Spoken Language Dialogue Systems

Report 6a, February 1994

Task-Oriented Spoken Human-Computer Dialogue

CPK - Center for PersonKommunikation, Aalborg University
CCS - Centre for Cognitive Science, Roskilde University

CST - Centre for Language Technology, Copenhagen

Printed at Roskilde University
February 1994
ISBN 87-7349-242-6

The present report is part of the documentation series from the strategic research programme *Spoken Language Dialogue Systems*. The programme is sponsored by the Danish Technical Research Council. The project partners are the Center for PersonKommunikation (CPK) (earlier Speech Technology Centre (STC)) (coordinating partner) at Aalborg University, the Centre for Cognitive Science (CCS) at Roskilde University, and the Centre for Language Technology (CST), Copenhagen.

Authors:

Niels Ole Bernsen
Laila Dybkjær
Hans Dybkjær

The project partners can be contacted at:

Center for PersonKommunikation (CPK):

Paul Dalsgaard
Aalborg University
Frederik Bajers Vej 7
DK-9220 Aalborg Ø, Denmark
Phone: +45 98 15 85 22. Fax: +45 98 15 15 83
Email: pd@cpk.auc.dk

Centre for Cognitive Science (CCS):

Niels Ole Bernsen
Roskilde University
P.O.Box 260
DK-4000 Roskilde, Denmark
Phone: +45 46 75 77 11. Fax: +45 46 75 45 02
Email: nob@cog.ruc.dk

Centre for Language Technology (CST):

Bente Maegaard
Njalsgade 80

DK-2300 Copenhagen S, Denmark
Phone: +45 35 32 90 90. Fax: +45 35 32 90 89
Email: bente@cst.ku.dk

Preface

This report is the sixth in the documentation series from the research programme *Spoken Language Dialogue Systems.* The objective of the programme is through the development of prototypes to gain new knowledge in the research fields of speech technology, natural language processing and human-computer interaction (cognitive engineering) and especially in the combination of these fields. The programme is scheduled to run for a four year period (primo 1991 - primo 1995) and is divided into the two sub-projects P1 and P2, each scheduled to produce a running prototype in the domain of flight ticket reservation and information. The prototype P1 has now been implemented and is currently being tested.

Report 6 has two parts, 6a and 6b [Dybkjær and Dybkjær 1994]. The present report (6a) focuses on spoken dialogue in terms of input from and output to the user. Whereas P1 has a highly system-directed dialogue, the aim is to create the basis for developing a less system-directed dialogue in P2 and beyond. Implementational issues and the interaction with the domain are presented in Report 6b.

Chapter 1 below provides an introduction to task-oriented human-computer dialogue theory including a number of central concepts. Chapter 2 provides a comprehensive overview of the design problems which were actually addressed in order to *prevent* the occurrence of problems of understanding between user and system. The chapter goes beyond P1 by introducing new possible design commitments which were not taken into account during P1 development but which would seem relevant to the design of less rigid and less system-directed dialogue models. Chapter 3 addresses the design of a satisfactory way of *resolving* the problems of understanding between user and system which inevitably occur during communication. Chapter 4 concludes by summarising central points, providing a comparison with two related dialogue systems, and briefly mentioning future work. Appendix A provides a structured representation of the design decisions which are referred to throughout Chapter 2. Appendix B briefly compares Grice's cooperativity principle with the more detailed principles of cooperativity identified during the development of the P1 prototype.

Acknowledgements

We would like to thank Tom Brøndsted and Lars Bo Larsen at CPK and Bradley Music, Lene Offersgaard and Claus Povlsen at CST for their helpful comments and suggestions on a draft version of the report.

Keywords

Task-oriented dialogue, dialogue theory, spoken human-computer dialogue.

Danish Summary

Denne rapport er den sjette i dokumentationsserien fra forskningsprojektet *Behandling af talt sprog i dialogstyrede applikationer*. Projektets formål er gennem udvikling af prototyper at erhverve ny viden inden for forskningsområderne taleteknologi, natursprogsbehandling og menneske-maskine interaktion (kognitionsforskning) og specielt i kombinationen af disse discipliner. Programmet løber over en fireårig periode (fra primo 1991 til primo 1995) og er opdelt i to dele, P1 og P2, der hver skal resultere i en kørende prototype inden for domænet flybilletbestilling og -information. Prototypen P1 er på nuværende tidspunkt færdigimplementeret og under afprøvning.

Rapport 6 består af to dele, 6a og 6b [Dybkjær and Dybkjær 1994]. Nærværende rapport (6a) fokuserer på talte dialoger i form af input til og output fra brugeren. Hvor P1 har en meget systemstyret dialog, er det målet at skabe baggrund for udviklingen af mindre systemstyrede dialoger i P2 og frem. Implementation og interaktion med domænedatabasen omtales i Rapport 6b.

Kapitel 1 nedenfor giver en introduktion til opgaveorienteret menneske-maskine dialogteori inklusive en række centrale begreber. Kapitel 2 giver et overblik over de design-problemer, der faktisk blev taget hånd om for at forebygge forståelsesproblemer mellem bruger og system. Kapitlet går ud over P1 ved at introducere og begrunde nye mulige designbeslutninger, som taget under udviklingen af P1, men som synes relevante for design af mindre stive og mindre systemstyrede dialogmodeller. Kapitel 3 omhandler hvordan man på tilfredsstillende vis løser de forståelsesproblemer mellem bruger og system, der uvægerligt opstår under en samtale. Kapitel 4 afrunder med at opsummere centrale punkter, give en sammmenligning med to andre dialogsystemer samt en kort oversigt over fremtidigt arbejde. Appendix A giver en struktureret repræsentation af de designbeslutninger, der refereres til i kapitel 2. Appendix B sammenligner Grices kooperativitetsprincip med de mere detaljerede principper for kooperativitet, der blev identificeret under udviklingen af P1.

Contents

1 Task-Oriented Human-Computer Dialogue

Restricted natural language is increasingly being used in the communication with computer systems. This raises the need for a dialogue theory which can support the design of restricted spoken and written natural language dialogues that are usable and reasonable seen from the user's point of view and which can be managed by the system.

This report focuses on restricted spoken language dialogues. Such dialogues are more difficult to handle than written input partly because of the current state of speech technology R&D (Research and Development), and partly because spoken natural language dialogue is spontaneous which implies the need to handle a number of complex phonological and linguistic phenomena which are rare or non-existent in ordinary written language and which, moreover, complicate recognition.

The background of the report is the development and implementation of a prototype, P1, of a spoken language dialogue system in the domain of Danish domestic airline ticket reservation and flight information accessed through the telephone. A more advanced version, P2, of the first prototype will be be developed in due course.

Figure 1.1 shows the typical flow structure in a spoken language dialogue system.

