COMMUNICATION FAILURES IN THE SPEECH-BASED CONTROL OF SMART HOME SYSTEMS

A. Oulasvirta*, K.-P. Engelbrecht*, A. Jameson†, S. Möller*

*Deutsche Telekom Laboratories, TU Berlin, Germany †DFKI, Germany

oulasvir@hiit.fi, klaus-peter.engelbrecht@telekom.de, jameson@dfki.de, sebastian.moeller@telekom.de

Keywords: Smart home systems, spoken dialogue systems, communication failures, error analysis.

Abstract

Despite their basic attractiveness as an interaction paradigm for controlling intelligent environments, the design of spoken dialog systems for this purpose raises some usability challenges that require careful attention. This paper examines closely the communication failures that can occur in the control of one particular type of intelligent environment: a smart home system that provides control for multiple domestic devices through a state-of-the-art mixed-initiative spoken-dialog interface. The 24 participants completed several tasks with the *INSPIRE* system in a controlled experiment, and interaction failures were categorized with an error taxonomy that is related to more general error taxonomies but specialized to this class of systems. Despite efforts devoted to supporting natural, mixed-initiative dialog and to the prevention of communication failures, over one fourth of user utterances were problematic, often leading to stagnation or regression. The causes and consequences of these problems are discussed, along with their implications for the design of spoken dialog systems for intelligent environments.

1 Introduction

A common assumption in connection with intelligent environments is that as interaction becomes more tightly interwoven with users' environments, interfaces should be more "natural". From this perspective, speech-based control is viewed as an attractive interaction paradigm, especially for situations where users are doing other things at the same time and cannot dedicate all cognitive resources to commanding a computer; when an expressive vocabulary already exists, when background noise is low, and when it is socially acceptable to speak to a machine. Speech-based control systems exploit the numerous ways our languages have to refer to action, objects, and events as they appear in time and space.

The richness of previously learned expressions and situational ways to modify them, which is considered an important argument for speech-based control, also raises a problem: The user has to "know what to say" [17]. Discussions of this problem pervade the literature on spoken dialog systems (SDS) ([7] and [9]), and various design strategies have been suggested (e.g., [5]).

But simply being aware of these strategies does not enable the designer to avoid the occurrence of user input which cannot be handled by the system. The most effective strategies tend to depend on the details of the individual case, and finding the best approach in each case can involve trial and error on the part of the designer.

In this paper, we aim to apply a strategy of "divide and conquer." We focus on a particular type of intelligent environment: the smart home; and our detailed data concern specifically a dialog system called *INSPIRE*. As we explain in Section 2, the design of the system was relatively innovative, but it was based on general principles of good dialog design and on extensive data collection. Using a substantial body of data collected in one study, we identify the types of interaction failures input that tend to occur in such a system, characterizing them more concretely than would be possible in the context of a more abstract discussion of the subject. For each type of failure, we discuss ways of preventing and responding to problematic user inputs that seem appropriate in the context of smart home control. We aim to show that this focus on a particular type of system makes it possible to achieve tangible progress, despite the inherently difficult nature of the general problem.

Because it is widely used in literature, we will use the term *error* when referring to interaction failures. This usage by no means implies that the interaction problem is the fault of the user – indeed it is often the fault of a system designer who did not cater for the needs and the behaviour of the user.

1.1 Related Work: Communication Failures

It has long been known that communication with situated systems such as intelligent environments raises novel challenges, compared to other human-computer communication paradigms. Many of the design solutions that have been worked out for GUIs are not available with sensing systems. Bellotti et al.'s [2] analytical framework includes questions such as that of how a user can *address* the system in the first place and how he or she can be sure that the system is *attending* to the user's actions. The most

relevant question of this framework for our purposes concerns what they call *Action*: How can the user effect a meaningful action, control its extent and possibly specify a target or targets for the action? The authors note that, in the absence of a visible interface and the possibility of direct pointing and manipulation, it can be difficult to specify objects and actions correctly. This paper presents a more detailed picture of problems of this sort that arise with a specific type of intelligent environment (smart homes) and a particular interaction modality (speech).

One example of a study that focused on errors and recovery strategies in a spoken dialogue system [3] concerned a system for reserving conference rooms. It focused on *nonunderstanding* errors – cases in which the system is unable to assign (with some minimal degree of confidence) an interpretation to an utterance of the user [3]. Two of the four levels that were identified concerned low-level problems of speech *recognition and segmenting of the audio signal*, respectively. These levels account for about 65% of the nonunderstanding problems that were observed with the system. In the study reported in the present paper, to focus more on communication failures rather than implications of speech recognition failures, we remove these types of error from consideration by using a Wizard-of-Oz paradigm for data collection, which eliminates speech recognition errors.

