encounter the respective state or replied something different in case they did. A mere 15% of the utterances not recalled belong to this group. If an infinite number of users would be simulated, these utterances would occur in the simulation as well.

Example: What do you want to do with the TV? – Set a reminder. {reminder}

System feature not implemented in system model

As mentioned in Section 3.2, a few features of the system could not be implemented in the model with reasonable effort. Therefore, some states the real users went through were not present in the system model. However, this group makes only 3.5% of the utterances.

Experimental Artifacts (Table 11)

These utterances occurred in the experiment, but are not related to the actual task given to the users. They make 21.5% of the utterances which were not recalled, and can be attributed to a lack of control over the experimental subjects, who consequently not always acted as foreseen in the experiment. For example, instead of having the start of a movie signaled by the system, some users chose news or had the system record the program. This could lead to states which cannot be reached when the users adhere to the task constraints. For example, selecting {record} leads to an explicit confirmation of the query.

ACCEPTED MANUSCRIPT

System Turn	User Reply (Concepts)	Example wording	Description
What type of program are you searching for?	{news}	News.	Correct: movie
What else can I do for you?	{record}, {movie}	Record a movie.	Correct: remind for a movie

Table 11. Examples for experimental artifacts.

5.3. Analysis of entire dialogs

As was shown in the previous sections, useful knowledge about the simulation can be derived from the comparison of simulated utterances to those from real users. However, to judge the realism of an utterance it should also be considered in the broader context of the dialog. Because of sparse data, this cannot be done statistically. Therefore, in this section two examples of entire dialogs are presented which were generated with our user model. From this, we derive further requirements for a more realistic simulation.

In the first example (Table 12), the user starts the dialog by naming a number, which makes sense as the number describes a specific TV program. However, in the next two steps the user remains in the start state because of a no-match, but changes her intention twice. A real user would rather try to reformulate the sentence than changing her intention, and would be even more unlikely to change her intention twice. Therefore, it seems necessary to take into account previous utterances in the selection of the current utterance, at least in cases of no-match.

System turn	User reply (concepts)	Example wording	Description
How may I help you?	{*number*}	Number one.	Not entirely plausible, but possible.
I could not understand you. How may I help you?	{today}	Today.	User tried a different concept.
I could not understand you. How may I help you?	{program_info} {TV} {reminder}	I would like to search for a film and set a reminder.	Changing the concept twice is unlikely.
I understood TV, program info and reminder. On which day?	…		

Table 12. Example dialog 1.

ACCEPTED MANUSCRIPT

System turn	User reply (concepts)	Example wording	Description
How may I help you?	{reminder}	Set a reminder.	
I could not understand you. How may I help you?	{program_info}	Electronic program guide, please.	
I understood program info. On which day?	{today}	Today.	
I understood today. At which time?	{evening}	After 6 pm	
I understood evening. What type of program are you searching for?	{movie}	Movies.	
I understood movie. Please select a title from the list by saying the number.	{*number*}	Number 1 please.	
I understood number 1. What do you want to do with the program?	{*channel_id*} {TV} {reminder}	Set a reminder for the TV, channel one.	Stating channel name and TV not necessary after program has been selected.

Table 131. Example dialog 2.

In dialog 2 (Table 13), the user makes one rather long utterance after short utterances in the previous states. This was never observed in our experiment. Rather, users would start with longer utterances and from some point onwards stick to providing just the one concept the system has asked for.

In the same dialog, as the TV show has been chosen already, it does not make sense to say the channel in the last utterance. Here, the model is not aware that the bits of information may be relevant to different sub-goals, and at this point the sub-goal of choosing a program has already been accomplished.

Overall, random selection of concepts which are not directly requested by the system seems insufficient for modeling realistic behavior. However, it is difficult to find a rule which concepts make sense at each point in the conversation. E.g., “remind (me on a show)” makes perfect sense as a start of a dialog, while “today” or “evening” does not. A simple, but insufficient guideline would be that after a subtask has been finished, utterances containing constraints for this subtask should not be produced. However, it is difficult even to just describe this behavior formally for a range of systems or tasks, and even more difficult to formulate general rules which generate such behavior automatically.