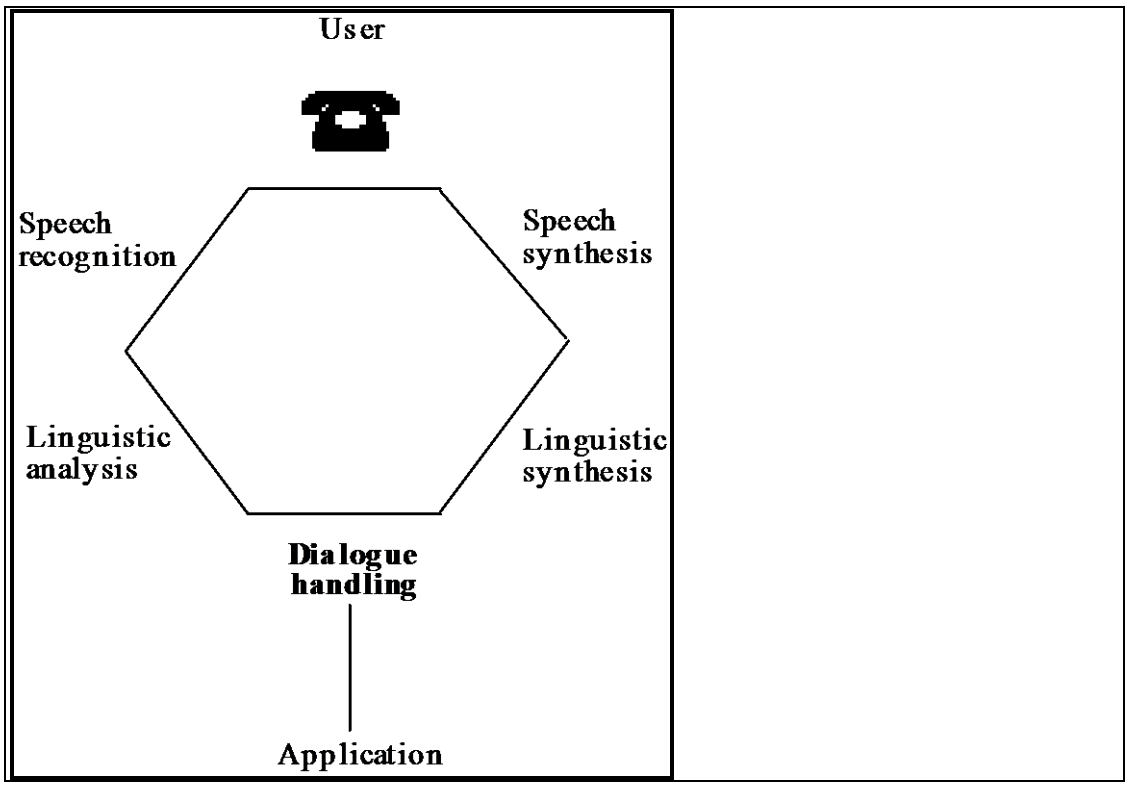

Figure 1.1: Typical logical structure of a spoken language dialogue system.

P1 takes as input a speech signal which is recognised and passed as a sentence hypothesis to the linguistic analysis module. This module makes a syntactic and semantic analysis of the sentence and represents the result in a set of frame-like structures called semantic objects. The dialogue handling module interprets the contents of the semantic objects received from the linguistic analysis module and takes action according to this input, e.g. through updates or queries to the application database or decisions on the next output to the user which is then being synthesized. In P1 the output module uses pre-recorded speech and hence does not involve linguistic synthesis and speech synthesis. A description of the architecture of P1 is provided in [Larsen et al. 1993].

P2 is planned to have more advanced output functionality based on linguistic synthesis and speech synthesis. Furthermore, improved recognition techniques, an improved parser and extended grammars and vocabulary are assumed to allow the design of more advanced and flexible dialogues than in P1.

The remainder of this chapter presents a framework for task-oriented spoken language dialogue systems development. Although primarily based on P1, the framework is intended to have a broader scope so as to enable the definition of a number of further steps to be taken towards the achievement of truly habitable systems.

1.1 The need for task-oriented dialogue theory

Most spoken or written language dialogue theory has so far dealt with unrestricted human-human dialogue. While no single, unified dialogue theory has yet appeared from the various frameworks and approaches which have been proposed in the literature, parts of these theories and the aspects of dialogue they cover are potentially relevant to the narrower purpose of establishing a task-oriented human-computer dialogue theory. This is true of the Gricean theory of cooperativity in dialogue [Grice 1975] (cf. Appendix B below), speech act theory [Searle 1969], discourse representation theory [Kamp 1981], plan-based approaches to dialogue [Litman 1985], Grosz and Sidner's intentional approach [Grosz and Sidner 1986, 1989], relevance theory [Sperber and Wilson 1986] and rhetorical structure theory [Mann and Thompson 1987, 1988], among others. However, before importing insights and notions from such theories it seems important to develop the basics of a spoken language dialogue theory which reflects the state-of-the-art of spoken human-computer dialogue. The latter theory should provide the proper perspective from which to relate to other dialogue theories in order to clearly define the theoretical and research priorities at this stage. A related approach is taken by [Bilange 1991].

Current spoken language dialogue systems using speaker-independent spontaneous speech do not allow unrestricted dialogue as they are characterised by a long list of limitations which are not present in unrestricted human-human dialogue [Dybkjær et al. 1993]. The list includes:

- limited speech recognition capability;
- real-time response constraints;
- limited vocabulary;
- limited grammatical coverage;

- limited capability of handling the grammar used in spontaneous speech as well as other characteristics of spontaneous speech such as hesitations, false starts, non-words and overlapping speech (of system and user);
- limited semantic expressiveness;
- limited capability of handling discourse phenomena such as anaphora and ellipses;
- limited discourse structure comprehension including discourse focus;
- limited language generation capability;
- limited speech generation capability which excludes phenomena such as stress and intonation.

Due to limitations such as the above, spoken language dialogue systems design cannot simply transfer results from dialogue theories which deal with unrestricted human-human dialogue. Instead, there is a need to define the level of dialogue which current spoken language dialogue systems are capable of in order to:

- precisely characterise each individual system including its limitations;
- precisely characterise similarities and differences between current systems;
- define the needs for research and technological development which might help to incrementally improve the capability of current spoken language dialogue systems; and
- facilitate the transfer of relevant results from human-human dialogue theories.

The level of dialogue which current spoken language dialogue systems are capable of is perhaps primarily characterised by being *task-oriented* [Smith 1991]. Spoken language dialogue systems enable users to accomplish a limited number of precisely circumscribed tasks. Most existing human-human dialogue theory does not clearly focus on task-oriented dialogue which is the only category of dialogues that can be handled by current spoken language dialogue systems. Together with meta-communication (see below), task-orientation is what enables current systems to successfully undertake spoken dialogue with humans despite the many limitations which characterise such systems by comparison with human interlocutors.

The following sections cover the elements of a principled, task-oriented human-computer dialogue theory which appear necessary to characterising spoken language dialogue systems such as P1. Some of the elements are only rudimentarily present in P1 but will be needed in more advanced systems such as P2. It is envisaged that more elements will have to be added to the theory to handle still more advanced systems when additional tasks become included and dialogue naturalness increases. The presentation links up with and introduces the detailed presentation of P1's dialogue structure in Report 6b. The theoretical terminology adopted is deliberately 'conservative' in the sense that its origins in HCI task analysis have been explicitly preserved instead of having been transformed into the terminology of some theory of human-human dialogue (contrast, e.g. the SUNDIAL task-oriented dialogue theory in [Bilange 1991]).

1.2 Dialogue level decomposition

The decomposition of task-oriented human-computer dialogue involves at least the following three levels of description and analysis each of which is illustrated by examples from P1:

1. Basically, task-oriented human-computer dialogue is performed in order for a user to complete some task through interaction with a computer system. A *dialogue task* N may consist of one or more other tasks which are referred to as *dialogue sub-tasks* relative to N. Tasks may be embedded in, and hence be sub-tasks relative to, other tasks. Task N is realised through realising its dialogue sub-tasks a, b, c, etc. which again are realised through realising their sub-tasks, etc.