Out-of-application errors were also distinguished [3], in which the user refers to entities outside of the system's domain or requires functions that the system is unable to provide – and *out-of-grammar errors* – in which the meaning of the user's utterance is acceptable but the way in which it is formulated is not in accordance with the system's grammar and lexicon. These two kinds of errors are also included in the analysis that we present below, but we will argue that additional categories are needed to capture essential aspects of communication failures.

At the most general level of analysis, theories of human action and error (e.g., [11], [12], [15], and [16]) cover a wide variety of errors in interactive systems, ranging from the misconceptions of novices to the slips of experts. Understandably, error analysis has been most energetically applied in connection with safety-critical systems (see, e.g., [8]), and, to our reading, has not been utilized in the domain of smart home control. One of the main goals of the present paper is to adapt and apply these notions of error to interaction with intelligent environments. Work toward this goal has involved going through data with various error taxonomies and sculpting a modified version that covers a large proportion of the types of errors that have been observed. The result is a taxonomy that captures both errors related to communication of intention and misunderstanding of the system.

1.2 Approach

To address this problem, we made the following methodological choices:

- *Realistic experimental situation.* For reasons of ecological validity, we took pains to stage and decorate a laboratory as a living room that gives the look and feel of an ordinary living room.
- *State-of-the-art interface.* In order the make the evaluation relevant to contemporary trends in the design of SDS, we deployed a system that features advanced techniques like mixed-initiative interaction.
- *Wizard-of-Oz study.* In order to focus on other problems than those stemming from recognition errors alone – in particular, problems for which some solutions may be relatively specific to smart home systems – we used the Wizard-of-Oz methodology, where a researcher types in the user's spoken commands for the system, thus bypassing the system's error-prone speech recognition component.
- *Three interface variations.* Although we studied just one system, we decided to have three alternative metaphors to control it. Types of error that are found across the three interfaces are likely to occur with some regularity in the application domain and to be less intertwined with the specifics of the interface.
- *Task-based analysis*, although some (e.g., [1]) have noted that task-based analyses might not be appropriate for the evaluation of ubiquitous systems, the identification of an error critically relies on knowing the goal of the user and is thus best studied in a context where those can be controlled by the experimenter.

2 The INSPIRE System

The *INSPIRE* smart home system has been developed in an EU-funded IST project. It provides German language speech control over a number of domestic appliances (TV, video recorder, electronic program guide, three lights, blinds, fan, answering machine) through a unified dialogue structure. Several design choices were made with the goal of achieving natural interaction. The dialogue manager allows for mixed initiative, and the system parses also incorrectly formulated sentences and proceeds on the basis of incomplete information.

The system has not been built for commercial use but to experiment with unified dialog strategies and to explore user behavior in an idealized spoken language smart home setting. The experiment was conducted in a laboratory decorated as a living room, the speech recognition module being replaced by a Wizard of Oz.

2.1 Dialog Flow

The *speech signal* originating from the user is captured by a microphone array and enhanced using a beamformer, including noise reduction and echo cancellation. This enhanced speech signal is transferred to a commercial speech recognizer. However, for the experiment, a human transcriber (i.e., the wizard) replaced the recognizer.

The *speech understanding* module matches possible surface forms (keywords) to canonical values (concepts). This simple form of speech interpretation proved to be

sufficient for the given purpose, although it shows problems in the grammatical analysis of the user's speech. Such problems were reduced by augmenting the keywords with regular expressions, but could not completely be avoided with the present design.

To appreciate how dialog works in *INSPIRE*, consider the following excerpt from our data ("S" = System, "U" = user):

```
S: What else can I do for you?

U: Please switch on the standard lamp also.

S: I understood lamp for the device and switch
on for the task. Where is the lamp that you
want to operate?

U: What options are there?

S: Possible positions are: on the left of the
sofa, on the right of the sofa, or on the
right-hand side on the table.

U: Switch on the lamp on the left of the sofa.
```

The dialog between user and system is generated from *generic dialog nodes* linked by an application-independent branching logic (cf. [4]). Each node is associated with a specific piece of information to be gathered from the user. Depending on the previous user input, the node generates a prompt when it gets activated by the branching logic: A *general prompt* asking for the respective information, an *error prompt* in the case where nothing was recognized (no input) or interpreted (no matching concepts) from the user utterance, or a *help prompt* in the case where the user seems to be lost in the dialog or explicitly asks for help. The branching logic defines which nodes are open to collect information at each point in time, acquires tasks from the database which fit the concepts extracted from the user utterance, asks for clarification when no matching task exists, or asks for additional information when more than one task matches the user's specifications. Thus, the user is allowed to take the initiative and provide more than one piece of information in each utterance.

For *dialog management*, generic dialog nodes were defined and instantiated according to the piece of information to be gathered (cf. [14]). Using these nodes, a mixed-initiative dialog between the user and the system is possible. The dialog manager accesses a task model in the form of a database; this database defines the domestic devices and the actions which are under the user's control. The designers of the system considered all possible ways they could imagine addressing the devices, and these were implemented in the database as task models.

