Chapter 9
Assistive and Adaptive Dialog Management

Florian Nielsen and Wolfgang Minker

Abstract One of the most important challenges in the field of human-computer interaction is maintaining and enhancing the willingness of the user to interact with the technical system. This willingness to cooperate provides a solid basis which is required for a collaborative human-computer dialog. For the dialog management this means that a *Companion*-System adapts the course and content of human-computer dialogs to the user and assists during the interaction through individualized help and explanation. In this chapter we elucidate our dialog management approach, which provides user- and situation-adaptive dialogs, and our explanation management approach, which enables the system to provide assistance and clarification for the user during run-time.

9.1 Introduction

The usual task of the dialog management (DM) is to control the structure, content, and flow of the dialog between human and computer. It communicates with the application and gathers from as well as provides to the user all necessary information, which is needed to accomplish a specific task in cooperation with a technical system. As an individual cognitive technical system should be able to adapt to a user's capabilities, preferences, and current goals and take into account the situation and emotional state, the dialog management is one of the most influential components to foster the realization of these system properties. These features and properties qualify a system as a so-called *Companion*-System (cf. Chap. 1). *Companion*-Systems are by definition "continually available, cooperative, reliable and trustworthy assistants which adapt to a user's capabilities, preferences, requirements, and current needs" [23].

From this *Companion*-System definition the authors derived those user characteristics (see Fig. 9.1) which are most important for the use in a dialog management component, fostering the realization of the *Companion*-properties. Here, *emotional state* is exchanged with *affective state*, because state-of-the-art research focuses on

F. Nielsen (✉) • W. Minker
Institute of Communications Engineering, Ulm University, Ulm, Germany
e-mail: florian.nothdurft@alumni.uni-ulm.de; wolfgang.minker@uni-ulm.de

S. Biundo, A. Wendemuth (eds.), *Companion Technology*, Cognitive Technologies,
DOI 10.1007/978-3-319-43665-4_9

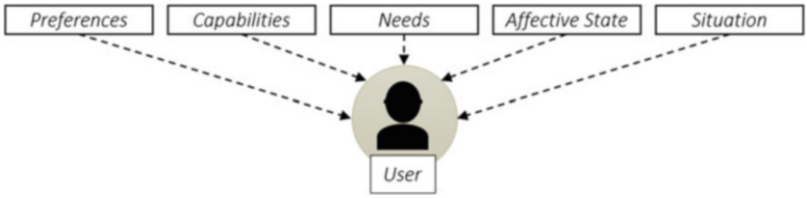

Fig. 9.1 The main user characteristics: A user's *capabilities* represent, for example, the user's knowledge or handicaps. *User needs* are derived from the user's current goals while the *situation* describes the user's present environment. *Preferences* are user made or learned adaptation criteria, and the *affective state* represents emotions as well as dispositions towards the technical system

recognizing and reacting not only to a user's primary emotions, but also to a user's disposition towards a technical system. Those dispositions are secondary emotions and can describe a user's stance towards a system better, by using terms like confusion, interest or frustration, rather than primary emotions like anger, sadness or joy [19]. Systems adapting their functionalities to users' general characteristics (e.g. capabilities or preferences), users' affective states or the present situation may help to lead to a cooperative and effective dialog between man and machine. Adapting the human-computer interaction (HCI) to the individual user does yield possible advantages. For example, the DM may adapt the flow of a dialog between human and machine to the user (e.g. to the user's knowledge) and by that help to prevent overextension or mental overload. However, individual systems that act upon implicit user information and adapt their system behavior accordingly may confuse or disturb the user's perception of the system. Especially proactive behavior (e.g. reacting autonomously to the user's affective state) might not be understandable to the user, or in other words incongruent to the user's mental model of the system. One of the main aspects in *Companion*-Systems is their ability to evolve the capabilities of solving problems collaboratively with the user. However, solving complex problems together with a *Companion*-System is only feasible if the human user interprets and understands the machine correctly.

These situations are critical and may have a negative impact on the human-computer trust (HCT) relationship [12]. The main problem is that if the user does not trust the system and its actions, advice or instructions, the way of interaction may change up to complete abortion of future interaction [18]. Glass et al. [6] observed that the capabilities to provide explanations in adaptive agents can reduce most of the trust concerns identified by the user. In addition, several other studies showed that explanations are a way to prevent the decrease of trust (e.g. [3, 4]). Therefore, a *Companion*-System should also be able to integrate explanations about the system's proactive behavior instantaneously in the dialog. This means that, depending on the current user model and situation, the *Companion*-System should be able to induce the appropriate kind of explanation to increase the chances of a successful and trustworthy HCI. Therefore, in the following we will elucidate how the DM may adapt the flow and structure of the HCI using explanations to individual user

characteristics, but also why and how a dialog system may cope with problems resulting from misunderstood system behavior.

The chapter is structured as follows. In Sect. 9.2 we provide introductory information on dialog management approaches. Section 9.3 describes the general concept and design ideas, including requirements for the user's knowledge and mental model. In Sect. 9.4 the implementation of the adaptive and assistive capabilities of the DM in a rule-based and probabilistic fashion is described, followed by the conclusion in Sect. 9.5.

9.2 Background

In the past decades several approaches to the DM task have been developed that can be classified in four basic categories [9]. First of all, basic *finite-state-machine* approaches (e.g. [25]), where a set of states is defined with a set of moves for each state which transitions to a new state in the automaton. Second, *frame-based* approaches (e.g. [8]), where the DM is monitoring the current so-called frame, which is specified by a set of needed information (slots), the context for the utterance interpretation and the context for the dialog progress. This is more advanced, since it allows for mixed-initiative interaction and allows multiple paths to acquire the information. Thirdly, stochastic-based approaches (e.g. [24]), which apply reinforcement learning techniques to the DM by determining the best policy or choice of actions from all available actions a system can take in a dialog, which will optimize the system's performance as measured by a utility function, such as the user's evaluation of the system [11]. The last main group of approaches is *agent-based approaches* (e.g. [14]) where the dialog is controlled by several intelligent agents capable of reasoning about themselves (e.g. in the BDI (beliefs, desires, intentions) approach [20]) using Artificial Intelligence techniques. The agent-based approach subsumes plan-based dialog models, where preconditions, actions and effects are used to control the ongoing dialog. Here, the structure of the dialog (i.e. the flow) is determined at run-time by the different agents using the current world state and the goals left to achieve. Thus, the agent-based approach is the closest to the one used here.

