User Errors in Spoken Human-Machine Dialogue

Niels Ole Bernsen, Laila Dybkjær and Hans Dybkjær[1]

Abstract. Controlled user testing of the dialogue component of spoken language dialogue systems (SLDSs) has a natural focus on the detection, analysis and repair of dialogue design problems. Not only dialogue designers and their systems commit errors, however. Users do so as well. Improvement of dialogue interaction is not only a matter of reducing the number and severity of dialogue design problems but also of preventing the occurrence of avoidable user errors. Based on a controlled user test of the dialogue component of an implemented SLDS, the paper takes a systematic look at the dialogue errors made by users in the test corpus. A typology of user errors in spoken human-machine dialogue is presented and discussed, and potentially important dialogue design advice derived from the fact that the notion of a 'user error' turns out to be one that must be handled with care.[1]

1 INTRODUCTION

This paper is based on a controlled user test of the dialogue component of the Danish dialogue system which is an advanced spoken language dialogue system (SLDS). When analysing the data from such tests, the natural focus is on dialogue design errors. Such errors have to be identified and diagnosed, and ways of remedying them must be found whenever possible. Dialogue design errors cause problems of user-system interaction, make user task performance unnecessarily bumpy and generate user dissatisfaction with SLDS technology. However, not everything that goes wrong in the dialogue between user and system is due solely to errors made by the dialogue designers. Users also make errors during dialogue and some interaction problems are the compound effect of dialogue design errors and user errors. This paper proposes to take a systematic look at the dialogue errors made *solely* by users. Based on the user test material, we ask: of which types are the user errors that were identified? What are their likely effects on the success of the dialogue? What, if anything, can be done about them? And what is a "user error in dialogue" in the first place?

The Danish SLDS prototype is a ticket reservation system for Danish domestic flights. The system runs on a PC with a DSP board and is accessed over the telephone. It is a walk-up-and-use application which understands speaker-independent continuous spoken Danish with a vocabulary of about 500 words. The prototype runs in close-to-real-time and is representative of advanced current systems. Comparable SLDSs are found in [1,3,7]. The system has five main modules. The *speech recogniser* produces a 1-best string of words. The *parser* makes a syntactic analysis of the string and extracts the semantic contents which are represented in frame-like structures. The *dialogue handling module* interprets the contents of the semantic objects and decides on the next system action which may be to send a query to the *application database*, send output to the user, or wait for new input. In the latter case, predictions on the next user input are sent to the recogniser and the parser. *Output* is produced by concatenating pre-recorded phrases under the control of the dialogue module.

In what follows, Section 2 provides a description of the dialogue model for the Danish dialogue system and presents an example dialogue from the user test. The user test is described in Section 3. Section 4 presents an analysis of the user errors that were identified in the user test. Section 5 concludes the paper.

2 THE DIALOGUE MODEL

The dialogue model for the Danish dialogue system was developed by the Wizard of Oz (WOZ) experimental prototyping method in which a person simulates the system to be designed in dialogue with users who are made to believe that they interact with a real system [8]. The dialogue model had to satisfy the following technological constraints imposed by the speech recogniser: to ensure real-time performance, at most 100 words could be active in memory at a time; to ensure an acceptable recognition rate, an average and a maximum user utterance length of 3-4 words and 10 words, respectively, were imposed. Other design goals, such as linguistic naturalness, dialogue naturalness and dialogue flexibility had to be traded off against these constraints [4].

The WOZ dialogue model development was iterated until the model satisfied the design constraints. In each iteration, the dialogues were recorded, transcribed, analysed and used as a basis for improvements on the dialogue model. We performed seven WOZ iterations yielding a transcribed corpus of 125 task-oriented human-machine dialogues corresponding to approximately seven hours of spoken dialogue. The 94 dialogues that were recorded during the last two iterations were performed by external subjects whereas only system designers and colleagues had participated in the earlier iterations. A total of 24 different subjects were involved in the seven iterations. Dialogues were based on written descriptions of reservation tasks (scenarios).