Generic error recovery strategies (e.g., repeating the prompt while adding some additional information) tend not to be optimal for all types of errors; the provision of recovery mechanisms that are appropriate for particular types of errors, while requiring more design and implementation effort, seems to produce better results [3]. Therefore, in adhering to the *mixed-initiative approach*, *INSPIRE* tries to make understandable what it has recognized of the user's command.

The dialog flow is consistent across all devices, the system typically asking first for the device the user would like to operate, then for its location (in case that there are several devices of the same type), then for the action to perform on this device, and finally for additional attributes which may be necessary to fully specify the task.

Speech output is generated from prerecorded speech units uttered by a male speaker, using concatenation templates. These templates define the concatenation of full sentences, of phrases, or of individual expressions or words. The concatenation is carried out without further signal manipulation, sometimes resulting in perceptible disfluencies at the concatenation points.

2.2 Other Steps Taken to Minimize Communication Failures in INSPIRE

In addition, there are a number of general strategies that were applied to minimize communication failures.

1. Providing introductory material that conveys a realistic mental model of the dialog system. The system should be a "walk-up-and-use" SDS. Although a smart home system is not very typical of this category of systems, in that the main users can use it over a long period of time, it was decided to built a system which does not require specific knowledge on the part of the user, because it should be operated by a potentially larger group of persons, with different abilities and with different types of "mental model" of the system – for example, visitors in a smart home as well as the residents themselves. For the sake of the experiment, a short introductory text was given to the participants as to the capabilities of the system (mainly the devices which would be operated, not which actions could be performed). In addition to that, a short story was distributed as a mind setting for the subjects.

2. Providing prompts that suggest possible inputs. In order to provide suitable prompts where the user knows at each point of the dialog what to say, the prompts were iteratively designed so that participants' problems were observed with a semifunctional prototype, and their reactions were taken into account in the prompt design. For example, many users were not aware of the fact that they had to specify the number related to a TV show displayed on the screen, and the prompt was rerecorded, putting a stronger emphasis on the word "number".

3. Specifying the dialog structure, grammar, and vocabulary in a way that corresponds to users' expectations. In order to improve the vocabulary and the "grammar" of the system, initial tests were carried out at different sites where users had to specify in textual form how they would address the *INSPIRE* system. At that point in time, the system was only roughly specified by its functionality, but the users could not experience it. In a second step, guided walkthroughs were carried out where a user sat next to the experimenter and dictated what he/she would say to the system, the experimenter corrected the utterance to make it understandable to the system, and both observed the reaction of the system (next prompt and potential system action) in order to proceed with the dialog. Data collected

in these two ways was used for the definition and refinement of the system's vocabulary and grammar.

4. Supplying, where possible, nonspeech prompts and feedback that help to suggest appropriate utterances. In this case, the only nonspeech prompt currently used is a list of television shows that is displayed on a screen so that the user can specify a show by its number.

2.3 Interface Variations: Three Agent Metaphors

In addition to dialog design, we implemented three *communication agent metaphors* to create conditions where interface-specific errors could be distinguished from more general types of error. Three different metaphors were implemented which differ with respect to the output modality, the system voice, as well as the sound direction:

1. *Multiple intelligent devices*: Each of the addressed devices is "intelligent" in that it is able to maintain a spoken interaction with the user, using a different voice. The sound is reproduced near the location of the device. (But the devices are still referred to in the third person, the system's utterances being the same as those used in the other two metaphors.)
2. *A single visible assistant* or servant which operates the devices on behalf of the user. This assistant is visible in terms of the talking head on the screen facing the user. The sound is reproduced by a loudspeaker near that screen.
3. *An invisible assistant*, similar to the visible one, but immaterial and invisible like a "ghost" somewhere in the room. The sound is presented from a number of loudspeakers at different locations of the room, generating a diffuse sound field.

Three output media could be utilized: 1) speech output generated by concatenating pre-recorded speech units, and played back through different loudspeakers installed in the test room, at a sound pressure level of approx. 79 dB(A) at the position of the user; 2) graphical output generated on a computer screen mounted on a wall of the test room; this type of output was used for presenting lists of items the user could select from; 3) an animated video supporting the speech output, showing the head and torso of a real male person who is moving his lips, synchronized with the activity of the speech signal (the assistant metaphor).

3 Method

A controlled laboratory experiment was carried out at the test site of Ruhr-University Bochum consisting of a room (5.7 x 3.6 m) with furniture typical for a living room (couch, armchairs, a low table, shelves), and equipped with the devices controllable by the *INSPIRE* system. An additional control room hosted the experimenter, the transcribing wizard replacing the speech recognizer.

3.1 Participants

Twenty-four native Germans (10 female, 14 male) participated in the test, mostly students or employees of the university. They were 19–29 years old, with a mean of 23.7 years. Sixteen of them had previous experience with SDS, two with speech recognition, and 14 with synthesized speech. All participants were paid for their effort.