9.3 Concept and Design

For an individual *Companion*-System, the DM integrates into a complex architecture and splits its responsibilities with the *planning framework* (PF). Due to the fact that DM and PF are closely linked together, a dialog management approach related to the planning approach used in the PF was chosen. This means that, as already described in Chap. 7, the course of plan steps representing the solution for the given planning problem is generated and executed by the planning-framework (i.e. the

plan generation and *plan execution* components). In case that the proposed plan steps require user interaction, plan steps are passed from the plan execution to the DM. The main purpose of this component is to decompose and refine the plan step (if necessary) into a user-adaptive dialog. If the dialog for the passed-on plan step requires adaptation to the individual user, the structure and the content of the dialog can be adapted. Therefore, our combination of DM and PF is closely related to the split into task level and dialog level used in agent-based approaches.

During this work two different approaches were used. First, a rule-based approaches using preconditions, thresholds and predefined rules to adapt the structure and flow of the dialog. Second, a hybrid approach using rule-based approaches for the task-oriented part of the interaction, and a probabilistic approach for dealing with adaptivity to user and situation characteristics based on uncertain or noisy information sources (e.g. user knowledge, user understanding or affective user states) (cf. Sect. 9.4.3). The framework for the integration of user- and situation-adaptive dialog management capabilities was a cognitive knowledge-based technical system, which extends the architectures of classic unimodal (e.g. spoken dialog systems) or multimodal dialog systems. This system [7] can be called a prototypical *Companion*-System [2]. To be able to provide *Companion*-functionalities in a completely individualized way, several interacting components (see Fig. 9.2) are necessary. The main differences with a classic architecture are the underlying knowledge base, and the cognitive capabilities by the use of a sensor system, and that the dialog management and the planning framework share the tasks of planning, controlling and structuring the HCI.

9.3.1 Required User Model

As already mentioned, the following adaptation criteria are needed for the integration of the desired explanation capabilities and stored in the user model: the general user knowledge, the fine-grained user knowledge and the user's mental model.

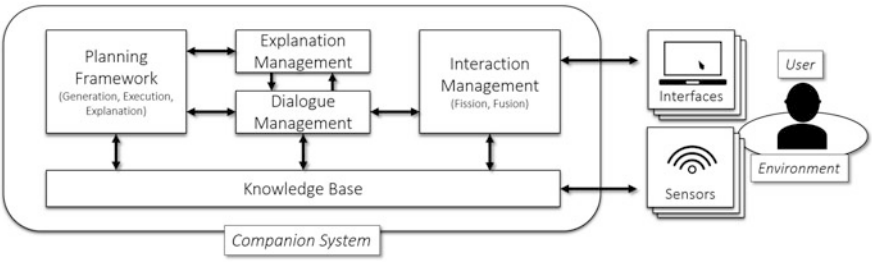

Fig. 9.2 This figure shows the prototypical *Companion*-System. This system serves as framework for the modules developed or extended: *explanation* and *dialog management*. Embedded between the *planning framework*, *interaction management*, and the underlying *knowledge base*, they control the user-adaptive dialog between human and machine

9.3.1.1 General and Fine-Grained User Knowledge

One of the first design decisions was to model user knowledge in a fine-grained way, using the same level of detail as used in the domain model of planning and dialog. For example, in the example domain used here, the task represented as relation *connect(TV, Receiver, HDMI)* models the knowledge of the action *connect* as well as the concepts *TV*, *Receiver*, and *HDMI*. Additionally, we draw a distinction between declarative knowledge and procedural knowledge. The former can be used to describe the being of things (i.e. appearance, purpose). Possessing declarative knowledge about something does not necessarily mean begin able to use this knowledge for a task or action. In comparison to that, procedural knowledge can be applied to a task. Procedural knowledge provides the knowledge on how to execute a task or on how to solve a problem (Fig. 9.3).

Those knowledge constituents are modelled as probability distributions over a five-step knowledge scale ranging from *novice* to *expert* for every constituent (see Fig. 9.7). This means that knowledge is modelled in small pieces instead of in only one general level. This helps the model to allow for a more realistic and exact individualization and adaptation to the user. Though we concentrate on a fine-grained knowledge representation, we also build a mean overall knowledge value for coarse adaptations of the dialog flow (cf. Fig. 9.4). This means that a mean of all domain knowledge constituents is calculated, representing the overall knowledge of the user. Hence, the categorization into groups or levels of expertise is based not only on a general assumption, but also on a more thoroughly combination of all domain elements. The development of the user's knowledge is based on observations made during the interaction and on past interaction episodes (cf. Fig. 9.5). Therefore, the knowledge levels are system-made assumptions about the user, requiring a probabilistic representation. Occurring events during the HCI influence the probability distribution of relevant contained knowledge objects.

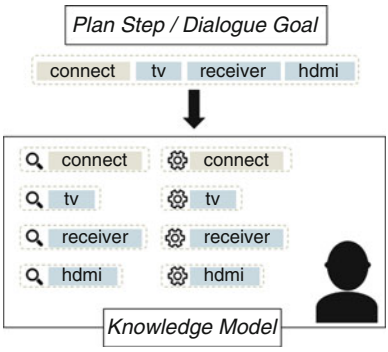

Fig. 9.3 The relation coming from either the dialog goal or the plan step directly can be analyzed separately by the explanation management, because they are stored in a fine-grained way in the *User's Knowledge Model*. Here, *the magnifying glass* represents declarative knowledge elements and the cogwheel procedural knowledge elements

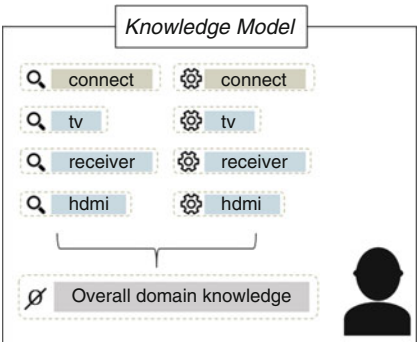

Fig. 9.4 The user's overall knowledge of a domain is used for the coarse adaptation of the dialog flow. It consists of a combination of all domain elements and therefore represents the mean domain knowledge