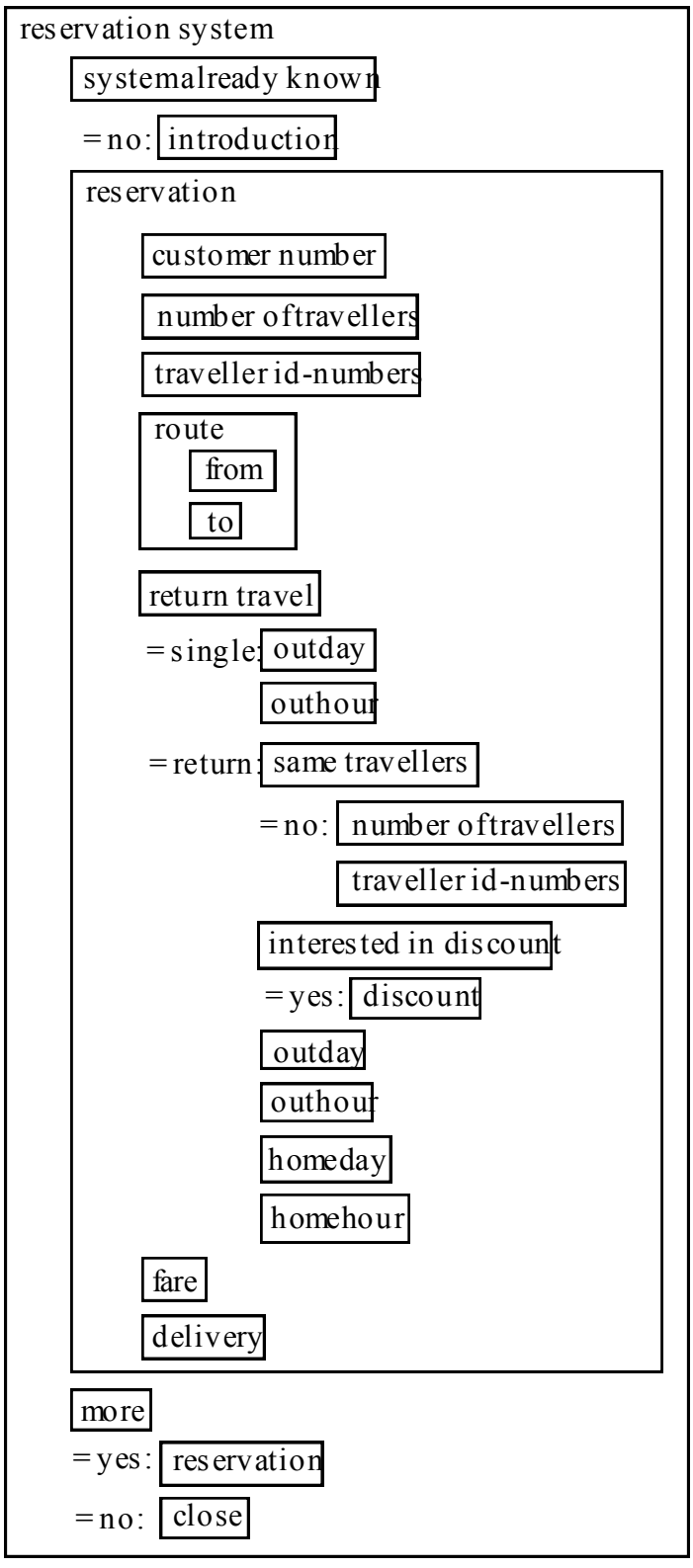

Figure 1.2: The unfolded dialogue task structure for the implemented P1. Only tasks concerning domain communication as distinct from meta-communication (cf. Section 1.8) are shown. A labeled box indicates a task. If a box A contains another box B then B is a sub-task relative to A. At some points during dialogue the structure to follow depends on the user's answer to the most recent question.

In such cases an answer is indicated as '= [answer]:' followed by the tasks to be performed in this case. In general the dialogue task structure is a cyclic graph with conditional branches.

The global unfolded dialogue task structure will show all tasks and their embeddings, i.e. which tasks are subtasks relative to a given task. The unfolded dialogue task structure for the implemented P1 is shown schematically in Figure 1.2. In principle, the task structure level is independent of the input/output modalities through which user and system have to interactively complete the tasks. P1 is a unimodal input/output system which uses spoken natural language for both input and output.

2. Task N is realised through user-system turn-taking involving the *dialogue turns* S (System)1, U (User)1, S2, U2, S3, etc. (see also Chapter 3). A turn consists of a user or system *utterance*. Most tasks require turn-taking. The only two tasks in P1 which do not require turn-taking are (1) the closing of the dialogue (close task) which is done by the system and in which user response is inconsequential and (2) the computation of the total price of the reserved ticket(s) (fare task) which is internal to the system. The computed price is stated in the confirmation of the reserved ticket(s) (cf. Section 1.5). Each turn can at least be characterised by the dialogue act(s) it contains and by whether the speaker (user or system) has the initiative or responds to an initiative taken by the interlocutor (see below and Sect. 1.4). At any given point during dialogue, either the system or the user has the initiative. For reasons to be explained shortly, P1 does not explicitly mark who among the interlocutors has the initiative and who responds. The following example shows the successful completion of three sub-tasks during six dialogue turns:

S1: How many persons are going to travel?

U1: One.

S2: What is the id-number of the person?

U2: Fifty-seven.

S3: Id-number fifty-seven Jens Hansen. Where does the travel start?

U3: In Aalborg.

3. A turn may contain one or more *dialogue acts*. A dialogue act is similar to a speech act [Searle 1969]. However, distinctions more fine-grained than those of Speech Acts Theory may have to be made during later developments of the basic dialogue theory presented here. In the example above, the system's third turn S3 contains two dialogue acts, the first being a statement which provides feedback on the dialogue act in the preceeding user turn U2, the second being a question to the user.

1.3 Task structure and expertise

Tasks have more or less structure, i.e. consist of a more or less ordered set of sub-tasks. Many tasks, such as the flight ticket reservation task of P1, have a stereotypical structure which, if known, shared and followed by the dialogue partners will lead to successful completion of the dialogue task. A *task stereotype* prescribes which information needs to be exchanged between the dialogue partners to complete the task and, possibly, roughly in which order this may be done naturally. The dialogue task structure in Figure 1.2 expresses

the reservation task stereotype. This structure conforms to the most common structure found in corresponding human-human reservation task dialogues recorded in a travel agency [Dybkjær and Dybkjær 1993].

Shared task stereotypes strongly facilitate dialogue systems design because they allow the computer to direct the dialogue through asking questions of the user without the user feeling this to be a major drawback of the design. In P1, users are instructed to answer the system's questions, as determined by the task stereotype and the immediately preceeding user input, one at a time and thus to perform only one dialogue act per turn. However, in some cases it is actually possible for the user to provide more than one piece of information per turn. Whether this is possible or not depends on the system focus at a certain point of the dialogue, i.e. on the scope of the predictions made by the system (see Section 1.6 below). The testing of P1 will show the extent to which the imposed restriction on the number of user dialogue acts per turn leads to user problems. It is quite possible that users will sometimes ignore the system's instruction.

Users who are familiar with the task stereotype are task experts whereas users who are not familiar with the task stereotype are task novices. In a task such as the reservation task of P1, most adult users can be expected to be task experts in some sense. In practice, task expertise is a matter of degree. Relative to P1, we may nonetheless roughly distinguish between the following levels of expertise:

- *top experts* are users who have expertise both in the stereotypical structure of the task and in how the particular system (i.e. P1) handles the task through spoken dialogue;

- *ordinary experts* are users who more or less know the stereotypical structure of the task and the meaning of the large majority of terms used by the system but are not familiar with the way the system handles the task through spoken dialogue;

- *novices* are users who know neither the system nor the main components of the task nor the precise meaning of all the terms used by the system to communicate task requirements to users.