3.2 Materials

In order to create a meaningful setting for the experiment, three *scenarios* were developed, each containing 9-11 tasks. The tasks were linked in the form of a short story and explained in an *indirect* way (e.g. "You return home from work and feel that it is quite hot in the living room. Please use the fan to get some fresh air.") in order to avoid direct priming of the user's vocabulary.

Each task addressed a specific device and action, although freedom for own decisions is left for some tasks (e.g. selecting a film). Furthermore, the tasks were not trivial but addressed several problems of spoken language control, like specifying location, specifying times or authenticating oneself. The overall number and type of tasks was similar in all scenarios, leading to three interactions of comparable complexity. Tasks were presented as an entire story before the interaction, and on paper cards for recall during the experiment.

A *user experience questionnaire* to be filled in after each scenario consisted of 37 statements grouped under 7 categories designed according to a Recommendation issued by the International Telecommunication Union, ITU-T, for speech-based services. (We report the exact wording of statements and the observed relationship between usability judgments and errors in another paper [13].)

3.3 Procedure

The experiment required three scenario-guided interactions to be carried out with the *INSPIRE* system. It comprises five parts: 1) A written and oral introduction to the system and to the purpose of the experiment; 2) an initial questionnaire through which general information on the test participants and their background were solicited; 3) a short story illustrating the use of the system by a couple at home, serving as a kind of mind-setting to the participants; 4) three scenario-guided interactions with the system, each followed by a questionnaire on different quality aspects; and 5) a final questionnaire where the participants were asked to rate their general impression of the system at the end of the experiment.

3.4 Design

Each participant carried out an interaction with each system condition (metaphor) and each scenario. The test design was partially balanced; i.e., the order of scenarios and conditions changed independently for each user, to reduce

as far as possible the effect of the scenario, the test condition, and the order of the dialog within the experiment.

3.5 Categorization of Errors

All tasks in the experiment involve a transformation or manipulation of an object (digital and artifactual) in the smart home system, achievable through a hierarchy of commands given to the system (as defined by the relevant dialog structure). For each task there is an optimal solution path or many paths through the dialog tree to a goal state, and we know these as experimenters. Relevant to our argumentation, we had control of user goals as we designed the task scenarios that the participants were to accomplish. We here use the term "error" broadly to refer all *deviations from optimal task solution paths*. By definition, errors inhibit the progress towards the goal of the interaction (partial progress, stagnation or regression).

GOAL-LEVEL
$=_{df}$ The system does not posses the function or capability assumed in the request. Subcategories: asking the system to control 1) objects that are not in the system, 2) at a level of granularity not possible, 3) in a way that is not possible due to extra-systemic restrictions.
TASK-LEVEL
$=_{df}$ Issuing a command that is progressive in one state of the dialogue, but not in the current one. Subcategories: 1) progressive command valid in a future state in the optimal solution path, 2) unprogressive command valid in a previous state.
COMMAND-LEVEL
$=_{df}$ Issuing a command that would be valid if one word was changed to its synonym or the grammatical order of words was changed, without changing the meaning of the utterance. Subcategories: 1) word (verb, noun, adjective, adverbial) poor phrasing error, 2) grammatical construction error.
CONCEPT-LEVEL
$=_{df}$ Issuing a command that would be valid if the system represented the world in a different way. It is possible to imagine another kind of model/categorization of the world in which this utterance would not constitute an error. Subcategories: referring incorrectly to 1) time, 2) space, or 3) attribute of an object.
OTHER
$=_{df}$ All other items recognizable as errors. Subcategories:
1. *No input error* $=_{df}$ Failing to issue a command during the timeout interval in which the system expects it to be issued.
2. *Common ground error* $=_{df}$ Issuing a command that refers to outcomes of previous states (e.g., "Please switch on *the other* lamp")
3. *Wizard error* $=_{df}$ The wizard typed the user's command incorrectly, or there was a problem with the computer.
4. (Other)

Table 1: Error categories and their definitions

For the errors, a new taxonomy of user errors custom-tailored for a SDS was constructed, making a distinction among 1) goal-level (i.e., misunderstanding the capabilities of the system), 2) task-level (i.e., not understanding *how* to reach the goal in interaction with the system), 3) command-level (i.e., vocabulary and grammar errors), and 4) conceptual errors (i.e., referring to the world in a way that is not understood by the system).

The *unit of analysis* for spotting errors is *one exchange of information* between the system and the user. For any system prompt, there is always at least one user response that lies on the optimal solution path. Because an error can affect only part of the information the user tries to convey to the system, one utterance can contain more than one error. In making interpretations of these kinds of errors, the categorization primarily deals with overt behavior, the user utterance, the task given to the user which is known to researchers, and the optimal path we also know as developers of the system. Errors such as goal-level errors can be recognized by comparisons of these three. Table 1 presents the categorization and typical subcategories distinguished among.