On the other hand, behavior observed in user tests can be unexpected, and therefore actions as in the examples cannot be ruled out completely. For example, there are no principal restrictions to how fast a user changes her goals after a no-match. This depends on multiple aspects of the system and the user, and even on the interpretation of the observers to some degree. Consequently, such rules are very difficult to acquire in experiments, if they aim at generalizability.

ACCEPTED MANUSCRIPT

In this line, we could argue that over-generation of utterances in the simulation is acceptable if the true interface problems can still be separated from the false ones (or if the false problems can be eliminated without effort). In our simulation we could identify a usability incident which was not observed in the experiment. Here, over-generation was beneficial. In the described case, the user started the dialog by naming the channel. The system then asked what to do with the TV, as the user could still switch the channel or use the program guide. However, the user could also easily choose an incompatible action, like “switch on”, rather than replying “program information” or something similar. Also, a user who has already discovered that the system infers that the EPG is used when she states a time or day might be irritated that after stating a channel name she has to provide this information.

To conclude, while some simulated dialogs are rather unlikely, these can help to identify interface problems of relevance to some users. Therefore, dialogs as shown above should not be eliminated from the simulation. However, an estimation of the probability for dialogs would allow the designer to adjust her focus depending on the available resources.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

6. Conclusions and future work

In the previous sections we have introduced a new approach to simulate user behavior for the sake of usability predictions. Simulations happen on an abstract level between models of the system and the user, in order to facilitate their applicatin during system design. Models can be constructed relatively easily with a graphical editor provided by our workbench environment. A comparison was made between a simulated corpus and an experimental corpus, both datasets covering two user groups and two system versions. The simulated task had shown a wide variety of user behavior in the experiment, making it more difficult to simulate all possible interactions. Despite that, with our simulation we could predict the rank order of the four experimental conditions for different interaction parameters; that is, we were able to predict which system version was rated better by which group of users. In the design lifecycle of a system, such information can be valuable to decide about the interface approach pursued in the future development.

We then looked at precision and recall of the simulated utterances. Many of the utterances in the experiment were not generated by the simulation; however, a closer analysis revealed that 15% of them could be generated without extensions to our models if more users were simulated. Another 28% of the unpredicted utterances could be generated with slight enhancements of the simulation method, such as generating repetition or help requests. Surprisingly, a good 21.5% of the utterances not covered by the simulation were classified as experimental artifacts, where users did not act in accordance with the task description. Given the poor predictability of user behavior, such errors may happen in subjective tests even when they are neatly prepared. Still, in the simulation we prefer to avoid predicting such behavior, as it negatively impacts the measurements. In real life, such problems do not occur of course, as users do not select their task according to a scenario description. If these were removed from the data and the slight enhancements were made to the user model, a *Recall* of almost 70% would be achieved. For a task or system involving less complex user errors, we expect that the *Recall* would be even higher.

At the same time, over-generation of utterances, measured by the *Precision,* increases if more dialogs are simulated and the models are given more flexibility. However, as the simulated behavior is based on reasonable rules, most utterances are meaningful even if they do not occur in the simulation. Furthermore, if the simulation aims at finding interface problems, implausible user utterances do not harm as long as they do not cause costly system changes. To prevent the latter, the system designer utilizing the simulation could just search for all interaction problems and decide about system changes depending on their plausibility.