In the application domain of P1, the relevant user population fortunately consists of top experts and ordinary experts. There is therefore no need to explain the basic task structure. The few task concepts which cannot be assumed to be familiar to users, such as, e.g., 'red' and 'green' discounts, must be explained by the system. There is a potential need to adjust system behaviour to the two different expert user groups. This has so far been done only in the case of the introduction to the system which can be circumvented by the top experts (cf. Figure 1.2). It remains an open question if additional adjustments to the two user groups are needed and what they might be.

The distinction between tasks which have stereotypical structure and tasks which do not have stereotypical structure is potentially important in systems design. P1 has given rise to the following hypothesis: *If a task has a stereotypical structure which is known to the user population, then the system may take and preserve the initiative during dialogue without any unacceptable loss of naturalness in dialogue.* If true, this hypothesis strongly facilitates spoken language dialogue systems design for stereotypical tasks.

The information task which is a candidate for implementation in P2, on the other hand, appears to be unstructured. The information task includes a wealth of information on time

tables, travel conditions, reduced fares, etc. However, by contrast with reservation task completion which is an all-or-nothing matter, in the information task a user would normally only need information on at most a couple of specific items. If the information task is structured in the same way as the reservation task, i.e. as a complex tree-structure, users may find that the information they need lies buried deep in the system and that they have to go through a tedious sequence of system questions before arriving at the particular bit of information they want. The only way to avoid this, it seems, is to allow user initiative in explaining the information needed by a particular user. The system should respond by immediately identifying the relevant piece of information. In other words, the hypothesis is that *if a task has no stereotypical structure and contains a large number of optional sub-tasks, then the system cannot take and preserve the initiative during dialogue without significant loss of dialogue naturalness.*

1.4 Dialogue initiative

The interlocutor who controls the dialogue at a certain point has the *initiative* at this point and may decide what to talk about next, e.g. by asking questions which the dialogue partner is expected to answer. As only the reservation task has been implemented in P1 and as this task has a stereotypical structure, it seems acceptable that the system, with two exceptions to be mentioned shortly, takes and preserves the initiative throughout the dialogue. The distinction between user and system initiative has not been explicitly marked in the implementation of P1. The system takes and preserves the initiative by concluding all its turns (except when terminating the dialogue) by a question to the user. The system questions serve to implicitly indicate that initiative belongs to the system rather than to the user.

The user can take the initiative at any time during dialogue by uttering one of the two following meta-communicative (see Section 1.8) commands: *repeat* and *correct*. The *repeat* command makes the system repeat its latest utterance. The *correct* command makes the system backtrack to its latest question to the user. Use of these commands ensures that the system has no difficulty in determining (a) when the user takes the initiative and (b) which task the user intends to perform in doing so. The down-side of this, of course, is that the user can perform these tasks only by using those commands and that this is the only way users can take the initiative during a dialogue. The latter point, however, is of secondary importance as long as we are dealing only with well-structured (stereotypical) tasks, as discussed above. The former point shows that P1 is, in effect, a command/keyword-based system as far as user initiative is concerned. In natural dialogue, initiative can be taken, given or negotiated. In P1, the system has the initiative; the user can only take the initiative by using specific keywords; and there is no room for giving or negotiating initiative. Once the latter becomes possible, the system will probably need an explicit representation of who has the initiative at any point during dialogue.

1.5 Evaluation

The system performs evaluation of each piece of user input. User input is checked with the domain database and/or already provided information, and information retrieved from the database or provided earlier but to be used now is checked with the user (cf. Report 6b).

The provision of sufficient *feedback* to users on the actions and commitments they have made in communicating with the system is a crucial component in interface design. Such feedback allows users to evaluate if the system has complied with their plans for interacting with it and whether dialogue repair or clarification is needed (see Section 1.8). P1 provides feedback on the user commitments made during a task. When the reservation task is closed, the system will summarise the information provided by the user by announcing that a reservation has been made for so many identified persons and for a certain route, date and hour. If a return ticket has been booked, the return date and hour are provided and it is stated if the ticket is on discount along with the total price of the reserved ticket(s). However, as the user has no means of making corrections or cancelling the reservation at this final stage of the dialogue, P1 also provides feedback when closing each sub-task. This is done by repeating the key information provided by the user, such as the names of the departure and arrival airports or the date or the hour of departure. Users who accept the feedback contents do not have to reconfirm their commitment as the system will immediately carry on with the next sub-task. If, on the other hand, the user wants to make a change at this point, use of the *correct* command is required.

In P1 a *correct* command will always cause the system to check with the user the piece of information provided as an answer to the system's previous question, and the system will allow the user to enter a new piece of information to replace the incorrect one. By the time the system confirms the entire reservation its preceeding question was to obtain the hour of departure. When the user has provided this information, the system confirms it and then confirms the entire reservation and asks how the ticket(s) should be delivered. If the user responds to this question by saying *correct* it is only the departure hour which can be corrected. A solution to this problem could be to turn the confirmation of the reservation as a whole into a separate task in which the user is asked to confirm the reservation as such. The confirmation of the entire reservation might be kept where it is now or be postponed until after the delivery of the ticket has been agreed upon, and hence be used to conclude the reservation task. If the user states that something is wrong with the reservation, the system should go through all of the information provided by the user and check each piece of information interactively. The implementation of P1 is prepared for doing this which, however, was not present in the specification resulting from the Wizard of Oz experiments (cf. Chapter 2).

1.6 Predictions, system focus and communication levels

A critical component of current spoken language dialogue systems is the speech recogniser with its limited recognition capabilities. The speech recogniser needs all the constraining information obtainable in order to improve recognition accuracy. The dialogue handling module can produce such constraining information in the form of predictions on the next possible user input. Predictions are expectations as to what the user will say next and thus specify the sub-vocabulary and sub-grammars to be used. Predictions constrain the search space and act as user allowances, i.e. determine the sub-tasks which the user is allowed to address in the next utterance. If the user chooses to address other sub-tasks, system understanding will fail. In P1, predictions are also sent to the linguistic analysis module to provide support for disambiguation of user input. The more stereotypical structure a task has, the easier it is to make good predictions provided the user is cooperative.

Predictions are based on the set of sub-tasks currently in system focus and on the level of communication. The set of sub-tasks in *system focus* are the tasks which the user is allowed to refer to in the next utterance. A useful heuristics seems to be that the set of sub-tasks in system focus always include the preceeding sub-task, the current sub-task and the possible succeeding sub-task(s) according to the default dialogue task structure. Ideally, the system focus should correspond to the *common dialogue focus* shared among the interlocutors. The heuristics just mentioned should make this correpondence achievable in many types of task-oriented dialogue based on task stereotypes, provided that suitably 'soft' technological constraints obtain. In such cases, the heuristics may ensure correspondence between system focus and the set of sub-tasks in user focus, i.e. the set of sub-tasks which the user will find it natural to talk about at a given point of dialogue. In general, of course, the more overlap there is between system focus and user focus, the more likely it is that the dialogue will proceed smoothly.