The dialogue model resulting from the WOZ iterations is mixed-initiative. Domain communication is system-directed. *Domain communication* is communication within or about the task domain. Because of the strong limitations on active vocabulary size (see above), it was necessary during domain communication to leave the main initiative with the system. The system maintains dialogue initiative by concluding all its turns by a non-open question to the user, i.e. a question which asks for a well-defined piece of information, such as a choice between binary options, a date or time, or a destination. A field study was made of the most natural order in which to exchange the needed information. The implemented task structure conforms to the most common structure found in human-human domestic airline ticket reservation dialogues recorded in a travel agency. Whereas domain communication is system-directed, users can take the initiative in meta-communication with the system. *Meta-communication* is

[1] Centre for Cognitive Science, Roskilde University, PO Box 260, 4000 Roskilde, Denmark, emails: nob@cog.ruc.dk, dybkjaer@cog.ruc.dk, laila@cog.ruc.dk, phone: +45 46 75 77 11 fax: +45 46 75 45 02

communication about the user-system communication itself and is usually being undertaken for purposes of clarification or repair. Whenever needed, users may initiate meta-communication to resolve misunderstanding or lack in understanding by using one of the keywords 'change' and 'repeat'. The system initiates meta-communication by saying "Sorry, I did not understand" or by asking the user, after a long pause, "Are you still there?"

In addition to contributions to meta-communication and to the achievement of particular reservation tasks, the system provides two pieces of general information: (i) the system's introduction provides information on what the system can and cannot do and how to interact with it (Figure 1). (ii) An explanation is provided of the different types of discount that are possible on return tickets. In order not to waste the time of experienced users, the system provides this information only to novice users. Figure 3 shows a dialogue from the user test of the implemented system. The dialogue is based on the scenario shown in Figure 2. The user has already made one reservation and continues without making a new call, thereby avoiding the introductory phrases shown in Figure 1. Dialogue and scenario examples have been translated from the Danish.

S1: Hello, this is the DanLuft reservation service for domestic flights. Do you know how to use this system?
U1: No.
S2a: The system can reserve tickets for Danish domestic flights. You use it by answering the system's questions. In addition you may use the two special commands "repeat" and "change" to have the most recent information repeated or changed. The system will only understand you when you answer its questions briefly and one at a time.
S2b: Please state your customer number.

Figure 1. The introduction (S2a) to the Danish dialogue system. S means system, U means user.

Anders Bækgaard (ID-number 6), Paul Dalsgaard (ID-number 3) and Børge Lindberg (ID-number 4) work in a department in Aalborg that has customer number 3. They are all going to Copenhagen on the first weekend in February. They want to depart by the earliest flight on Saturday at 7:20 and return by the latest flight on Sunday at 22:40.

Figure 2. The scenario T32.

3 THE USER TEST

The user test was carried out with a simulated speech recogniser [2]. A wizard keyed in the users' answers into the simulated recogniser. The simulation ensured that typos were automatically corrected and that input to the parser corresponded to an input string which could have been recognised by our real speech recogniser. In this set-up, the recognition accuracy would be 100% as long as users expressed themselves in accordance with the vocabulary and grammars known to the system. Otherwise, the simulated recogniser would turn the user input into a string which only contained words and grammatical constructions from the recogniser's vocabulary and rules of grammar.

The test was based on 20 different scenarios which had been constructed to enable exploration of all aspects of the task structure. As the flight ticket reservation task is a well-structured task in which a prescribed amount of information must be exchanged between user and system, it was possible to extract from the task structure a set of sub-task components, such as number of travellers, age of traveller and discount versus normal fare, any combination of which should be handled by the system. The scenarios were generated from systematically combining these components.

Twelve novice subjects, mostly professional secretaries, participated in the user test. The subjects conducted the scenario-based dialogues over the telephone in their normal work environments in order to make the task as realistic as possible. The subjects were given a total of 50 tasks based on 48 scenarios. A *task* consists in ordering one or more tickets for one route. The number of recorded dialogues was 57 because subjects sometimes reiterated a failed dialogue and eventually succeeded in the task. A *dialogue* is one path through the dialogue structure. Subjects failed on seven tasks. The task transaction success rate thus was 43/50% = 86%.