The error categorization is somewhat similar to more general conceptual frameworks proposed for describing HCI. However, the definitions and subcategories have arisen bottom-up and they have been optimized for a smart home environment controlled via a mainly speech-based interaction. It remains to be shown that a similar categorization can be applied to other intelligent environments, involving other concepts than "devices" and "actions", and potentially other interaction modalities. For non-speech-based systems, at least the command-level errors (vocabulary and grammar) will have to be adapted. Initial analyses described in [13] show that the error frequencies are correlated with user ratings of system quality; thus, the number and nature of errors coincides with users' negative perception of the system.

3.6 Consequences of Errors

We were also interested in the consequences of errors, and the fact that we know the task and the dialog structure gave us an easy operationalization for this:

1. *Stagnation*. The system takes the user to a prompt that is as close to the task goal as the previous prompt, i.e., the goal can still be reached with as many steps as before. Two special cases of this are called *Repetition* and *Rephrasing*: The system repeats the prompt (word to word or just the end of it but meaning the same thing and being pragmatically the same prompt with same action alternatives). A third special case is *Help*-prompt, in which possible utterances are proposed to the user.
2. *Regression.* The system goes to a state that is farther away from the task goal than the previous state; i.e., the user has deviated from an optimal solution path and now has to go through at least one extra state in order to achieve the goal. A special case of this is called *Restart*: The system returns to its initial state, losing any progress achieved in the task before the error occurred.
3. *Partial Progress.* The system goes to a state which is closer to the task goal, but not all the information in the utterance is processed (thus the term partial).

3.7 Coding Procedure

Several initial sessions were held for the definition of the error categories. After agreeing on the general scheme presented in Table 1, one of the authors started to code the whole data set. Five *calibration sessions* were held altogether in refining the categories when problematic in-

Error category	% of all exchanges	*Consequence for dialogue flow*				
		Regression	Repeating/ rephrasing	Help prompt	Partial progress	Mixed/None
Goal-level	4.0	7%	21%	7%	57%	8%
Task-level	5.9	11%	28%	12%	34%	15%
Command-level	12.8	7%	22%	31%	39%	1%
Concept-level	7.1	10%	19%	29%	35%	7%
Other	2.4	N/A	N/A	N/A	N/A	N/A

Table 2: Distribution of error types per exchange and a breakdown of consequences to dialogue flow.

stances appeared. After each change, the category in question was recoded in the data to ensure reliability of coding.

The data of one participant could not be analyzed because of technical problems. This resulted in the final data set consisting of 2343 exchanges, which was coded in its entirety.

3.8 Reliability

To assess the *reliability* of the taxonomy, an outside coder was hired to code 300 exchanges randomly sampled from the data. She was trained to use the coding scheme, and several examples were provided that were not part of the to-be-coded sample. Overall, we were satisfied with the reliability for the error categorization.

In calculating an *inter-coder reliability measure* between the first and the second coder, we found that all four categories show Cohen's Kappa of over .60, which is considered to be an appropriate threshold for claiming substantial inter-rater agreement (e.g., [10]). Nevertheless, there were subtler difficulties *within* categories that had only few data points. For example, Time and Space subcategories were much less reliable than the Attribute subcategory in concept-level errors. Generally, the Consequence category showed poorer agreement (Kappa falling in the range .19-.58) than error categories. Because of the poorer agreement for the consequence categories, we sustain from deeper analyses than reporting them per error categories (Table 2).

4 Quantitative Results

The users were 99.2% successful in accomplishing the tasks given in the scenarios. On average, 35 system–user exchanges were needed to accomplish a whole scenario (SD 6.36). This corresponds to an average of between 3 and 4 commands per task, which seems to be reasonable considering that several pieces of information have to be provided for each task. A user utterance had on average 3.0 words, whereas an utterance by the system had 16.6 words on average, reflecting its wordiness. On average, it took 11.1 minutes to accomplish a scenario (SD 3.2 min), each comprising of 9-11 tasks on the devices. The quickest user accomplished a scenario in 6.0 minutes, although the users were told to carry out the scenarios at their own pace, and no particular encouragement was given to be quick.

The coding revealed that 26% of exchanges involved one or more errors of the four types. (Note that one exchange can contain more than one error, which is why Table 1's error figures add up to more than 26%.) Of all errors, 12% goal-level errors, 18% task-level errors, 40% were command-level errors, and 22% concept-level errors. (2.4% of utterances were categorized as "other" error/problem.) Because of an error, dialog flow was stagnated as a consequence in 49% of cases (repetition/rephrasing of prompt or a help prompt); in 11% there was regression, and in 36% partial progress in the task was achieved despite the error. Table 2 above presents a more accurate breakdown.

Weak to moderate correlations were found between the number of errors in a given category and the time needed for accomplishing a scenario. With command-level errors, this correlation was $r = 0.45$; with each of the other error categories, the correlation was lower, in the range $.23 < r < .28$.