In addition to task structure, the *level of communication* is another important factor in constraining the system's focus and hence its predictions on sub-vocabulary and sub-grammar. The level of communication determines the way in which the user is allowed to express information, i.e. whether the user is allowed to express information freely or in various, more constrained ways. Co-determining the system focus, the level of communication also influences how output to the user is expressed. If, e.g., the level is spelling, the user will be asked to spell the information to be provided. We have found it appropriate to distinguish between the five levels listed in Figure 1.3 (cf. also Chapter 3). As can be seen the full set of sub-tasks in system focus is only used in the system's predictions at the open level. In all other cases the system's predictions will in principle be more limited, i.e. they will only refer to information on one sub-task.

Level	**Predictions as influenced by level**
1. spell	the answer is spelled
2. yes/no	yes or no
3. multiple choice	list of acceptable values
4. focused	the current sub-task
5. open	the set of sub-tasks in system focus

Figure 1.3: Levels of communication and their influence on predictions.

The approach to system focus expressed in Figure 1.3 is not uncontroversial. It is reasonable to delimit predictions according to level as shown in Figure 1.3 if the system has problems in understanding the user (cf. Section 3.1). But sometimes the most natural way of expressing, e.g., a question will not correspond to the actual level. Therefore, if there are no problems of understanding and the system has expressed itself at a lower level than the actual one, it may be an advantage to use the whole set of sub-tasks in system focus in the system's predictions even when the level is not open, because this makes it possible to recognise user utterances which are formulated differently from what is expected at a given level.

Levels of communication play a major role in *graceful degradation* which is a method for solving the system's understanding problems by using clarification dialogue (cf. Section 1.8 on meta-communication). For a more detailed description of graceful degradation we refer to Section 3.1.

In P1, the dialogue handler predicts the next possible user utterances and informs the speech recogniser and the parser to download the relevant sub-vocabulary and sub-grammars. To obtain both real-time performance and acceptable recognition accuracy in P1 it has been necessary to restrict sub-vocabularies to contain at most 100 words. User utterances which are out of system focus and hence are not being predicted, will not be correctly understood. Other systems might be able to discover that an utterance is not within the predictions (e.g. if it only matches the garbage models in the recogniser) and then perhaps resort to a relaxed set of predictions. In this way the user input may be recognised after all, thus saving extra turns but at the expense of slower processing times.

In P1 the user is asked to provide one piece of information per turn. The system's predictions include the current sub-task plus the possibilities of the user saying 'correct' or 'repeat'. In a few cases the system's predictions include more than the current sub-task. For instance, when the system expects an arrival airport, the departure airport is also included in its predictions and may therefore be provided by the user in the same turn as the arrival airport. This was done because information on arrival airport and departure airport are often provided in a single utterance by users.

Information on the sub-tasks in system focus is hardwired in P1 as is the information on communication level and how the system should express itself. Note that P1 does not have the spelling level and only asks one question ("Do you want more?") which may be classified as open, see also Chapter 3. For each point in the dialogue structure it has been decided which sub-grammars should be active, i.e. the sub-tasks in system focus, and how the system's utterances should be expressed. The decision on sub-grammars depends on the number of active words required. In P2 it would be desirable to have a dynamically determined set of sub-tasks in system focus. In fact this is so in all cases where the focus set depends on the actual context. For example, when part of the initiative is left to the user it is expected that deviations from the default task structure may occur from time to time and in these situations it is desirable that the system is able to determine the set of sub-tasks in system focus at run-time. Full flexibility of predictions will require the possibility of combining grammars to make it possible to parse more complex utterances. In P1 sub-grammars are disjoint graphs without a common root node [Povlsen and Music 1994], which means that a sentence with two parts each obeying a sub-grammar in the current sub-grammar set will not be accepted because there is no way of coming from the first branch of the sub-grammar to the second one.

In unstructured tasks such as the information task envisioned for P2, it will be much more difficult to make good predictions unless the system preserves the initiative throughout, which has the negative consequences on usability noted in Section 1.3 above. This would seem to imply that, if complex unstructured tasks are to be handled naturally by the system, there is no way around providing the system with a capability to initially identify the sub-task a user intends to perform based on the user's utterance (see Section 1.9).

1.7 Dialogue history

A dialogue history is a log of the information exchanged so far in the dialogue. In P1 every piece of information obtained from the user and the domain data-base and the extent to which this information has been checked by the system is recorded. However, there is no log

of the *order* in which the information was obtained so in this sense P1's dialogue history is rudimentary. P1 only logs:

1. an idenfication of the previous sub-task in order to be able to make corrections;
2. the logical contents of the latest system question. It is important to the interpretation of a yes/no answer from the user to know how the question was formulated, e.g. it makes a difference if the question was "Is it a one-way travel?" instead of "Is it a return travel?";
3. the semantic contents of the user's latest utterance.

The testing of P1 will show the effectiveness of its rudimentary dialogue history. With increased user initiative, a dialogue history logging more of the user and system utterances will be necessary to allow, i.a, returning to non-completed sub-tasks, correction of more than the most recent user input or improved anaphora resolution. Thus the system may have to suspend the current task if it discovers that it needs some value in order to proceed, which can only be obtained by performing a task which is prior in terms of the task structure. For instance, to determine whether a certain departure hour is acceptable it is necessary to know the date of departure.

1.8 Meta-communication

Given a task-oriented approach to dialogue theory, there is a relatively clear distinction between two types of dialogue between user and system. The first is the basic, task-oriented dialogue described above, or the task-communication. The second is the meta-communication between user and system which has an important auxiliary role in both oral human-human and human-machine dialogue. Meta-communication or 'communication about the communication' serves as a means of resolving misunderstandings and lacks in understanding between the dialogue partners during task-oriented dialogue. In current spoken language dialogue systems, meta-communication for *dialogue repair* is essential because of the sub-optimal quality of the systems' recognition of spontaneous spoken language. Similarly, meta-communication for *dialogue clarification* is quite common in human-human dialogue and the ability to carry out clarification dialogues is generally desirable in spoken language dialogue systems.

In P1, meta-communication can be initiated by the user through the *repeat* and *correct* commands (cf. Section 1.4). The lack of naturalness in user meta-communication as currently implemented has been pointed out (cf. Section 1.4). The *correct* command initiates repair meta-communication. The system may also initiate repair meta-communication. This happens when it has failed to understand the last user utterance, so that the semantic representation sent as input to the dialogue handling module from the linguistic analysis module is empty. In this case the system will tell the user that it did not understand what was said. The testing of P1 will show how efficient and natural the system actually is in handling the situations in which it has failed to understand the user.

The *repeat* command is the only means users have to initiate clarification meta-communication. Thus the user cannot successfully ask, e.g., "What do you mean by the term '...'?" or "What is '...'?". However, much effort has gone into the design of individual system utterances in order to minimise the users' needs for clarification of system utterances (see

Chapter 2). The P1 system also has a limited ability to initiate clarification sub-dialogues. If the user did not provide the information asked for but provided a different piece of information which was understood by the system, that information will be stored and the system will ask again for the information which was not provided. Thus, for example, the system may ask for the departure airport and the user may tell the arrival airport. In this case the arrival airport is stored and the system asks again for the departure airport. In another example, if a piece of information has already been provided but the system is not quite sure that it is the correct one, it will check with the user. If the user, e.g., provides the departure and arrival airports in the same turn but was only asked for the departure airport, the system may check the arrival airport with the user instead of asking for it. Checking is done by using an interrogative voice in a statement such as "The arrival airport is Aalborg?". The testing of P1 will show just how efficient and natural the system is in handling clarification sub-dialogues. Chapters 2 and 3 below discuss a number of additional meta-communicative dialogue acts which might be needed in P2.