Each dialogue was recorded and all transactions between the individual system modules were logged. The recorded dialogues were transcribed and analysed. The analysis aimed at detecting dialogue interaction problems and was done as follows. Based on the dialogue structure, a template was built which contained the system's questions. For each scenario, normative system questions and user answers were filled into the template. The key contents of the actual dialogues were then plotted into the template. Comparison between normative and actual system and user answers led to the identification of three major classes of interaction problems: (1) linguistic problems, (2) problems of dialogue interaction, and (3) other problems, such as system breakdown. (2) splits into (A) dialogue design problems and (B) user errors. The dialogue design problems (A) identified in the user test are presented and discussed in [6]. The following section focuses on describing and illustrating (B).

4. USER ERRORS

The concept of a 'user error' is a complex one, as we shall see. For now, let us define a user error as a case in which a user does not behave in accordance with the full normative model of the dialogue. In controlled user testing, the full normative model of the dialogue is defined by (i) explicit designer instructions to users, i.e. the scenarios that users have to carry out in dialogue with the system; (ii) explicit system instructions to users, such as the system's introduction to itself (Figure 1); (iii) explicit system utterances during dialogue; and (iv) implicit system "instructions" to users. (i) through (iv) will be illustrated and discussed below. (i) is absent in field testing of systems and in practical system use. In such cases, the full normative dialogue model reduces to (ii)-(iv). (ii) is very important to SLDS design and use because SLDSs are vastly inferior dialogue partners when compared to humans. This means that humans must be told how to interact with an SLDS lest they treat the system as a human dialogue partner. Users, of course, should react appropriately to the system's utterances during dialogue (iii), given their scenarios and the dialogue context. The need to in-

S1: Do you want anything else?
U1: Yes.
S2: Customer number 4. Is that correct?
U2: No, customer number 3.

S3: Customer number 10 does not exist. Please state your customer number.
U3: Customer number 3.
S4: Customer number 3.
U (overlapping speech): Yes.
S4 (continued): How many people will travel?
U4: Three people.
S5: Three people. Please state the id-number of the first person.
U5: Number 6.
S6: Please state the id-number of the second person.
U6: Number 3.
S7: Please state the id-number of the third person.
U7: Number 4.
S8: Id-numbers 6, 3 and 4, Anders Bækgaard, Paul Dalsgaard and Børge Lindberg. Where does the journey start?
U8: The journey starts in Aalborg.
S9: From Aalborg. Where does the journey end?
U9: In Copenhagen.
S10: In Copenhagen. Do you want a return ticket?
U10: Yes.
S11: Are you particularly interested in discount?
U11: Yes.
S12: On which date will the journey start?
U12: The first weekend in February.
S13: Friday February 10th. At which time?
U13: It must be on Saturday at 7:20. **E2, E4, E6a**
S14: There is no departure at 7:20. The closest other red departure is at 10:50. Do you want this departure?
U14: No.
S15: At which time?
U15: Saturday at 7:20. **E6a**
S16: There is no departure at 7:20. The closest other red departure is at 10:50. Do you want this departure?
U16: Yes.
S17: On which date will the return journey start?
U17: On Sunday February 5th.
S18: Sunday February 12th. At which time?
U18: 22:40. **E2**
S19: 22:40. You have now booked 3 tickets for Anders Bækgaard, Paul Dalsgaard and Børge Lindberg from Aalborg to Copenhagen on Friday February 10th at 10:50 returning on Sunday February 12th at 22:40. The tickets are on green conditions. Your reference number is 51. The total price is 2820 kroner. Are the tickets to be mailed or will the travellers pick them up at the airport?
U19: They should be mailed. **E2**
S20: The tickets will be mailed about one week before the departure. Do you want anything else?
U20: Yes.

Figure 3. The dialogue T32a from the user test of the Danish dialogue system (translated from the Danish) [5]. The dialogue is based on the scenario shown in Figure 2. S means system, U means user. Utterances marked with an E and a number contain user errors and are discussed in more detail in Section 4.

clude (iv) follows from the important fact that it is impractical to explicitly inform users about all the types of behaviour theyshould avoid during dialogue with the system. Rather, users should be made to grasp the general fact that the system is a severely restricted dialogue partner which should be treated accordingly.