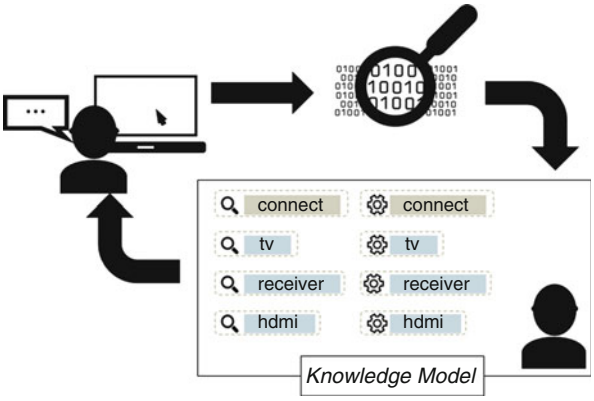

Fig. 9.5 The fine-grained user's knowledge evolves over time. This means that in a first step user interactions or other observations are processed. This may then influence the knowledge model and its constituents. The changed knowledge model may subsequently lead to adaptation and influence the HCI

These events may be, for example, given explanations, failed actions or, plainly elapsed time. Contained means, for example, not only tasks the user executed, but also entities that were used for this task and whether the task completion was successful.

9.3.1.2 User's Mental Model

Our main interest in mental models is the consequence that they are built by the user and only imitate the perceived system behavior. This means that the perceived mental model may not correspond to the real-world model and may therefore be

incorrect. This process of adapting existent mental models or "fine-tuning" them is very important. Whenever some feedback is observed which is incongruent to the present mental model, the model has to be adapted. Additionally, these situations are those in which the user does not understand the system behavior, because of incongruent models, and the risk increases that the user's perception of the system's trust components (e.g. understandability, reliability) might be impaired. To foster effective mental model correction it showed that transparency explanations worked best [17]. However, as mental models cannot be directly observed we are left to estimate whether incongruence in mental and actual system models is present. However, is we look for situations in which not only the mental model incorrect, but also the receptor is not sure whether the mental model applies.

Therefore, due to the lack of direct observability, the mental model is hard to build. However, what can be used, in order to estimate a mental model, is affect recognition combined with context information like the interaction history and current state of the dialog. For this we are keeping track of the HCI in a so-called dialog history. This history records the decisions and actions of the user and the technical system. The history is used to note when, for example, something was explained to the user, when the user executed a task or requested help from the system. Here, information of specific system actions or decisions that have been performed in the past is stored in the history as well. For example, the estimated mental model includes whether explanations were given for system decisions or tasks and whether the transparency on decisions was upheld by providing information on the reasoning process that led to the decision.

9.3.2 Consequences for the Design

While the description of the underlying architecture already gave some hints on where the main adaptation criteria, derived from the design requirement, are applied, this section will elucidate the theoretical intent behind these design decisions. Where, why and how those criteria (*general user knowledge*, *fine-grained user knowledge*, and *user's mental model*) may be used in the present system will be described in the following.

9.3.2.1 General User Knowledge

This model represents the user's general knowledge about a domain or sub-domain. Therefore, it is an arithmetic mean of user's knowledge about a specific topic, for example, the user's knowledge about cooking, technical systems or cars. Analogous to human-human interaction (HHI), a person's knowledge is based on the subjective opinions and objective criteria of another party. Humans tend to assign competencies to other persons, like "he is a good cook", "she knows how to repair cars" or "he knows everything about computers". However, we do not only assign abstract

competencies like that, but also very specific competencies, like “she makes a great lasagna”, “he knows everything about Linux”, or “she can change the tires of her VW Golf”. Now, if a human seeks help or clarification, the most important classification criteria are the abstract ones. However, if the human already possesses a certain amount of knowledge in that domain, the more specific competencies tend to become more important. This process of selecting the most competent help for the user’s current situation gets more and more complex the more specific his or her problem and existing knowledge in the domain is. Transferred to technical systems, this means that a user’s general knowledge is suitable for a coarse adaptation of the dialog flow and structuring of HCI. However, if more specific competencies in a domain are required, a more fine-grained estimation of the user’s knowledge is needed for an effective and sound HCI.

Therefore, in our present architecture, the dialog management and its *guard-based* dialog model are suitable for adapting to a user’s general knowledge. The dialog model is able to adapt to *guards*, which control the flow of the dialog. The drawback of increasing complexity for higher numbers of *guards* is not relevant if the user’s general knowledge is regarded as one of only a small number of sub-concepts.

9.3.2.2 Fine-Grained User Knowledge

As already explained in the former paragraph, a detailed estimation of the user’s knowledge is needed to assist the user in a more specific and more effective manner. Since the *guard-based* dialog model is not suitable for a fine-grained adaptation to the user’s knowledge model, an automatic solution not based on predefined dialog paths seems preferable. Hence, an automatic, proactive augmentation of the already roughly adapted dialog has to be done. Analogously to HHI, where, for example, “chef apprentices” get, on top of their already adapted instructions, more detailed information based on their specific competencies (e.g. “cut the chicken filet in this direction”), in HCI this has to be done as well. While observing the user’s behavior and interaction history, one has to decide whether additional clarification or instruction is needed in order to achieve the desired goal. In the present architecture the explanation management has the purpose of dealing with these kinds of adaptations at run-time. It observes the ongoing dialog and decides based upon the history of interaction and the upcoming dialogs whether additional explanations are needed to match the user’s fine-grained knowledge model. Therefore, dialogs have to be modelled in a way that they can be assessed in terms of required knowledge. For example, the task “prepare the béchamel sauce” requires knowledge about béchamel sauce in general, as well as knowledge about the ingredients and of course about the procedure for preparing it. In order to enable the system to assess the requirements, the knowledge constituents have to be modelled for the dialogs.

9.3.2.3 User's Mental Model

The user's mental model cannot be observed directly. As explained before, only symptoms rather than the disease combined with present context information can be used to estimate whether the user's mental model and the actual system behavior do align. Hence, affective states that indicate, for example, *frustration* or *confusion* have to be used in combination with information on the dialog history and the surroundings to estimate mental model incongruences. However, the recognition of affective user states is error-prone and burdened with uncertainty, especially considering non-acted data. Thus, to treat the uncertainty-based affective states in a proper way, a probabilistic model is needed.

The main idea of the probabilistic model is the integration of proper uncertainty treatment for information based on uncertain observations. Apart from factors like uncertain environments and decision-making, incorporating mental model adaptivity directly into the *guard-based* DM is not good idea. Modelling the incongruence of mental models as a guard variable would be possible, but predefining the adaptations is not.