Despite the overall positive result, we are interested in improving the simulation to maximize its benefits. We therefore outlined in the previous section which traits of user behavior are still missing in the simulation. As a first issue, modeling all features of a dialog system still demands more flexibility in the system model editing process. As the simulation itself is concerned, the modeling of speech understanding errors should be more sophisticated. Some related work incorporates the generation of user utterances (e.g. Chung, 2004). Integrating this into our workbench would require the inclusion of system grammars in the system model. In an earlier project, we have developed a tool which estimates the confusion probabilities between semantic concepts of a grammar by comparing the respective entries phonetically (Möller et al., 2007c). By integrating this tool we would avoid to generate acoustic representations of utterances and, at the same time, incorporate a developed method to analyze the grammars in depth. Unfortunately, the prediction of confusions has not been tested extensively so far.

ACCEPTED MANUSCRIPT

Currently, we cannot derive knowledge about potential errors on the level of the Mental Model from the system and user characteristics alone. We will have to find ways to employ knowledge sources on the Internet or the designer's knowledge to generate such errors in a realistic and comprehensive way. Up to now, we have built a tool which calculates the semantic similarity between *output interactions* and user task knowledge (*keyword engine*), but it is not yet clear how this can be employed for the generation of all errors on the Mental Model level. As such a model describes how tasks are planned, we will probably also have to extend the representation of the user task knowledge with a more sophisticated structure.

Therefore, our next step will be to plug the *keyword engine* into the generation of task knowledge, and if necessary adapt the representation of the user task knowledge in the model. Furthermore, the user interaction model will be enriched in flexibility, so that generalization of concepts or generic actions such as help requests can be integrated in the simulation. When testing the resulting simulation, we can use a statistical model of understanding errors, e.g. an AVP confusion matrix derived from a corpus. If we consider the user model to be ready for evaluation tasks we will plug in a more complex mechanism to generate understanding error probabilities from scratch.

In parallel to this, we currently develop a usability profile generated from the simulated corpora. To do this we plan to utilize "dynamic user attributes", in which variable user factors (e.g. satisfaction, frustration) are tracked during the interaction. These attributes should be changed according to events in the dialog and influence the actions of the user model. We are currently conducting an experiment in which we measure user satisfaction throughout the interactions of users with the INSPIRE system. In addition to this, we need to develop methods for the detection and description of errors on the Mental Model level, which will allow the designer to draw conclusions for the improvement of the interface.

Finally, for testing future developments in the simulation, we will have to define better criteria to measure the quality of the simulation. As over-generation of utterances does not seem to be problematic, richness of user behavior seems to be a better quality criterion than similarity to a corpus. Recall of real user utterances can serve as an indicator here. By the time, as more usability problems can be found with the simulation, the number of true usability problems discovered will be a reasonable measure for the simulation quality.

Acknowledgement

The work presented in this paper was conducted in the frame of the MeMo project as a collaboration of Deutsche Telekom Laboratories, DAI-Labor, DFKI and Fraunhofer FIT. The authors gratefully acknowledge the implementations and inventions by Jens Haupert, Britta Hofmann, Marc Hümmer, Maximilian Kern, Dorothea Kugelmeier, Michael Kruppa, Wai-Lung Lee, Michael Quade, Mathias Runge, and Carsten Wirth, as well as the experimental support by Florian Gödde. We would also like to thank the reviewers for their valuable comments which greatly helped to improve this paper.

References

Ai, H., Litman, D., 2006. Comparing Real-Real, Simulated-Simulated, and Simulated-Real Spoken Dialogue Corpora, in: Proc. of the AAAI Workshop on Statistical and Empirical Approaches for Spoken Dialogue Systems, Boston, MA, USA.

ACCEPTED MANUSCRIPT

Ai, H., Weng, F., 2008. User Simulation as Testing for Spoken Dialog Systems, in: Proc. of the 9th SIGdial Workshop on Discourse and Dialogue, Columbus, OH, USA.

Ai, H., Litman, D., 2008. Assessing Dialog System User Simulation Evaluation Measures Using Human Judges, in: Proc. 46th Ann. Meeting of the Assoc. of Computational Linguistics (ACL), Columbus, OH, USA.