According to the above definition of 'user error', 102 individual user errors were found in the user test corpus which contains a total of 998 user utterances. A single utterance may contain several errors. Each user error was characterised with respect to its context, its nature was explained and, whenever possible, a preventive measure was proposed. The six user errors identified in the dialogue shown in Figure 3 are analysed in Figures 4 to 9. E(n) refers to the user error typology in Table 1. ER means error, EX means explanation, PM means preventive measure. S means system, U means user. The number succeeding each S and U refers to the dialogue in Figure 3. The dialogue was a transaction failure. The error which is considered the direct cause of the transaction failure is indicated by an italicised dialogue number.

ER: S12: On which date will the journey start? U12: The first weekend of February. S13: Friday February 10th. At which time? U: It must be Saturday at 7:20.
EX: The user ignores the date fed back by the system and only tries to change Friday into Saturday.
PM: People sometimes do not listen sufficiently carefully. They may also care less in experimental settings than in real life.

Figure 4. A user error identified in the dialogue shown in Figure 3. The error is of type E2: Ignoring clear system feedback. This error was considered a direct cause of the transaction failure.

ER: S17: On which date will the return journey start? U17: On Sunday February 5th. S18: Sunday February 12th. At which time? U18: 22:40.
EX: The user ignores the system feedback on date.
PM: People sometimes do not listen sufficiently carefully. They may also care less in experimental settings than in real life.

Figure 5. A user error identified in the dialogue shown in Figure 3. The error is of type E2: Ignoring clear system feedback. This error was considered a direct cause of the transaction failure.

ER: S19: You have now booked ... on Friday February 10th at 10:50 returning on Sunday February 12th at 22:40 ... at the airport? U19: They should be mailed.
EX: The user ignores the system feedback on date.
PM: People sometimes do not listen sufficiently carefully. They may also care less in experimental settings than in real life.

Figure 6. A user error identified in the dialogue shown in Figure 3. The error is of type E2: Ignoring clear system feedback. This error was considered a direct cause of the transaction failure.

ER: S13: Friday February 10th. At which time? U13: It must be Saturday at 7:20.
[*Continued on the next page.*]
EX: The user is too occupied with the present problem to remember to use 'change' when trying to change Friday into Saturday.
PM: 'Change' is not natural. Prefer mixed-initiative meta-communication.

Figure 7. A user error identified in the dialogue shown in Figure 3. The error is of type E4: Change through comments.

ER: S13: Friday February 10th. At which time? U13: It must be Saturday at 7:20.

EX: Natural user response package.
PM: Allow naturally related information, such as date and time, to be provided in the same user answer.

Figure 8. A user error identified in the dialogue shown in Figure 3. The error is of type E6: Answering several questions at a time.

ER: S: At which time? U: Saturday at 7:20.
EX: Natural user response package.
PM: Allow naturally related information, such as date and time, to be given in the same user answer.

Figure 9. A user error identified in the dialogue shown in Figure 3. The error is of type E6: Answering several questions at a time.

A more thorough analysis of the user errors revealed, however, that a significant number were caused by problems in the design of the system's dialogue contributions. For instance, users responded differently from what they should have responded according to the scenario because of missing system feedback or because a system question was too open and invited users to respond in ways which we had not intended. We shall ignore such cases and focus on the dialogue errors that were made solely by users. This leaves 61 individual user errors for discussion in what follows.

The remaining 61 user errors are of eight different types as shown in Table 1. Two error types (E3 and E6) were divided into sub-types. E1 includes the scenario violations, i.e. violations of explicit designer instructions. E2 and E3a include cases in which users did not pay attention to explicit system utterances (feedback and questions). E3b is closely related to E5 (see below). E3b, E4, E5, E6 and E7 represent violations of explicit system instructions provided in the system's introduction (Figure 1). In E8 the user violates implicit system instructions. We will now discuss each error type in more detail.