Task-oriented communication and auxiliary meta-communication are both controlled by the dialogue handler. Separate and independent representation of the two types of communication by two different modules might be preferable. Task communication depends on the domain and the dialogue model. The function of meta-communication, on the other hand, is generally independent of task and domain. Ideally, then, a meta-communication function might be transferred between applications which are different in task and/or domain without itself having to change from one application to another [Bilange 1991]. There are limitations to the possibilities of transfer, however. Thus the help function cannot be transferred between applications without change in phrases because these are domain dependent, although the basic structure of the function may be preserved. Similarly, meta-communication functions can of course not be immediately transferred between applications having different levels of communication.

1.9 Summary of needs for extending the dialogue capacity of P1

Let us assume that P2 will allow users to carry out at least a second major task in addition to that of reservation and that this second task is an unstructured one such as the information task. As we have seen (Section 1.4), the natural handling of unstructured tasks by the system requires that the user is given the initiative to initially explain which specific sub-task the user wants to accomplish. This requirement imposes four important demands on the design of P2 which will be discussed below.

Firstly, the system will have to be able to *identify which task the user intends to perform,* based on the user's utterances. P1, which only addresses one highly structured task (i.e. reservation) and does so through preserving the dialogue initiative, has no problem in identifying which sub-task a user is performing at a certain point provided that the sub-task is legitimate on the system's own premises. As we have seen, a capability for intended user task identification may help improving system naturalness in several respects. This capability is needed for the natural handling of unstructured tasks but also seems desirable for the performance of meta-communication tasks such as dialogue clarification and repair. If users were able to take the initiative in explaining or defining their clarification and repair needs at a

certain point during dialogue, then the system would no longer require a command/keyword-based approach to clarification and repair (cf. Section 1.4). Furthermore, a system capability for intended user task identification might improve the naturalness of user performance on *structured* tasks. Thus (a), a study of user-travel agent communication showed that the user normally had the initiative at the opening of the dialogue in which the user explained the nature of the intended task to the travel agent. From this point onwards, however, the travel agent took over and completed the task by asking a series of questions of the user [Dybkjær and Dybkjær 1993]. (b) Users might be allowed to deviate from the strict pattern of one system question/one cooperative dialogue act response and might even be understood by the system when producing utterances which are out of sub-task context (cf. Sections 1.6 and 1.7).

Secondly, *user initiative* will become more widespread in user-system dialogues, which raises the question of whether to explicitly represent who has the initiative and who responds at any given point during dialogue.

Thirdly, if increased user initiative is to be allowed along the lines just described, user dialogue turns will necessarily involve *longer utterances* than is currently the case. P1 requires that user utterances have a maximum average length of 4 tokens (words) and that no user dialogue turn includes more than 10 tokens. These constraints will have to be significantly relaxed with the result that the amount of spontaneous spoken language, linguistic, discourse and speech act phenomena the system will be required to handle will grow rather dramatically (cf. Section 1.1).

Fourthly, the described increase in user initiative seems likely to impose stronger demands on the system's capability to represent and make use of *dialogue history* (cf. Section 1.7). It will probably be necessary for the system to keep track of the course of the entire dialogue in order to ensure that the system obtain all the information it needs from the user as well as model what the user has already been told. The latter aspect requires the system to maintain a model of the user during dialogue. In P1 the only information registered on the user is whether s/he knows the system or not.

2 Dialogue Design Problems

Fundamentally the dialogue design process is one of setting up a number of design constraints and criteria and then trading them off against one another during an iterative development process until an acceptable result is achieved. No matter what method(s) are used during this phase the central point to dialogue development is to observe the user-system interaction to see whether the dialogue model is in all respects appropriate in functionality and satisfies the design decisions. So what to look for are user problems, system problems, and feasibility constraints which are not satisfied during interaction. The identification of such problems during test of the (simulated) system may lead to new design commitments and should result in changes to the dialogue model before a new test is performed.

This chapter provides a comprehensive presentation of types of problems and violated design constraints illustrated by concrete examples found during the development of a dialogue model for the P1 prototype. The P1 dialogue model development, and hence the identification of problems, was based on Wizard of Oz experiments supplemented with analyses and walk-throughs of dialogue transcriptions, interviews with users, questionnaires and the designer team's intuitive feel for how natural and easy the communication with users proceeded. The final design commitments made for the P1 dialogue model are represented in DSD-frame No. (7). DSD (Design Space Development) is a notation for representing design space structure, designer commitments and design rationale during artifact design [Bernsen 1993a]. DSD-frame No. (7) is presented in Appendix A and will be referred to throughout this chapter. The presentation below goes beyond the design commitments of DSD (7) in that it prepares the testing and revision of P1 by proposing possible new design commitments. It should be noted that the WOZ experiments for P1 covered both the reservation task implemented in P1 and two non-implemented tasks, i.e. those of change of reservation and information. The design problems dealt with below pertain to all three tasks.

The aims of the chapter are twofold: Firstly, to present different types of design problems many of which are probably quite general and hence may be found in other development processes. The list of problems cannot be used as a complete check-list but may sharpen the developer's awareness of what to look for. Secondly, solutions to each type of problem are discussed and the involved design commitments mentioned. The presentation of each problem type follows a structure based on analysing the problem with respect to the following five questions:

(a) *Problem*: What is the problem?

(b) *Violation of the DSD (7) design commitment(s) and possible new design commitments*. Which design commitment(s) are violated by the user-system interaction structure prior to the change of that structure? If the appropriate design commitment is already present in DSD (7), this commitment is referred to. If the design commitment is not present in DSD (7), a new possible commitment is expressed.

(c) *Justification*: What is the justification of the design commitment(s)? In most cases (except for Problem No. 18 below), the justification is expressed in terms of user properties which have to be taken into account in designing a usable system.

(d) *Examples*. Examples are provided of each problem type. How was each particular example identified: through analysis by the designers themselves or through empirical discovery?

(e) *Solution*: How was the problem managed? Was it fully solved, was a trade-off necessary and might other solutions have been possible?

In cases where a solution is a trade-off between two or more design commitments this has been represented by mentioning the relevant options, the conflicting commitments involved, the resolution and the justification for choosing it.

2.1 Problems addressed during dialogue development

To develop a dialogue model for P1, seven iterations of WOZ experiments were performed [Dybkjær and Dybkjær 1993]. During the iterations a number of problems and violated design commitments were identified leading to changes in the dialogue model which was expressed in a graph structure. Below, a specific iteration will be referred to as WOZ x where x is a number between 1 and 7. Most of the problems were anticipated through design analysis and walk-throughs of the graph structure and changes made before users met the problems. Some problems have not resulted in changes to the graph structure. The majority of these were discovered during the seventh and last iteration. Some of the changes were subsequently made in the implemented dialogue model. A small number of problems were not solved in P1 for some reason or other (see below).

All the types of problems to be described in what follows have been derived from the problems and violated design commitments found during the WOZ iterations. The problem types have been divided into three main categories: Problems concerning domain and task coverage, problems concerning interaction, and problems concerning feasibility. However, before the description of problem types is given, three groups of design commitments are listed. They are, first, the DSD (7) design commitments which were found to be violated during dialogue development. Secondly, the DSD (7) design commitments which were not violated are briefly discussed. Finally, a few new commitments are proposed which represent potentially desirable design commitments and which may provide solutions to some of the problems met with during the design of P1. Hence they should be considered for P2.