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

Araki, M. and Doshita, S., 1997. Automatic Evaluation Environment for Spoken Dialogue Systems, in: ECAI '96: Workshop on Dialogue Processing in Spoken Language Systems, Springer, London, UK, pp. 183-194.

Bohus, D., Rudnicky, A. I., 2005. Sorry, I Didn't Catch That! - An Investigation of Non-understanding Errors and Recovery Strategies, in: Proc. of the 6th SIGdial Workshop on Discourse and Dialogue, Lisbon, Portugal.

Card, S. K., Moran, T. P., Newell, A., 1983. The Psychology of Human-Computer Interaction. Erlbaum, Hillsdale, NJ, USA.

Chung, G., 2004. Developing a flexible spoken dialog system using simulation. Proc. of the 42nd Annual Meeting on Association for Computational Linguistics, Barcelona, Spain.

Eckert, W., Levin, E., Pieraccini, R., 1997. User Modeling for Spoken Dialogue System Evaluation, in: Proc. of the IEEE Workshop on Automatic Speech Recognition and Understanding, Santa Barbara, CA, USA.

Engelbrecht, K.-P., Kruppa, M., Möller, S., Quade, M., 2008. MeMo Workbench for Semi-Automated Usability Testing, in: Proc. of Interspeech 2008, Brisbane, Australia.

Fraser, N. M., Gilbert, G. N., 1991. Simulating Speech Systems. Computer Speech and Language 5, 81–99.

Griol, D., Hurtado, L. F., Segarra, E., Sanchis, E., 2008. Acquisition and Evaluation of a Dialog Corpus through WOZ and Dialog Simulation Techniques, in: Proc. of the 6th International Conference on Language Resources and Evaluation (LREC'08), Marrakech, Morocco.

Hermann, F., Niedermann, I., Peissner, M., Henke, K., Naumann, A., 2007. Users Interact Differently: Towards a Usability-Oriented Taxonomy, in: Proc. of HCI International 2007.

Ito, A., Shimada, K., Suzuki, M., Makino, S. 2006. A User Simulator Based on VoiceXML for Evaluation of Spoken Dialog Systems, in: Proc. of Interspeech 2006, Pittsburgh, PA, USA.

ITU-T Supplement 24 to P-Series Recommendations, 2005. Parameters Describing the Interaction with Spoken Dialogue Systems. International Telecommunication Union, Geneva, Switzerland.

ITU-T Recommendation P.851, 2003. Subjective Quality Evaluation of Telephone Services Based on Spoken Dialogue Systems. International Telecommunication Union, Geneva, Switzerland.

Ivory, M. Y., Hearst, M. A., 2000. The state of the art in automating usability evaluation of user interfaces (Technical Report UCB/CSD-00-1105). EECS Department, University of California, Berkeley, CA, USA.

Janarthanam, S., Lemon, O., 2008. User simulations for online adaptation and knowledge-alignment in Troubleshooting dialogue systems, in: Proc of SEMDIAL 2008 (LONDIAL), London, UK.

John, B. E., Salvucci, D. D., 2005. Multipurpose Prototypes for Assessing User Interfaces in Pervasive Computing Systems, IEEE Pervasive Computing 4(4), 27-34.

Kieras, D. E., 2003. Model-based Evaluation, in: Jacko, J., Sears, A. (Eds.), The Human-Computer Interaction Handbook, Erlbaum, Mahwah, NJ, USA, pp. 1191-1208.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

López-Cózar, R., de la Torre, A., Segura, J. C., Rubio, A. J., 2003. Assessment of dialogue systems by means of a new simulation technique. Speech Communication 40(3), 387-407.

Möller, S., Englert, R., Engelbrecht, K.-P., Hafner, V., Jameson, A., Oulasvirta, A., et al., 2006. MeMo: Towards Automatic Usability Evaluation of Spoken Dialogue Services by User Error Simulations, in: Proc. of Interspeech 2006, Pittsburgh, PA, USA.