E1. Misunderstanding the scenario

As remarked earlier, scenario misunderstandings are artefacts of controlled user testing. Nevertheless, controlled user testing is important in systems design and it may be worth considering ways of preventing user errors in controlled test environments. It should be noted that scenario misunderstandings cannot give rise to transaction failure. Transaction failure occurs only when users do not obtain the reservation they actually ask for. In fact, scenario misunderstandings rarely lead to other forms of dialogue interaction problems. Users just carry out a different scenario. On the other hand, this may affect system evaluation. A scenario which is not carried out may result in that part of the dialogue model remains untested.

Table 1. The identified user error types and sub-types.

Error Types	Error Sub-Types	No. of Cases	Preventive Measure
E1. Misunderstanding of scenario	a. Careless reading or processing	14	Use clear scenarios, carefully studied, to reduce errors.
E2. Ignoring clear system feedback	a. Straight ignorance	7	Encourage user seriousness to reduce errors.
E3. Responding to a question different from the clear system question	a. Straight wrong response	4	Encourage user seriousness to reduce errors.
	b. Indirect response	3	Disguised dialogue design problem..
E4. Change through comments (including "false" keywords)	a. Cognitive overload	17	Disguised dialogue design problem..
E5. Asking questions	a. Asking for decision-relevant information	3	Disguised dialogue design problem..
E6. Answering several questions at a time	a. Natural response "package"	10	Disguised dialogue design problem.
	b. Slip	1	None.
E7. Thinking aloud	a. Natural thinking aloud	1	None.
E8. Non-cooperativity	a. Unnecessary complexity	1	None.

Almost one fourth of the 61 user errors were due to users acting against the instructions in the scenarios. These errors were of three (task-dependent) kinds: (a) users asked for one-way tickets instead of return tickets; (b) users were not interested in discount although according to the scenario they should be; and (c) users tended to miscalculate the date of departure if only given indirectly in the scenario. It seems likely that the main reason for the many scenario misunderstandings is the artificial experimental situation. People care less in an experiment than they do in real life and therefore tend not to prepare themselves sufficiently for the dialogue with the system. In addition, unclear scenarios cause errors. E1 thus raises two issues in the preparation of controlled user testing: (i) to reduce the number of errors, scenarios should be made as clear as possible. Nothing is gained by unclear or misleading scenarios. Clear scenarios should not be confused with *simple* scenarios. Scenarios should reflect the types of information real users actually have when addressing the system. This information may be complex and some scenarios should reflect that. This means that users may have to perform some mental processing of the scenario information in order to provide correct answers to the system's questions. (ii) Users should be encouraged to carefully prepare themselves on the scenarios they are to complete in conversation with the system. This should mirror the interest real users have in getting the system to deliver what they want.

Whatever preventive measures are taken, however, scenario misunderstandings are not likely to be totally absent from controlled user tests but reducing their number is an important goal.

E2. Ignoring clear system feedback

The speech recognition capabilities of most telephone-based systems are still fragile. It is therefore important that users listen carefully to the system's feedback to verify that they have been correctly understood. Of the seven transaction failures in the user test, one was caused by a combination of a dialogue design problem and a user who ignored clear system feedback. A second transaction failure occurred solely because the user did not pay sufficient attention to the system's feedback which made it clear that the user had been misunderstood (Figures 3, 4, 5 and 6). Three of the seven detected E2 cases occurred in this dialogue in which the user continuously ignored the system feedback on dates (Figures 4, 5 and 6). Thus, four out of the seven detected cases of ignored system feedback had severe implications for the success of the transaction. Moreover, had the user test included a real recogniser, more cases of system misunderstanding would have

occurred and hence more cases in which users would have had to identify system misrecognitions from the system's feedback.

E2 raises the issue of encouraging test subjects to "act" seriously in dialogue with the system and be very attentive to what the system says because recognition in SLDSs is much more error-prone than the hearing capabilities of normal humans. This would help reducing the number of user errors caused by their ignoring system feedback. Nothing is gained by having subjects who care too little about what is going on during the dialogue. Whatever preventive measures are taken, however, the problem of user inattentiveness is not likely to completely go away. This is true of both "artificial" user tests and real-life use of commercial systems.

The notion of a transaction failure that is caused by a "clean" user error may be controversial. It might be argued that transaction failures should be caused by systems design errors of one kind or another. On the other hand, it might be said that most user errors of ignoring clear system feedback only arise because the system has misunderstood the user in the first place.