Violated DSD (7) design commitments

System commitments:

500 words vocabulary;

Large enough task-related vocabulary;

Close-to-real-time response;

Take users' relevant background knowledge into account;

Take into account possible (and possibly erroneous) user inferences by analogy from related task domains;

Sufficient task domain coverage;

Same formulation of the same question (or address) to users everywhere in the system's dialogue turns;

Be fully explicit in communicating to users the commitments they have made;

Reduce system talk as much as possible during individual dialogue turns;

Avoid evoking wrong associations in addressing users;

Avoid superfluous or redundant interactions with users (relative to their contextual needs), i.a. by reusing information;

System task commitments:

Understand user utterances in task domain;

Clear and comprehensible communication of what the system can and cannot do;

Make system limitations clear to users from the outset;

Provide clear and sufficient instructions to users on how to interact with the system;

Provide feedback on each piece of information provided by the user;

User task commitments:

Use single sentences (or max. 10 words);

Use short sentences (average 3-4 words);

Ability to communicate that system or user understanding has failed (*correct* and *repeat);*

It should be possible for users to fully exploit the system's task domain knowledge when they need it;

Separate whenever possible between the needs of novice and expert users (user-adaptive dialogue): Introduction to system optional for expert users;

Non-violated DSD (7) design commitments

A number of design commitments from DSD (7) were not violated during the WOZ experiments. These commitments are listed below with an explanation of why they were not violated.

System commitments:

The system should understand natural grammar.

The system should be able to handle appropriate semantics.

The system should be able to handle natural discourse.

These three design commitments supporting the functionality of the system were not violated because the wizard did not restrict his understanding as regards grammar, semantics and discourse.

Limited speaker-independent recognition of continuous speech.

This system functionality design commitment was not violated because the wizard did not simulate limited speaker-independent recognition. He simply understood all users.

Intelligible, practicable and principled limitations on natural system performance;

Terse system language;

We did not meet problems which immediately could be referred to these two design commitments. Maybe they were violated in the early iterations but not discovered because of more salient problems.

Task commitments:

The system's task is to make it possible for the user to obtain information on and perform booking of flights between two specific cities.

The systems meta-communication task is to repair if system understanding fails.

The user's task is to obtain information on and perform booking of flights between two specific cities.

Right from the beginning of the experiments it was possible for the user to perform the intended tasks.

The meta-communication facilities for the user should be extended at least by the functions *help, wait* and *restart* (see below).

These functions could have been developed for P1 but were not incorporated. Our attention was drawn to them by the fact that users missed them. Meta-communication problems will be addressed in Chapter 3 below.

Possible new design commitments

Meta-communication facilities (user task commitments):

Help: A facility which can give guidelines on expected or possible user answers at any point during dialogue.

Restart: If something has gone totally wrong in the dialogue, the user should be allowed to start all over again without having to waste time on re-dialling the system. In this case initiated tasks should be cleaned up. For instance, an initiated reservation will be cancelled.

Wait: The user should be allowed to ask the system to wait.

The violated DSD (7) design commitments are addressed in more detail in the following three subsections on problem types concerning domain and task coverage, problems concerning interaction, and problems concerning feasibility.

Domain and task coverage

1. (a) *Problem:* The system specifically announces to have a larger task domain coverage than it actually has.

(b) *Violation* of the DSD (7) design commitments: (i) Clear and comprehensible communication of what the system can and cannot do and (ii) Sufficient task domain coverage.

(c) *Justification:* Risk of communication failure in case of lacking knowledge about what the system can and cannot do. In this case violation of the design commitment leads users to have exaggerated expectations about the system's task domain coverage which again leads to frustration during use of the system.

(d) *Examples:*

1.1. In the Opening Information about the system's task domain: "Danish domestic flights" should perhaps be "domestic flights between Copenhagen and Aalborg".

This problem was discovered through design analysis.

Several identical examples of this user problem were found. No other examples were found.

(e) *Solution:* The concrete problem was never experienced by users, because they were always told the route they were going to book for. A realistic system would comprise all domestic destinations and even during the concluding development of P1 a new destination was added. For these reasons it seemed more clean to maintain the phrase "Danish domestic flights" and not clutter up the opening information with a list of possible destinations which will be extended later. In the final system, of course, the announced domain coverage should be correct.

2. (a) *Problem:* The system has a smaller task domain coverage than it ought to have, given its general task domain purpose.

(b) *Violation* of the DSD (7) design commitment: (i) Sufficient task domain coverage and (ii) Make system limitations clear to users from the outset.

(c) *Justification:* Risk of communication failure in case of lacking task domain information. Full task domain coverage is necessary in order to satisfy all relevant user needs in context. Otherwise, users will become frustrated when using the system.

(d) *Examples:*

2.1. The group travel option is not offered to groups of more than 9 persons (WOZ 5). In WOZ 6 the group travel option is offered to groups of more than 9 persons which are told, however, that the system cannot handle such reservations and that reservation should be made by contacting a travel agent. A solution could be to announce this limitation in the Introduction to the System (cf. example 2.4 below). This would respect the DSD (7) design commitment: Make system limitations clear to users from the outset. On the other hand the introduction should preferably be kept short.

This problem was discovered through design analysis.

2.2. Flight Times: The possibility was overlooked that all flights on a specific day might be fully booked (WOZ 5). In WOZ 6 a new sub-dialogue was introduced to handle such cases.

This problem was discovered through design analysis.

2.3. Change of Reservation: The possibility was overlooked that a client may simultaneously have more than one reservation in the database so that the reservation to be changed would have to be identified among those. The problem was solved in WOZ 4.5.

This problem was discovered through design analysis.

2.4. Clients are not warned from the outset that the system cannot handle reservations involving pets, special luggage or need of being accompanied by flight personnel. A solution could be to announce this in the Introduction to the System (cf. example 2.1 above). This would respect the DSD (7) design commitment: Make system limitations clear to users from the outset. On the other hand the introduction should preferably be kept short.

This problem was discovered through design analysis.

2.5. In WOZ 6 it is not possible to obtain information on the total price of booked tickets nor is it possible (under Information) to obtain a computed price even if the user can provide detailed information on the persons who are going to travel. Only the standard and discount prices for one person can be provided.

This problem was discovered through design analysis.

2.6. In WOZ 7 a subject was very surprised not to be offered the possibility of being put on a waiting list in a case where a flight was already fully booked. This is an example of lacking domain and task coverage due to a lack in the designers' knowledge. The possibility of a waiting list will not be offered in P1.

This problem was discovered empirically.

(e) *Solution:* Many more examples of this type of problem could be quoted. This was the largest category of user problems in the studied material. Presumably, this result is typical of knowledge acquisition phases for artifact development. Such problems are generally due to designers' lack of domain expertise and they were much more frequent in early WOZ versions of the system from whence they were gradually removed. The task domain eventually became more and more well-delimited as the designers' domain expertise increased. In WOZ 7 the system was tested with a number of different scenarios, some of them performed by more than one user. Apparently there were no problems with respect to the domain coverage during the performance of these scenarios which contained a large number of different tasks. However, problems like 2.1 and 2.4 are not so simple to solve. The solution is a trade-off between, on the one hand, a clear and short introduction which does not communicate in detail what the system can and cannot do and on the other hand a lengthy introduction which communicates the limitations of the system but which novice users may be unable to grasp because it contains too much information for them to remember. One solution might be to provide new users with a written introduction to the system and its capabilities. This, however, would go against the aim of building a walk-up-and-use system.

3. (a) *Problem:* The system does indeed possess the required domain knowledge but users do not have access to it or do not have access to it when they need it.