The appliances differed in terms of errors. Controlling the TV and lights was particularly problematic, 39% of utterances for both appliances contained one or more errors. Program guide and VCR involved 31% and 24% errors, respectively, whereas answering machine (21%), fan (15%), and blinds (13%) had notably fewer errors.

Supporting the idea that error categories are general, there were no clearly significant differences in the four categories between the three interface conditions ("agent metaphors"). Therefore, we collapsed the data across the three interface conditions for the subsequent analyses.

5 Results Concerning Particular Error Types

Against the background of the quantitative analyses presented above, we will now discuss in turn each of the types of error distinguished in Table 1. We will refer back briefly to some of the relevant general properties of *INSPIRE*, but we will focus on measures that are specifically relevant to the type of error in question.

- Why did errors of this type occur with *INSPIRE* despite the measures taken to minimize them?
- What improvements to the design of *INSPIRE* are suggested by this analysis that might (further) reduce the incidence of this type of error?

5.1 Goal-Level Errors

Goal-level errors were found less frequently (as 12% of all errors) than other error types. Note that, since the tasks had been invented by a researcher familiar with the system, all of the high-level goals instructed to the users were basically achievable. We therefore do not have the sort of example that can arise when a novice user of a smart

home system thinks of high-level goals that are outside of the system's scope. Still, the user had some freedom of choice in terms of how she chose a method for goal achievement that consisted of subgoals; it is on this level of subgoals that goal-level errors occurred.

These errors consisted almost entirely (96%) of what we call *control mismatch errors*. One subtype involves mismatches between the content of the command and the capabilities of the *INSPIRE* dialog manager. A typical (and rather frequent) example involves commands that involve two of the three lamps (e.g., "Please switch on two lamps"). In fact, manipulating two lamps requires two separate commands; but this fact is not easy for the user to guess, because in fact it is possible to operate *all* (in this case, three) lamps in the room with one command (e.g., "Switch on all of the lamps").

In fact, this latter type of command is an example of one way in which the designers of *INSPIRE* tried to avoid control mismatch errors: by supporting commands that users seemed likely to desire and expect – in particular, those that were observed during the early data collection efforts. As this example shows, an attempt to accommodate one user expectation can prevent one type of error but at the same time lead to other errors that might not have occurred otherwise, by raising expectations about what the system can understand. In this sense, a smart home system that tries to accommodate more and more user expectations is like a person trying to catch up with his shadow.

Accordingly, one approach to avoiding this type of error is to keep the set of possible subgoals simple and easily learnable – for example, by enforcing the presumably easy-to-learn principle that each command must refer to a single object.

Another type of control mismatch involves commands that request an operation on a device that is not in fact available for that device. For example, one of the three lamps did not have a dimming function. Understandably, some users gave commands to have this lamp dimmed. Avoiding this type of error by informing the user in advance via speech about which operations were possible for each individual device would be likely to be tedious. A strategy applied in GUIs would be to ensure that the appearance of the devices themselves was such that even a user who was controlling a device remotely via speech could see what operations were supported – either on the basis of the device's physical form or on the basis of labels. But this strategy would tend to conflict with aesthetic requirements in a smart home, and it would be infeasible for complex devices such as TVs with electronic program guides.

It seems inevitable that, even with the implementation of strategies such as those just discussed, control mismatch errors involving device functionality will occur with some regularity in smart home systems, except perhaps with users who are quite familiar with the specific system (e.g., the residents in the home, as opposed to guests). Therefore, appropriate error handling is important. The general style of response of *INSPIRE* to this type of error – essentially, informing the user that the requested operation is not possible – does not appear especially problematic in the context of smart home operation.

5.2 Task-Level Errors

This category refers to errors that involve executing a valid command at an inappropriate point in the dialog flow. In other words, the user has a valid (sub)goal but does not know at what point(s) she can take the action in order to achieve it, given the dialog structure of the system. In 72% of these cases, the user "jumped the gun", supplying some information at a point at which the system was not yet expecting it, as in the following example:

```
S: I understood movie as TV show type. Your
choice leaves several possibilities. Please
name the number of a title in the list on the
screen.

U: One, and signal the beginning.
```

Here, the user supplements a valid response ("One") with a command that the system is not expecting at this point in the dialog and can therefore not handle.

The design of *INSPIRE* actually aimed to avoid this type of problem with its mixed-initiative approach, which generally allows the user to give commands independently of the system's prompt. Therefore, even if the user changes her mind suddenly and gives a command that is unrelated to the preceding context, *INSPIRE* will usually be able to handle the command. One reason why errors of the type just illustrated nonetheless occurred is that this policy was not implemented with complete consistency: In a few dialog states, the system does accept only a limited number of possible inputs. A corresponding remedy would be to eliminate these exceptions as far as possible – in the present example, by having the system first recognize the number specified by the user and then treat the rest of the input as a separate command. On the other hand, complete input flexibility of this sort is much harder to realize successfully in terms of natural language processing and speech recognition (the latter of which did not play a role in our study, because of the Wizard-of-Oz design), than the processing of a limited range of inputs in each state. Once again, we have a case where the provision of some desirable dialog features tends to create an expectation that the system cannot consistently fulfill.