E3. Responding to a question different from a clear system question

E3 has at least two sub-types. The first sub-type, E3a, includes four cases in which users gave a straight wrong response to a system question, for instance by answering "Saturday" to the question about departure airport. In one case the answer was not understood by the system and in three cases it was misunderstood. E3a raises the same issue as did E2 of encouraging users to seriously pay attention to the system's utterances. Similarly, E3a errors are not likely to go away completely, neither in "artificial" user tests nor in real-life interaction.

The second sub-type, E3b, concerns *indirect* user responses. For instance, a user answered "it must be cheap" to the question of hour of departure. In human-human conversation, indirect answers of this type would be perfectly all right. An indirect response indicates that the speaker does not possess the information necessary to provide a direct answer. In response to the indirect user answer quoted above, a human travel agent would list the relevant departures on which discount may be obtained. Our SLDS, however, has limited inferential capabilities and is not able to cope with indirect responses. They will be either not understood or misunderstood.

E3b is among the most challenging types of user errors in the test material. Indirect responses are natural to humans in situations in which they do not have sufficient information to produce a direct response. In such cases, we provide instead the information that we actually possess, leaving it to the interlocutor to infer the information asked for. We do this cooperatively, of course, only in cases in which the interlocutor can be assumed to have the information needed to perform the inference. The system, posing as a perfect domain expert, may legitimately be assumed to possess the required information. What the user overlooks, however, is that the system does not have the *capability to draw the proper inferences* from the user's information. The E3b cases therefore raise the hard issue of to which extent the dialogue designers should consider providing their system with the appropriate inferential skills. There does not currently appear to exist a principled answer to this problem. Furthermore, it may be argued that indirect user responses are not user errors at all. They do not conflict with the system's introduction (Figure 1). At best it might be argued that indirect responses conflict with the difficult requirement on users which we have called 'implicit instructions' to users (see above). If, however, we are right in the above interpretation of E3b-type user contributions, they are really oblique questions for information (see E5 below). We shall return to E3b in the concluding discussion.

E4. Change through comments

E4 gave rise to numerous (almost 30%) user errors in the test. In 16 out of 17 cases, users tried to make corrections through natural sentences rather than by using the keywords prescribed in the system's introduction (Figure 1). An example is shown in Figure 4. In none of these cases was the requested correction understood as intended. Only in one case did the user achieve the intended correction. The user used a keyword different from 'change' but meaning the same, which accidentally was recognised as 'change'. The theoretical importance of these findings is that of emphasising the undesirability of including designer-designed user keywords in dialogue design for SLDSs. Such keywords will neither correspond to the keywords preferred by all or most users nor to the natural preference among native speakers to reply in spoken sentence form rather than through keywords. It is furthermore our hypothesis that the more cognitive load a user has at a certain stage during dialogue task performance, the more likely it is that the user will ignore the system's instructions concerning the specific keywords to be used.

E4 raises the hard issue of allowing users a more natural form of repair meta-communication.

E5. Asking questions

E5 is among the most challenging types of user errors in the test material and is closely related to E3b (see above). Like the E3b cases, the E5 cases all occur when the system has asked for an hour of departure. For instance, a user then asks "what are the possibilities". What the observed cases show is that reservation dialogue, in its very nature, so to speak, is *informed reservation* dialogue. It is natural for users who are going to make a reservation or, more generally, order something, that they do not always possess the full information needed to decide what to do. In such cases, they ask for the information. Since the system poses as a perfect domain expert, this is legitimate. What users overlook, however, and despite what was said in the system's introduction (Figure 1), is that the system does not have the skills to process their questions. As with E3b above, it is not clear what the dialogue designer should do about this problem in the short term. Current systems are not likely to be able to understand all possible and relevant user questions in the context of reservation tasks. The optimistic conclusion is that E3b and E5 only constitute 4 user errors in total and that skilled users of the system will learn other ways of eliciting the system's knowledge about departure times. However, a principled solution to the problem only seems possible through enabling the system to conduct rather sophisticated mixed-initiative domain dialogue.