In previous research we showed that especially transparency explanations, which aim for explaining the system processes, lead to a higher perception of understandability of the system's behavior [15]. However, this goal of explanation cannot be predefined for every situation by the designer. That's why the adaptivity to a user's mental model has to be done during run-time, and for our systems this means in the *explanation management*. In Sect. 9.4.3 we explain how the estimated mental model can trigger explanations. Though we can decide on whether explanations are needed, the explanation itself has to be generated by the modules responsible for the decision-making in terms of the dialog and plan, which will be elucidated in the next section.

9.4 Implementation

As already mentioned, different approaches and facets were developed, which will be explained in the following. In Sect. 9.4.1 the dialog flow adaptation of a task-oriented dialog is exemplarily explained on the user's knowledge level. Section 9.4.2 describes how a fine-grained user knowledge model can be used to provide user-adaptive assistivity by augmenting a task-oriented dialog in a rule-based way without interfering in its original dialog approach. How a probabilistic approach can be used and why such an approach is necessary to integrate explanation capabilities in a realistic and effective fashion is then explained in Sect. 9.4.3.

9.4.1 Rule-Based Adaptivity

As described in Sect. 9.3, the generated plan of the planning framework serves as skeleton of the user-adaptive dialog. The provided plan steps are decomposed one by one into a hierarchical dialog structure which consists of so-called *dialog goals* [16]. Each dialog goal represents a single interaction step between human and system (e.g. one screen on a device filled with specific information). The term dialog goal arises from the fact that every step in the interaction pursues a goal. This goal is, in this case, to achieve one or several of the desired plan step effects. Therefore, the term dialog goal is to be distinguished from the term goal used in the PF components. This means that a plan step may be decomposed into several dialog goals and that for every desired plan step effect a set of similar dialog goals may exist. These similar dialog goals usually have so-called *guards* which formulate conditions that need to be fulfilled in order for the dialog goal to be entered at run-time. These guards may, for example, take into account general user knowledge levels and therefore help to adapt the dialog to user characteristics stored in the knowledge base as marginal.

Goals can be arranged in a vertical structure and also in a horizontal structure. In a vertical structure each goal may yield several sub-goals. In a horizontal structure each goal may have a fixed successive goal that is next in the dialog. This implies that the dialog may be roughly structured like a finite state machine, but there is enough room left to dynamically arrange the sub-goals to satisfy the user's needs. Such an arrangement is handled by the way the guards for the goals are defined. Guards are preconditions protecting the related dialog goal of inappropriate execution, leading most likely to dialog failure (e.g. due to inadequate user knowledge). Those roughly made dialog structure adaptations are later augmented by assistive behavior manifested by additional explanations during run-time, adaptive to the user's fine-grained knowledge, which will be explained in Sect. 9.4.2.

During the interaction, the dialog management traverses through its dialog structure (see Fig. 9.6) to select the path of dialog goals most suitable for the

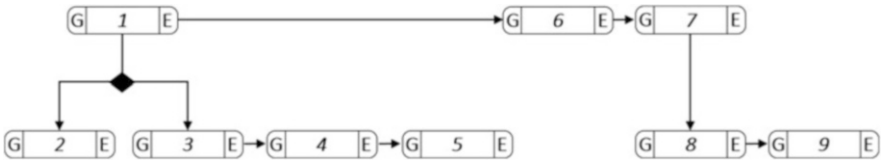

Fig. 9.6 *Dialog Goals* may have sub-goals (cf. 2 and 3 for goal 1) or link to the next goal in the sequence (cf. 6 to 7). More abstract goals (e.g. 1 and 7) are not directly executable, but have to be refined. The dialog content and flow may change according to predefined variables used as *guards*, depicted as *G*. For example, if one *guard* of goal number 2 requires expert knowledge, the resulting sequence of dialog steps, in case of fulfilment, would include a simplified instruction (i.e. 2,6,8,9). Contrary to that, a novice user would receive a more detailed as well as longer sequence of dialog steps with additional extent of assistance (i.e. 3,4,5,6,8,9). Dialog Goals have effects as well, describing the effects of execution (depicted as *E*)

current user. The selection of the next dialog goal is therefore made in a user-adaptive manner and leads to an individual dialog appropriate for the current user. In order to conduct the selection of the next appropriate dialog goal, a constraint solving algorithm is used. Constraint programming has proved to be especially useful in problems on finite domains where many conditions limit the possible variable value configurations [5]. It is a technique to find solutions to problems by backtracking and efficient reasoning. As a dialog in our case is limited to a certain number of possibilities how the system can traverse through the dialog structure, it is reasonable to use a finite domain for the variables that constrain the execution of the dialog goals. These conditions make the dialog model suited for applying a finite domain constraint-solving mechanism.

The decision about which dialog goals can be executed is based on the values that variables specified in the guard conditions are allowed to have. Applied to constraint programming, we consider the conditions in the guards as constraints, and, based on the current values of the variables, the constraint solver tells which guard conditions can be fulfilled. Based on this variable configuration we can select a number of dialog goals which can be executed at a certain time in the dialog. The described adaptation of the dialog flow is only suitable for roughly customizing the dialog using the mean user knowledge due to complexity issues. Using the fine-grained knowledge model would lead rapidly to overly complex dialog structures. This means that knowledge is here regarded as one mean value for the complete domain or coarse subdomains. For example, in the sample domain of connecting a new home theater system, the user's mean technical knowledge is considered the main adaptation variable. The dialog content and flow will change if two different users interact with the system. For example, if the first user is an *expert* (i.e. has a mean technical knowledge ≥ 4.5), the sequence of dialog steps should include simplified instructions, sufficient for an expert. Contrary to that, a novice (i.e. mean technical knowledge <1.5) should receive a more detailed as well as a longer sequence of dialog steps with additional extent of assistance. The user variable *mean technical knowledge* is designed to be the arithmetic mean of the combination of the user's default knowledge model and the evolution of its fine-grained constituents during past episodes of HCI (see Fig. 9.5).

The mean technical knowledge is used as a *guard* to formulate conditions necessary for the execution of the dialog goal. Due to the rule-based DM approach using links to sub-goals or to proceeding goals, combined with the requirement to define those links by hand, the overhead vastly explodes in the case of a large number of guard variables. Therefore, the *user knowledge adaptation* is only used for few selected variables, which influence the dialog flow to a great extent.