5.3 Concept-Level Errors

Time-related errors (11% of conceptual errors) occurred in two different ways. First, some users categorized time differently than the system, for example following the clock instead of using terms like *morning* and *evening*. The second type of "error" (actually due to a system limitation) is to refer implicitly to the current time, which the system cannot understand because it is not aware of the current time. For example, users asked the system to start recording "now" or to record a show that "just started". However, referring to the current *date* ("today", "tomorrow") works, which might be confusing. In this context it

might be noted that users never specified a day using date, but always used the *relative* terms.

Here again, the problem can be seen as one of inconsistency on the part of the system: It supports natural, intuitive temporal expressions in some cases but not in others. In contrast to the examples of inconsistency discussed above, where total consistency would be hard to achieve, this appears to be a case where good coverage could be attained with reasonable effort, including giving the system a concept of the current time and including all of the frequently occurring expressions in the vocabulary.

Space- and place-related errors (35% of conceptual errors) are a more complex issue, since spatial expressions can take various forms and refer to various objects, including the user ("behind me") or other devices in the room ("on the table"). Some users referred to the standard lamp as the "lamp on the bottom", which the system did not understand. Furthermore, users occasionally *underspecified* the location, meaning that they uttered a phrase which described the location of two lamps at the same time (e.g., "The one on the right", while there were two lamps to the right of the user). The system's response is to list possible unambiguous formulations (e.g., "Possible positions are: 'to the left of the couch', 'to the right of the couch', and 'to the right, on the table'"). Given the difficulty for the system of understanding all possible forms of spatial description, this error-handling strategy may represent the best solution to the problem: It allows the user to achieve her goals without excessive delay, at the same time learning how to identify the objects so that her next attempt can be successful right away.

Finally, the rest of the conceptual errors (54%) were *attribute type errors*. Here is a typical example, involving the labels of categories of movies:

```
S: I understood TV for the device, show for
the task, today for the day, evening for the
time and record for the TV show action. What
type of program would you like to watch?

U: Please give me an overview of the programs.

S: What type of program would you like to
watch?

U: Science fiction.

S: Types of program are: movies, documenta-
ries, series, news or shows.

U: Movies.
```

The designers chose not to list the possible "types of program" in the prompt to avoid tedium for frequent users; and in fact, as this example shows, the consequences of an incorrect guess by a less experienced user are not serious, because of the system's error recovery method. Some errors of this sort could be avoided through the strategy of *tapering* [5][17], in which longer, explicit prompts are used for inexperienced users and gradually replaced by shorter prompts that presuppose familiarity with the possible inputs. To be sure, any strategy that involves adaptation to individual users presupposes that the system can identify each user and keep track of the interaction with him or her over time – a requirement that may not be easy to fulfill in all smart home systems.

A more interesting example where it seems desirable to treat less and more experienced users differently concerns errors in which a user refers to a TV show by name instead of (as is required by *INSPIRE*) by number: This error hardly ever occurs in cases where the system explicitly asks the user to choose a show by number from a list displayed on the screen. (The system prompt in this case was in fact adapted on the basis of early tests to emphasize the need to specify a number.) But the error does occur in response to open system prompts such as "What can I do for you?". More generally, the use of such open prompts, while advantageous especially for the expert user, can seduce the less experienced user into making some errors that would be prevented by the carefully crafted prompts that are available elsewhere in the system.

5.4 Command-Level Errors

Errors in this category concern discrepancies between the words and grammatical constructs used by users and those understood by the system. Of these command-level errors, 38% involved nouns and 26% involved verbs. An example involving a grammatical construction concerns the command "I would have liked to check my messages on the answering machine" given by one user, using a construction which is fairly common in spoken German in the sense of "I would like to check my messages" but which was not understood by the system.

During the development of *INSPIRE*, considerable effort was made to align the system's vocabulary with the vocabulary likely to be used by users, so that errors due to vocabulary discrepancies would be minimized: Extensive data collection was done at different sites and with different user groups. Also, the guideline was applied that the words used by the system should be ones that the system itself can understand when they are used by a user (cf. [7]).

Although future versions of *INSPIRE* will benefit from the addition of synonyms collected in this study, some vocabulary-related problems that involve multiple meanings of a word will remain. For example, the German term *Nachrichten* ("news") is problematic, because the same word can also occur with the answering machine (for "messages"). Another word with more than one meeting is *Licht* ("light"), which can mean either "lamp" or "brightness". Reliance on context for the disambiguation of such words partly conflicts with the goal of allowing users flexibility in what they can say at any time.