Möller, S., Engelbrecht, K.-P., Oulasvirta, A., 2007a. Analysis of Communication Failures for Spoken Dialogue Systems, in: Proc. of Interspeech 2007, Antwerp, Belgium.

Möller, S., Smeele, P., Boland, H., Krebber, J., 2007b. Evaluating Spoken Dialogue Systems According to De-Facto Standards: A Case Study. Computer Speech and Language 21, 26-53.

Möller, S., Engelbrecht, K.-P., Pucher, M., Fröhlich, P., Huo, L., Heute, U., et al., 2007c. TIDE: A Testbed for Interactive Spoken Dialogue System Evaluation, in: Proc. 12th International Conference "Speech and Computer" (SPECOM'2007), Moscow, Russia.

Newell, A. (1990). Unified Theories of Cognition. Harvard University Press, Cambridge, MA, USA.

Nielsen, J., 1993. Usability Engineering. Academic Press, San Diego, CA, USA.

Norman, D. A., 1981. Categorization of Action Slips. Psychological Review 88, 1-15.

Norman, D. A., 1983. Some Observations on Mental Models, in: Gentner, D., Stevens, A. L. (Eds.), Mental Models, Erlbaum, Hillsdale, NJ, USA, pp. 7-14.

Pietquin, O., Renals, S., 2002. ASR System Modeling for Automatic Evaluation and Optimizatin of Dialogue Systems, in: Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Process. (ICASSP 2002), Orlando, FL, USA.

Pietquin, O., 2006. Consistent Goal-Directed User Model for Realisitc Man-Machine Task-Oriented Spoken Dialogue, in: Proc. of the IEEE International Conference on Multimedia and Expo (ICME), Toronto, ON, Canada.

Pietquin O., 2009. Machine Learning Methods for Spoken Dialog Simulation and Optimization, in: Mellouk, A., Chebira, A. (Eds.), Machine Learning, In-Teh, 167-184.

Rieser, V., Lemon, O., 2006. Cluster-based User Simulations for Learning Dialogue Strategies, in: Proc. of Interspeech 2006, Pittsburgh, PA, USA.

Schatzmann, J., Georgila, K., Young, S., 2005. Quantitative Evaluation of User Simulation Techniques for Spoken Dialogue Systems, in: Proc. of the 6th SIGdial Workshop on Discourse and Dialogue, Lisbon, Portugal.

Schatzmann, J., Thomson, B., Weilhammer, K., Ye, H., Young, S., 2007a. Agenda-based User Simulation for Bootstrapping a POMDP Dialogue System, in: Proc. of HLT/NAACL, Rochester, NY, USA.

Schatzmann, J., Thomson, B., Young, S., 2007b. Statistical User Simulation with a Hidden Agenda, in: Proc. of the 8th SIGDial Workshop on Discourse and Dialogue, Antwerp, Belgium.

Schatzmann, J., Thomson, B., Young, Steve, 2007c. Error Simulation for Training Statistical Dialogue Systems, in: Proc. of ASRU, Kyoto, Japan.

Scheffler, K., Young, S., 2001. Corpus-based Dialogue Simulation for Automatic Strategy Learning and Evaluation, in: Proc of the NAACL-2001 Workshop on Adaptation in Dialogue Systems, Pittsburgh, PA, USA.

Seneff, S., 2002. Response Planning and Generation in the MERCURY Flight Reservation System, Computer Speech and Language 16, 283–312.

Walker, M. A., Litman, D. J., Kamm, C. A., Abella, A., 1997. PARADISE: A Framework for Evaluating Spoken Dialogue Agents, in: Proc. of the ACL/EACL 35th Ann. Meeting of the Assoc. for Computational Linguistics, Madrid, Spain.

ACCEPTED MANUSCRIPT

Walker, M., Kamm, C., Litmann, A., 2000. Towards developing general models of usability with PARADISE, Natural Language Engineering 6(3-4), 363-377.

ACCEPTED MANUSCRIPT