9.4.2 Rule-Based Assistivity

Adaptation of structure and flow of the dialog to levels of user characteristics (e.g. mean technical knowledge) yields only coarse individualization results. Due to the

effort necessary to design a specific dialog course or flow for every level of user characteristic and the resulting complexity, it is uneconomical to use the former approach for more fine-grained adaptation and individualization. Therefore, the course of interaction steps is adapted and extended for a more individualized dialog to the individual user's knowledge during run-time.

The assistive part of the DM, the *dialog augmentation*, deals with the automatic extension of the ongoing dialog with additional dialog steps, helping the user to accomplish tasks ahead. Contrary to the rule-based adaptation, which chose the most appropriate already predefined dialog path for the user, this approach integrates independent dialog steps into the running dialog. This means the designer does not have to cope with the knowledge-based fine-grained individualization, but the explanation management makes sure that the upcoming dialog steps are conform to the user model.

As already explained in Sect. 9.3, the course of steps the user has to fulfil in order to solve the task he wants to accomplish is first planned by the planning framework and later decomposed further by the DM into so-called dialog steps. Each of these steps is represented as a relation with the name of the task and its appendant concepts as its arguments. As our main goal is to prevent task failure we have to ensure that the upcoming or current tasks and appendant concepts do not exceed the user's knowledge. Therefore, prior to sending the dialog steps for presentation to the *multimodal fission* (see Chap. 10), they are first sent to the explanation management. Here, the content of the predefined dialog is analyzed and compared to the user's knowledge model, stored in the knowledge base as marginal. If the user's knowledge is probably not sufficient, the course of the dialog is updated by including additional dialog steps to better fit the user's knowledge model. Those additional dialog steps are meant to explain missing knowledge to the user.

As previously reported, in our knowledge model we distinguish between declarative knowledge and procedural knowledge. For example, in the domain used in the demonstrator, the task represented as relation *connect(TV, Receiver, HDMI)* models the knowledge of the action *connect* as well as the concepts *TV*, *Receiver* and *HDMI*. This means that compared to other systems the user's knowledge is modeled in small pieces instead of in only one general level. Of course, the resulting model is very complex, but also more realistic and conforming more to the idea of human knowledge modelling.

The knowledge constituents are modelled as probability distributions over a five-step knowledge scale ranging from *novice* to *expert* (see Fig. 9.7). As the

```
<proceduralGoalKnowledge goalName="connect" goalID="3.2">
  <knowledgeValue value="novice" probability="0.5"/>
  <knowledgeValue value="advanced novice" probability="0.3"/>
  <knowledgeValue value="intermediate" probability="0.1"/>
  <knowledgeValue value="advanced intermediate" probability="0.1"/>
  <knowledgeValue value="expert" probability="0"/>
</proceduralGoalKnowledge>
```

```
<declarativeConceptKnowledge conceptName="hdmi">
  <knowledgeValue value="novice" probability="0.5"/>
  <knowledgeValue value="advanced novice" probability="0.3"/>
  <knowledgeValue value="intermediate" probability="0.1"/>
  <knowledgeValue value="advanced intermediate" probability="0.1"/>
  <knowledgeValue value="expert" probability="0"/>
</declarativeConceptKnowledge>
```

Fig. 9.7 In this excerpt of the user's knowledge model the procedural knowledge of the action *connect* and the declarative knowledge of the concept *HDMI* as well as their respective probability distributions over the knowledge levels are listed

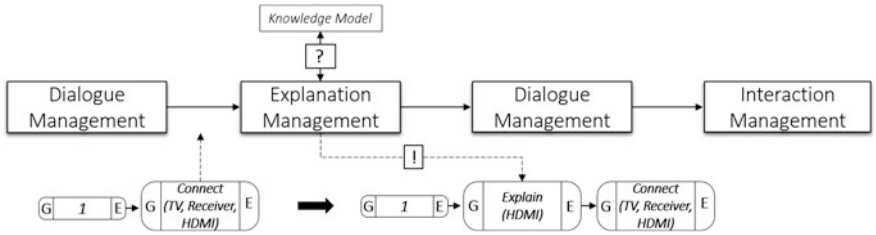

Fig. 9.8 Here we can see that the original predefined dialog on the *left* is augmented during runtime by an additional explanation dialog. The current dialog *1* is completed and before execution the next dialog is analysed by the explanation management. By checking the knowledge model it concludes that the user's knowledge about HDMI is not sufficient and decides to augment the dialog with an additional explanation dialog

user's knowledge is based on observations made during the interaction and on past interaction episodes and the user's knowledge is based on system-made assumptions about the user, a probabilistic representation is required. During the interaction we check if the user's knowledge for the current dialog step constituents is most probably high enough. The concepts and the action are analyzed by the explanation management, and if needed the dialog flow is augmented with additional dialog steps (see Fig. 9.8). If the user's knowledge is most probably too low, the explanation manager generates an additional explanation dialog, which tries to impart the missing knowledge for the user to execute the dialog step successfully.

The explanation management then selects which type of explanation is the appropriate one for the current lack of knowledge. As the dialog step consists of several parts, so does the explanation. Each task and concept are analyzed to generate a summarized explanation. The explanation management sends the content of the explanation to the knowledge base to be stored in the *information model*. This explanation may consist of pictures, text or text meant to be spoken. Afterwards, a dialog step is generated, which references the content of the explanation stored in the information model and sent to the DM to be included in the course of interaction. The DM will then proceed to send the additional dialog to the multimodal fission component, which then decides by modality arbitration on which device and how the content should be presented.

As mentioned earlier, the explanation management not only verifies the dialog step's contained knowledge, but also coordinates and processes explicitly stated explanation requests from the user and generates, if necessary, additional dialog steps to be included in the ongoing dialog.

One of the main drawbacks of the rule-based approach is the improper treatment of uncertainty. Though we are using probability distributions to model the user's knowledge levels from novice to expert, the processing of events does not meet all wished for requirements for uncertainty treatment. The development of the probability distribution is a rather imprecise rule-based approach, only incorporating explicit human-computer interaction. However, implicit information coming from the user (e.g. affective states) is an important indicator for the development of

user knowledge (i.e. did the user understand or not). Obviously, recognizing those situations cannot be done solely by using information coming from interaction and its history. Multimodal input such as speech recognition accuracy, facial expressions or any other sensor information can help to improve the accuracy of recognizing critical moments in HCI. Especially affective user states like *confusion* and *frustration* can help to reason about understanding or accomplishment of tasks, instructions, or explanations. However, mapping implicit user information coming from sensor input to semantic information is usually done by classifiers, and those classifiers convey a certain amount of probabilistic inaccuracy, which has to be handled. Therefore, a decision model has to be able to handle probabilistic information in a suitable manner. How this kind of implicit user information is integrated to foster consistent and proper uncertainty treatment will be explained in the next section.