For this reason, for example, the designers introduced the term *Fernsehnachrichten* ("TV news"), which is rarely used in spoken German. Predictably, it took the users time to catch on to this unusual term. In such cases, an error recovery strategy like that of *INSPIRE*, which involves the identification of the problematic word and the provision of possible substitutes (either directly in the system prompt or via the help system) may be the best available solution, in that it minimizes the consequences of such errors and

allows users to learn the necessary words and expressions over time.

6 Conclusions

The preceding sections have yielded a number of ideas about how communication failures with the speech-based control of a smart home system can be minimized. Most of the points made can be generalized to some extent to the speech-based control of other ubiquitous computing systems and/or even to spoken dialog systems in general. Still, the analysis illustrates the usefulness of the strategy of focusing on one particular type of system: There is no "silver bullet" that can lead to great progress in dealing with the long-familiar and largely inherent communication problems associated with SDS, but as we have seen, when we focus on a particular type of problem in a particular type of system, specifically applicable solutions can often be identified.

In some cases, the results reported above make it clear that the design of the *INSPIRE* was suboptimal and that the specific problem in question is not likely to appear to the same degree in other, comparable systems. Even in these cases, we believe that the analysis of the problems encountered with *INSPIRE* can be instructive to designers: the same problems can appear in more subtle forms that are harder to recognize, and the same solutions are worth considering.

Acknowledgements

This study was supported by the EC-funded IST project INSPIRE (IST-2001-32746) and by the MeMo project funded by Deutsche Telekom AG. The authors would like to thank all colleagues who supported the research.

References

[1] G. Abowd, E. Mynatt, E. Charting past, present, and future research in ubiquitous computing, *ACM Transactions on Computer-Human Interaction*, Vol.7, No. 1, pp. 29–58, 2000.

[2] V. Bellotti, M. Back, W.K. Edwards, R .Grinter, A. Henderson, C. Lopez. Making sense of sensing systems: Five questions for designers and researchers. *Proceedings of CHI 2002*, pp. 415–422, 2002.

[3] D. Bohus, A. Rudnicky. Sorry, I didn't catch that! --- an investigation of non-understanding errors and recovery strategies. *Proceedings of the SIGdial Workshop on Discourse and Dialogue*, Lisbon, 2005.

[4] T. Bui, M. Rajman, M. Melichar. Rapid dialogue prototyping methodology. Proceedings of the 7th International Conference on Text, Speech and Dialogue, pp. 579–586, Berlin, Germany, 2004.

[5] M.H. Cohen, J.P. Giangola, J. Balogh. *Voice user interface design*, Addison-Wesley, Boston, MA, 2004.

[6] P.C. Constantinides, A.I. Rudnicky. Dialog Analysis in the Carnegie Mellon Communicator. Proc. 6th Europ. Conf. on Speech Communication and Technology (Eurospeech'99), Budapest, 1:243-246, 1999.

[7] L. Dybkjær, N.O. Bernsen. Usability issues in spoken dialogue systems. *Natural Language Engineering*, Vol.6, No.3–4, pp. 243-271, 2000.

[8] J. Galliers, A. Sutcliffe, S. Minocha. An impact analysis method for safety-critical user interface design. *ACM Transactions on Computer-Human Interaction*, Vol.6, No.4, pp.341-369, 1999.

[9] K. Klöckner, A. Jameson. The usability of deployed telephone dialog systems: An evaluation. *Proceedings of the 2006 IASTED International Conference on Computational Intelligence*, San Francisco, 2006.

[10] J. Landis, G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, Vol. 33, pp. 159-174, 1977.

[11] D. Norman. Design rules based on analyses of human error. *Communications of the ACM*, Vol. 26, pp. 254.258, 1983.

[12] D. Norman. Cognitive engineering. In D. Norman & S. Draper (Eds.), *User centered system design: New perspectives on human-computer interaction*, pp. 31-61, Erlbaum, Hillsdale, NJ, 1986.

[13] A. Oulasvirta, K-P. Engelbrecht, A. Jameson, S. Möller. The relationship between user errors and perceived usability of a spoken dialogue system. *The 2nd ISCA/DEGA Tutorial & Research Workshop on Perceptual Quality of Systems*. Berlin, Germany, 4-6 Sep, 2006.

[14] M. Rajman, T. Bui, A. Rajman, F. Seydoux, A. Trutnev, S. Quarteroni. Assessing the usability of a dialogue management system designed in the framework of a rapid dialogue prototyping methodology. *Acta Acustica united with Acustica*, Vol. 90, No. 6, pp. 1096-1111, 2004.

[15] J. Reason. *Human error*. Cambridge, New York, Cambridge University Press, 1990.

[16] C. Wickens, J. Hollands. Engineering psychology and human performance. Upper Saddle River, NJ, Prentice Hall, 2000.

[17] N. Yankelovich. How do users know what to say? *interactions*, Vol 3, No.6, pp. 32-43, 1996.