E6. Answering several questions at a time

E6 has at least two sub-types. The first sub-type, E6a, gave rise to many (about 16%) user errors in the test. Examples are a user who answers "the journey starts on Friday at 8:15" when asked for a

date of departure, and a user who answers “no, change” when asked if it is correct that the destination is Karup. Other examples are shown in Figures 8 and 9. In 7 of the 10 cases, only the part of the user’s response which answered the system’s question was understood. In the remaining 3 cases the entire user response was misunderstood. What this error type suggests is that (i) users naturally store information in “packages” consisting of several pieces of information. This means that they are unlikely to consistently split these packages into single pieces of information despite having been told to do so in the system’s introduction (Figure 1). Dialogue designers should be aware of the existence of such natural information packages and enable their system to understand them. (ii) Users have stereotypical linguistic response patterns, such as prefixing a ‘change’ keyword with a ‘no’. Dialogue designers should be aware of these natural stereotypes and enable the system to understand them. This problem appears solvable by today’s technology. Our SLDS is already able to accept such stereotypes in several cases, such as when information on departure and arrival airports is being provided in the same utterance. However, due to the present, strong limitations on active vocabulary we have not been able to allow natural information packages and stereotypes throughout the reservation dialogue.

The second sub-type, E6b, illustrates a phenomenon which no feat of dialogue design is likely to remove, i.e. the naturally occurring slips-of-the-tongue in spontaneous speech. Slips do not appear to constitute any major problem, however. Only one slip causing an interaction problem occurred in the entire corpus: when asked for the customer number, the user said “four, no sorry, change, change”. Only the number was recognised forcing the user to change it in the following utterance.

E7. Thinking aloud

E7 illustrates another phenomenon which no dialogue design effort is likely to remove, i.e. the naturally occurring thinking-aloud in spontaneous speech. Thinking-aloud does not appear to constitute a major problem, however. Only one case of natural thinking-aloud occurred in the entire corpus: when asked for the hour of departure, the user said “well, let me see, at 8:30 at the latest”.

E8. Non-cooperativity

E8 illustrates yet another phenomenon which cannot be removed through dialogue design, i.e. the deliberately non-cooperative user. Only one case of deliberate user non-cooperativity was detected in the test corpus. The user replied “the ticket should not be sent” to the system’s question of whether the ticket should be sent or would be picked up in the airport. This reply would not have been considered non-cooperative if produced in human-human conversation. However, the reply is unnecessarily complex and cannot be handled by our SLDS. We know that the particular user who caused the problem was deliberately testing the hypothesis that the system would be unable to handle the input because she said so in the telephone interview following her interaction with the system. SLDSs designers have no way of designing dialogues with sufficient robustness to withstand deliberately non-cooperative users. Nor should SLDSs designers attempt to do so, apart, of course, from ensuring that the system will not break down and that deliberately non-cooperative users cannot cause any harm. The simple fact is that deliberately non-cooperative users, when successful, will fail to get their task done.

5. CONCLUSION

The E1 errors are of only minor importance as they will disappear when the system is being used in real life. Furthermore, E1 errors cause no real dialogue interaction problems. Similarly, E8 errors are of minor importance because users will stop experimenting with the system when they want the task done. E6b and E7 can hardly be prevented but, at least according to our test material, they are infrequent and do not cause severe problems of interaction.

E2 and E3a seem to have a much larger effect on dialogue transaction success. Although they can hardly be completely avoided, it is likely that their number can be reduced by clearly making users aware of the importance of paying attention to system feedback and system questions. Real-life users are likely to be more attentive.

E3b, E4, E5 and E6a are the most challenging user error types. They would all be perfectly acceptable in human-human dialogue. However, because of the limited dialogue capabilities of our SLDS, it is clearly stated in the system’s introduction how users should interact with it in order to prevent these errors. Whereas E3b is less clear (Section 4 above), the E4, E5 and E6a errors all violate the system’s explicit instructions. The important question is why so many users violate exactly these instructions. A likely explanation is that, at least for many users, it is not *cognitively feasible* to follow the system’s explicit instructions. In an extreme example: had we asked users to always use exactly four words in their responses to the system’s questions, this would clearly have been cognitively infeasible. Similarly, several of the things which the system’s introduction asks users to do or avoid doing turn out to be unrealistic given the dialogue behaviour that is natural to most people. This reveals a fundamental shortcoming in our initial concept of user errors (Section 4). It is not sufficient to provide clear and explicit instructions to users on how to interact with the system. *It must also be possible for users, such as they are, to follow these instructions in practice.* The conclusion is that E3b, E4, E5 and E6a are *not* user errors at all but rather constitute more or less difficult problems of dialogue design.