9.4.3 Probabilistic Assistivity

The former sections concentrated on the adaptation of the dialog to high-level user variables such as the user's mean knowledge or verbal intelligence, and on the augmentation of the task-oriented dialog with conceptual or procedural explanation dialogs. However, one of the most important factors in HCI is the user's affective state or disposition towards the system. This means that situations in which the user does not understand the system's actions or instructions are very critical for a trustworthy and sound HCI. Incomprehensible, not understandable system behavior may lead to the loss of trust and in the worst case to the abortion of interaction.

As elucidated in [17], the different goals of explanation are suitable for different situations in HCI. Therefore, we need to estimate the user's state and reason about the interaction state in order to decide upon the most appropriate and most effective system reaction strategy for the current situation and user. For this, not only explicit, but also implicit interaction information has to be used. Apart from explicit, data (e.g. touch, speech, or click), especially implicit data (e.g. affective state, location, or user profile) can help in estimating the user's state, for example the user's mental model, in a better way.

The main idea of the *probabilistic assistivity* is the integration of proper uncertainty treatment for information based on uncertain observations. However, since decision-making under uncertainty requires complex and elaborate models, and tends to get unsolvable fast, only the augmentation process is controlled by a probabilistic model. For the problem representation of when and how to react, a so-called partially observable Markov decision process (POMDP) was chosen and formalized in the Relational Dynamic Influence Diagram Language (RDDL) [21]. RDDL is a uniform language which allows an efficient description of POMDPs by representing its constituents (actions, observations, belief state) with variables. On the one hand we are using a classic dialog approach as described in Sect. 9.4.1 for the task-oriented part of the dialog. On the other hand a planner [13] is integrated

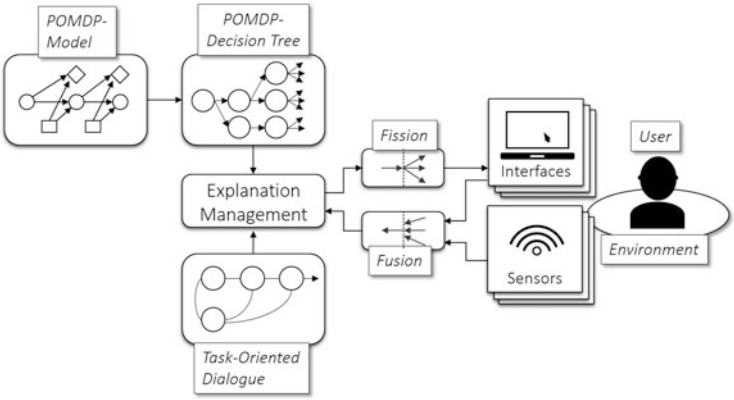

Fig. 9.9 The architecture consists of two dialog models, a fission and fusion engine, sensors as well as the multimodal interface representation to interact with the user. The dialog models can be separated in a task-oriented dialog model and into a POMDP-based decision tree for explanation augmentation. This decision tree is generated from a POMDP-model by a planner

to generate from a POMDP a decision tree. This POMDP is used only for the augmentation of the task-oriented part of the dialog (Fig. 9.9).

9.4.3.1 Dialog Augmentation Process

The task-oriented dialog is modelled in a classic dialog approach. Each dialog action has several interaction possibilities, each leading to another specified dialog action. Each of those dialog actions is represented as a POMDP action a as part of C (*communicative function(c)*). As already mentioned, only the communicative function is modelled to reduce the complexity in the POMDP.

The HCI is started using the classic dialog approach and uses the POMDP to check whether the user's trust or components of the user's trust are endangered. At run-time the next action in the task-oriented dialog is compared to the one determined by the POMDP (see Fig. 9.10). This means that if the next action in the task-oriented dialog is not the same as the one planned by the POMDP, the dialog flow is interrupted, and the ongoing dialog is augmented by the proposed action. For example, if the user is currently presented a communicative function of type *inform* and the decision tree recommends providing a transparency explanation because the understanding and reliability are probably false, the originally next step in the dialog is postponed and first the explanation is presented. The other way around, if the next action in the task-oriented dialog is subsumed by the one scheduled by the POMDP, the system does not need to intervene. For example, if the next dialog step is to instruct the user about how to connect *amplifier* and *receiver* and the POMDP would recommend an action of type communicative function *instruct*, no dialog augmentation is needed.

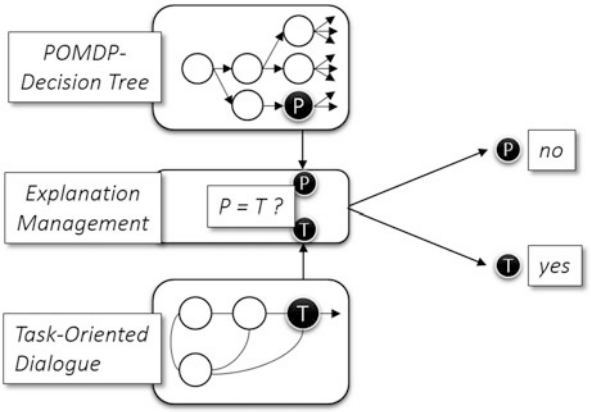

Fig. 9.10 This figure shows the comparison of task-oriented dialog to the POMDP-generated Decision Tree. If the next action T in the task-oriented dialog does not correspond to the one endorsed by the POMDP Decision Tree (P), the dialog will be augmented by the POMDP action. However, if they align, no intervention is necessary, and the planned action T is executed

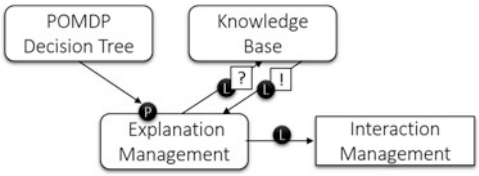

Fig. 9.11 The proposed decision P of the POMDP model is in this case a *learning* explanation request. Hence, the explanation management looks up the matching learning explanation (the *question mark L*) for the current topic in the knowledge base and returns the explanation content. This is then converted to a dialog step (the *exclamation mark L*), which is then passed to the interaction management for presentation

While the selection of the explanation is one thing, the integration of the real explanation is another. The POMDP only tells us whether and what kind of explanation should be integrated.

9.4.3.2 Explanation Selection

Basically there are four goals of explanation we are interested in covering. The first two are imparting declarative and procedural knowledge by using *conceptual* and *learning* explanations respectively. These typical explanations realised in contemporary systems by tutorials (learning) or plain help texts (conceptual) are usually explanations defined by experts. When the POMDP recommends integrating one of these explanations, the content of the explanations is gathered from the database, converted to a dialog step and integrated into the dialog (see Fig. 9.11). In our

system these explanations are also stored in a predefined way. There are information fragments available, which can be used for explanations, for all contained concepts of the used planning and dialog domains. These information fragments can then be combined to explanation dialog steps, which can then be integrated in the ongoing dialog flow. For the actions themselves, information fragments exist as well, representing learning or parts of learning explanations. These fragments are requested from the explanation management and subsequently processed into dialog steps and passed to the interaction management.

However, incomprehensible system behavior cannot be explained using these goals of explanation. Incomprehensible situations are prone to influencing the relationship between human and technical system negatively [12]. Especially if the user does not trust the system and its actions, advice or instructions anymore, the way of interaction may change up to complete abortion of future interaction [18]. Therefore, not understandable system behavior is critical for a sound and trustworthy HCI. However, the risk of negative consequences can be reduced by providing specific types of explanations [6, 10]. The most effective explanation goals to handle incomprehensible system behavior are *transparency* and *justification* explanations [15, 17]. Hence, these are the other two goals of explanations we want to handle.

Justifications are the most obvious goal an explanation can pursue in HHI discussions. The main idea of this goal is to provide support for and increase confidence in given advice or actions. For example, "you have to eat an apple a day, because that keeps the doctor away" would be some sort of justification. The goal of transparency is to increase the user's understanding of how the interlocutor's behavior (e.g. providing advice or instructions) was attained, in terms of the interlocutor's inner processes; for example, "I just heard that your car's exhaust is rattling, which is a bad sign, therefore you have to bring it to the car repair shop". If a technical system can provide these kinds of explanations, it can help the user to change his perception of the system as a black-box to a system the user can comprehend. Thereby, the user can build a correct mental model of the system and its underlying reasoning processes. In general, one can remark that justifications tend to be abstract explanations, which do not necessarily directly relate to system behavior or processes. Thus, justifications can be predefined, at least to a certain degree, by experts. If the user requests *why* a dialog step *ds* (e.g. "connect(TV, Receiver, HDMI)") has to be done, an explanation prepared beforehand (e.g. "You have to connect TV and receiver with an HDMI cable to transmit audio and video signals") can be presented to the user. Alternatively, one could provide an explanation based on the hierarchy in the domain, such as "this has to be done to connect your home theater", a combination of both, or even something not related to the structure or dialog concepts, such as "to be able to watch movies". The integration of predefined justification explanations is the same as depicted in Fig. 9.11 for *learning* and *conceptual* explanations. A drawback of using predefined explanations is of course the vast amount of work necessary to cover all task available in the domain. Additionally, the content of justifications is limited to information known beforehand to the expert, and

especially in incomprehensible situations transparency explanations might be the better option. Transparency explanations are, though, always dependent on the inner processes of a system and a distinct transformation of those, for the user, to a comprehensible form. Hence, in contrast to justifications, transparency explanations have to be generated dynamically during run-time.

While previously conducted experiments (see [15]) showed that providing transparency explanations is the best way to deal with incomprehensible situations in HCI, this is also the most complicated way of explanation. Here, this means that the decision-making processes performed in the planning framework to generate a plan, in turn leading to the coarse structure of the dialog, might have to be explained to the user. Fortunately, the PF does include a *plan explanation* module [22], which focuses on so-called *Why* explanations, which in this case correspond to some form of transparency explanations. Their explanations describe, for example, why a certain plan step p is part of the plan P or the ordering of two plan steps in P. This module generates a logically sound explanation, which is guaranteed by using a technique based on formal proofs. Specifically, they use the task in question p_e, the plan P, its decomposition structure, to generate a set Σ of first-order axioms, which in the end enables a logical interference. Basically, the plan explanation consists of an explanation on used causal links and their matching preconditions, as well as the used decomposition methods. However, the generated formal explanation still has to be transformed by the explanation management, in order to be understandable for human users. Hence, it is translated by the explanation manager using template-based natural language generation into human-readable text. This new generated content is then transmitted to the information model in the knowledge base. The output the explanation manager generates is in either way a new dialog step, which references content from the information model and is sent to the DM to be included in the ongoing interaction.

9.4.4 Intertwining with the Architecture

For information presentation the dialog goal is passed on to the IM (cf. Fig. 9.9) and by that transferred to an XML-based format called *dialog output*. Hereby, the dialog content is composed of multiple information objects referencing so-called *information IDs* in the information model. Each information ID can consist of different types (e.g. text, audio, and pictures) which are selected and combined at run-time by a fission sub-component to compose the user interface in a user- and situation-adaptive way (see Chap. 10 for more details).

After the user interaction, the DM receives the interaction results from the multimodal fusion. The results are then analyzed and if the results are related to the desired plan step effects implemented by the dialog step, these effects are transmitted to the knowledge base as observations. Additionally, the plan execution component is notified that the current plan step has been processed by the dialog management.

9.5 Conclusion

In this chapter we presented different approaches for a user-adaptive and assistive dialog management approach. Although some attempts were made to make the rule-based dialog model more flexible, for example by using so-called dialog widgets, which implement reusable parts of a dialog (e.g. confirmations) [1], the main parts of the rule-based dialog are still rigidly predefined finite-state automatons. Additionally, due to the knowledge base storing information a major part of which originates from sources prone to uncertainty, another approach using probabilistic methods was developed. This general probabilistic DM approach contributes to a more coherent and flexible *Companion*-System.