E3b, E4, E5 and E6a are otherwise very different. E3b and E5 result from a mismatch between generic task type (ordering) and the type of dialogue initiative adopted for the application (system-directed domain communication). E4 and E6a belong to a much more general class of human-machine interaction problems. For years, in fact, experts on human error in the field of human factors have been aware of the broad category of errors illustrated by E4 and E6a. The reason why they are easy to overlook during design and until the user and field test data come in, is that, *in principle*, we can all avoid them. For instance, we can all easily say ‘change’ when we want to correct a system misunderstanding. During actual task performance, however, whether the task be one of driving a car or communicating with an SLDS, we tend to fall back on our natural skills and the human cognitive processing architecture, more or less ignoring rules or instructions that conflict with those skills and that architecture.

What are the implications of the findings reported in this paper? We emphatically do not want to argue that, because of problems such as E3b, E4, E5 and E6a, SLDSs of the same general type as ours cannot be used in realistic applications. None of these problems caused transaction failure. The E6a problems can be removed using current dialogue design techniques (Section 4). The E3b and E5 problems were few. And the E4 problems, of which

there were many, might at least be reduced in number through a larger active vocabulary.

ACKNOWLEDGEMENTS

The Danish dialogue system was developed in collaboration between the Center for PersonKommunikation at Aalborg University (speech recognition, grammar), the Centre for Language Technology, Copenhagen (grammar, parsing), and the Centre for Cognitive Science, Roskilde University (dialogue and application design and implementation, human-machine aspects). The project was supported by the Danish Research Councils for the Technical and the Natural Sciences. We gratefully acknowledge the support.

REFERENCES

[1] H. Aust, and M. Oerder, Dialogue Control in Automatic Inquiry Systems, *Proceedings of the ESCA Workshop on Spoken Dialogue Systems*, Vigsø, 121-124, 1995.

[2] N.O. Bernsen, H. Dybkjær and L. Dybkjær, Exploring the Limits of System-directed Dialogue. Dialogue Evaluation of the Danish Dialogue System, *Proceedings of Eurospeech '95*, Madrid, 1457-60, 1995.

[3] R. Cole, D.G. Novick, M. Fanty, P. Vermeulen, S. Sutton, D. Burnett and J. Schalkwyk, A Prototype Voice-Response Questionnaire for the US Census, *Proceedings of the ICSLP '94*, Yokohama, 683-686, 1994.

[4] H. Dybkjær, N.O. Bernsen and L. Dybkjær, Wizard-of-Oz and the Trade-off between Naturalness and Recogniser Constraints. *Proceedings of Eurospeech '93*, Berlin,. 947-50, 1993.

[5] L. Dybkjær, N.O. Bernsen and H. Dybkjær, Evaluation of Spoken Dialogues. User Test with a Simulated Speech Recogniser. *Report 9b from the Danish Project in Spoken Language Dialogue Systems*. Roskilde University, February 1996. 3 volumes of 18 pages, 265 pages, and 109 pages, respectively.

[6] L. Dybkjær, N.O. Bernsen and H. Dybkjær, Reducing Miscommunication in Spoken Human-Machine Dialogue. *Proceedings of AAAI ''96 Workshop on detecting repairing and preventing human-machine miscommunication*, Portland, 1996.

[7] W. Eckert, E. Nöth, H. Niemann and E. Schukat-Talamazzini, Real Users Behave Weird - Experiences Made Collecting Large Human-Machine-Dialog Corpora, *Proceedings of the ESCA Workshop on Spoken Dialogue Systems*, Vigsø, 193-196, 1995.

[8] N.M. Fraser and G.N. Gilbert, Simulating Speech Systems, *Computer Speech and Language 5*, 81-99, 1991.