The modelling of the user's knowledge is made in a realistic fashion, considering the uncertainty of one's knowledge distribution, though the process of updating the knowledge over time is currently not. While presenting an explanation to the user does increase the chances of understanding, it does not guarantee it. Therefore, the update process of one's knowledge values should integrate uncertainty as well, e.g. using information indicating understanding (e.g. user affective states like engagement, interest, and disposition). Nevertheless, we presented an approach which may integrate such information in the future and by doing that facilitate a more individual, co-operative and reliable system that is a trustworthy and understandable partner for the user.

Acknowledgements This work was done within the Transregional Collaborative Research Centre SFB/TRR 62 "*Companion*-Technology for Cognitive Technical Systems" funded by the German Research Foundation (DFG).

References

1. Bertrand, G., Nothdurft, F., Minker, W.: "What do you want to do next?" Providing the user with more freedom in adaptive spoken dialogue systems. In: 2012 8th International Conference on Intelligent Environments (IE), pp. 290–296 (2012)
2. Biundo, S., Wendemuth, A.: *Companion*-technology for cognitive technical systems. Künstl. Intell. **30**(1), 71–75 (2016). Special Issue on Companion Technologies
3. Cheverst, K., Byun, H.E., Fitton, D., Sas, C., Kray, C., Villar, N.: Exploring issues of user model transparency and proactive behaviour in an office environment control system. User Model. User Adap. Inter. **15**, 235–273 (2005)
4. Dzindolet, M.: The role of trust in automation reliance. Int. J. Hum. Comput. Stud. **58**(6), 697–718 (2003)
5. Fernandez, A.J., Hortala-Gonzalez, T., Saenz-Perez, F., Del Vado-Virseda, R.: Constraint functional logic programming over finite domains. Theory Pract. Log. Program. **7**(5), 537–582 (2007). doi:10.1017/S1471068406002924. http://dx.doi.org/10.1017/S1471068406002924
6. Glass, A., McGuinness, D.L., Wolverton, M.: Toward establishing trust in adaptive agents. In: IUI '08: Proceedings of the 13th International Conference on Intelligent User Interfaces, pp. 227–236. ACM, New York (2008)

7. Honold, F., Bercher, P., Richter, F., Nothdurft, F., Geier, T., Barth, R., Hoernle, T., Schüssel, F., Reuter, S., Rau, M., Bertrand, G., Seegebarth, B., Kurzok, P., Schattenberg, B., Minker, W., Weber, M., Biundo, S.: Companion-technology: towards user- and situation-adaptive functionality of technical systems. In: 10th International Conference on Intelligent Environments (IE 2014), pp. 378–381. IEEE, New York (2014). doi:10.1109/ie.2014.60
8. Larsson, S., Traum, D.R.: Information state and dialogue management in the trindi dialogue move engine toolkit. Nat. Lang. Eng. **6**(3&4), 323–340 (2000)
9. Lee, C.J., Jung, S.K., Kim, K.D., Lee, D.H., Lee, G.G.B.: Recent approaches to dialog management for spoken dialog systems. J. Comput. Sci. Eng. **4**(1), 1–22 (2010)
10. Lim, B.Y., Dey, A.K., Avrahami, D.: Why and why not explanations improve the intelligibility of context-aware intelligent systems. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '09, pp. 2119–2128. ACM, New York (2009)
11. McTear, M.F.: Spoken dialogue technology: Enabling the conversational user interface. ACM Comput. Surv. **34**(1), 90–169 (2002). doi:10.1145/505282.505285. http://doi.acm.org/10.1145/505282.505285
12. Muir, B.M.: Trust in automation: Part I. Theoretical issues in the study of trust and human intervention in automated systems. In: Ergonomics, pp. 1905–1922. Taylor & Francis, London (1992)
13. Müller, F., Späth, C., Geier, T., Biundo, S.: Exploiting expert knowledge in factored POMDPs. In: Proceedings of the 20th European Conference on Artificial Intelligence (ECAI 2012), pp. 606–611 (2012)
14. Nguyen, A., Wobcke, W.: An agent-based approach to dialogue management in personal assistants. In: Proceedings of the 10th international conference on Intelligent user interfaces, pp. 137–144. ACM, New York (2005)
15. Nothdurft, F., Minker, W.: Justification and transparency explanations in dialogue systems to maintain human-computer trust. In: Proceedings of the 4th International Workshop On Spoken Dialogue Systems (IWSDS). Springer, Berlin (2014)
16. Nothdurft, F., Bertrand, G., Heinroth, T., Minker, W.: GEEDI - guards for emotional and explanatory dialogues. In: 6th International Conference on Intelligent Environments (IE'10), pp. 90–95 (2010)
17. Nothdurft, F., Richter, F., Minker, W.: Probabilistic human-computer trust handling. In: Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL), pp. 51–59. Association for Computational Linguistics, Philadelphia, PA (2014). http://www.aclweb.org/anthology/W14-4307
18. Parasuraman, R., Riley, V.: Humans and automation: use, misuse, disuse, abuse. Hum. Factors J. Hum. Factors Ergon. Soc. **39**(2), 230–253 (1997)
19. Picard, R.W., Picard, R.: Affective Computing, vol. 252. MIT, Cambridge (1997)
20. Rao, A.S., Georgeff, M.P.: BDI agents: from theory to practice. In: Proceedings of the First International Conference on Multi-Agent Systems, ICMAS-95, pp. 312–319 (1995)
21. Sanner, S.: Relational dynamic influence diagram language (RDDL): language description (2010). http://users.cecs.anu.edu.au/ ssanner/IPPC2011/RDDL.pdf
22. Seegebarth, B., Müller, F., Schattenberg, B., Biundo, S.: Making hybrid plans more clear to human users – a formal approach for generating sound explanations. In: Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS 2012), pp. 225–233 (2012)
23. Wendemuth, A., Biundo, S.: A companion technology for cognitive technical systems. In: Esposito, A., Vinciarelli, A., Hoffman, R., Müller, V.C. (eds.) Proceedings of the EUCogII-SSPNET-COST2102 International Conference (2011). Lecture Notes in Computer Science. Proceedings on Cognitive Behavioural Systems, Dresden (2012)
24. Williams, J.D., Young, S.: Partially observable Markov decision processes for spoken dialog systems. Comput. Speech Lang. **21**(2), 393–422 (2007)
25. Zeigler, B., Bazor, B.: Dialog design for a speech-interactive automation system. In: Second IEEE Workshop on Interactive Voice Technology for Telecommunications Applications, 1994, pp. 113–116. IEEE, New York (1994)