(b) *Violation* of the DSD (7) design commitment: It should be possible for users to fully exploit the system's task domain knowledge when they need it.

(c) *Justification:* Risk of communication failure in case of inaccessible (or not easily accessible) task domain information. In such cases, users may pose questions which the system is unable to understand or answer.

(d) *Examples:*

3.1. Information on flight times: “At what time of day?” The user might be interested in being told all departures or arrivals on a specific day, but the system is not prepared for understanding that question so this problem has not been solved in P1.

This problem was discovered through design analysis.

3.2. Information on prices: “Which price type do you want to be informed on?” The user does not have the possibility of asking about all price types at the same time. This problem has not been solved in P1.

This problem was discovered through design analysis.

3.3. Reservation: There is no link from the Reservation task to Prices for the user who, during reservation, turns out to be interested in special fares. This link was subsequently created.

This problem was discovered through design analysis.

3.4. In WOZ 5 information on a number of travel conditions existed in the dialogue model but could not be accessed through the dialogue structure.

This problem was discovered through design analysis.

3.5. In WOZ 1 and 2 it happens several times that the system books a ticket and asks if the ticket should be sent or if it will be picked up at the airport. The user chooses, e.g., the airport and the system says OK but does not tell when the traveller has to pick it up. Instead the user in almost all cases asks this question after a few moments hesitation as if s/he was waiting for the system to provide the information without being asked.

This problem was discovered empirically.

3.6 Users could obtain full general price information through the Information task. However, they might want to know the price of their tickets as part of the Reservation task. Until WOZ 7 it was only possible to get the general price of a ticket and not the total price when more than one ticket was reserved. It should be remembered here that customers may make reservations for several persons at a time, for different itineraries for each person and of different price types. Many customers (mostly the professional travellers) are not interested in such price information.

Option 1: Provide full price information at the end of a Reservation task.

Option 2: Users who want to know the price of their reserved tickets have to compute the full price information themselves by performing the Information task and repeat most the information which has already been given to the system during reservation.

Conflicting commitments involved:

(a) Avoid superflous or redundant interactions with users (relative to their contextual needs).

(b) It should be possible for users to fully exploit the system’s task domain knowledge when they need it.

Resolution: This is a direct clash between two different design commitments because of the existence of different needs in the user population. A third option was identified and selected:

Option 3: Always inform users about the total price of their reservation (but not its break-down into the prices of individual tickets).

Justification: Compromise between the two relevant design commitments. Professional users loose time on an extra dialogue turn. Users wanting the price information have an important part of what they want when they want it.

(e) *Solution:* Many more examples could be quoted. They were much more frequent in early WOZ versions of the system and many of the identified problems have been solved. Many of them were due to a dialogue structure which contained disconnected fragments which could not be accessed at all or at least not when needed as illustrated in examples 3.3 and 3.4. In WOZ 7 it was possible to access every part of the dialogue structure without the wizard having to improvise. But, as mentioned, not all problems were solved. It seems that this type of user problem represents a major challenge in the design of current systems having limited speech recognition and natural language understanding capabilities. Probably the only way to fully solve this class of problems is through allowing a much freer user-system dialogue than is currently possible (see also problem (7) below).

4. (a) *Problem:* The system does not contain enough phrases.

(b) *Violation* of the DSD (7) design commitment: Sufficient task domain coverage.

(c) *Justification:* Risk of communication failure when the system does not contain the necessary phrases to cover the task domain. Especially in the first WOZ generations phrases were missing and if the wizard had not generated the missing sentences ad hoc the dialogues could not have been performed in a meaningful way.

(d) *Examples:*

4.1. In WOZ 1 only four phrases were questions. This was not sufficient because few system questions meant that the user had to take the initiative and ask questions of the system throughout the dialogue. This does not work because the user cannot be expected to know all the pieces of information which the system needs. An example of a question missing in WOZ 1 is that the system needed the name of the person who was going to travel in order to book a ticket.

This problem was discovered empirically.

(e) *Solution:* The problem was eventually solved by the introduction of additional phrases when there was an obvious need. Typically, the wizard generated ad hoc phrases in cases where no suitable phrase was found in the dialogue model.

Interaction

This section has been sub-diveded into system communication, system limitations and user-system understanding.

System communication

5. (a) *Problem:* The system uses different formulations of the same question to users.

(b) *Violation* of the DSD (7) design commitment: Same formulation of the same question (or address) to users everywhere in the system's dialogue turns.

(c) *Justification:* Need for unambiguous system response (consistency in system task performance). The criterion is meant to reduce the possibility of communication error caused by

users' understanding a new formulation of a question as constituting a different question from the questions they have encountered earlier. Also, since the maximum vocabulary is limited and since users tend to model the system's phrases consistency in formulation is preferable.

(d) *Examples:*

5.1. "On which date will the travel begin?"; "On which date is the travel to start?" (WOZ 6).

This problem was discovered through design analysis.

(e) *Solution:* Many more examples could be quoted. They were much more frequent in early WOZ versions of the system and were gradually removed. That they were not systematically removed at an earlier stage seems to be due to the fact that the designers were not aware of them as constituting a category of its own until the final simulation. The solution was to carefully work through the phrases in the graph representing the dialogue model and re-formulate phrases so that the same formulation was used in similar situations. The problem was fully solved fairly easily. It should be mentioned that experiments have been reported where variation in formulations has been chosen to make the system more human-like but this does not work when the aim is to develop and implement a system with a vocabulary of at most 500 words.

6. (a) *Problem:* The system produces unclear questions or information.

(b) *Violation* of the DSD (7) design commitment: Avoid evoking wrong associations in addressing users.

(c) *Justification:* Need for unambiguous and sufficient system response. The criterion is to reduce the possibilities of communication error caused by evoking wrong associations in users, which in their turn may cause users to make irrelevant statements which the system cannot understand.

(d) *Examples:*

6.1. "Are you particularly [stressed by the wizard] interested in making use of special fares?" (WOZ 6). The word 'particularly' was introduced in order to avoid that users who did not want to make use of special fares answered 'yes'. Experience with users had shown that this might otherwise happen. The change caused an improvement in this respect but the problem did not go away. In the discussion we came across the possibility that users interpret the question (with or without 'particularly') as the question whether they have an interest in travelling as cheaply as possible, which perhaps most people have. Several alternative design options were discussed, including:

Option a. Special fares are offered only after all the information relevant to reservation has been entered into the system's database. This will not do, however, as users who are interested in special fares may have to go through most of the reservation dialogue once again.

Option b. At an early stage in the dialogue the system asks if the user's choice of time of travel depends on the possibility of obtaining special fares.

This problem was discovered empirically.

6.2. In the basic task Change of Reservation: "Do you want to change time, price or destination?" Since the system covered only travels between Copenhagen and Aalborg at the time, this question may cause various kinds of confusion in the user leading to irrelevant questions

or remarks which the system will not be able to understand. A solution is to replace 'destination' by 'outward journey or home journey'. However, as mentioned under problem 1 (e) above, a realistic system would comprise all domestic destinations and during the concluding development of P1 a new destination was added.

This problem was discovered through design analysis.

6.3. To support interactive user-system problem-solving, an 'Interrupt' function had been introduced which gave access to the three functions 'Correct' (to be used, e.g., when the system had manifestly misunderstood a user request), 'Change Subject' and 'Help'. However, many users tended to interpret the 'Interrupt' function as the function 'End Now' (WOZ 5). The function was later removed.

This problem was discovered empirically.

(e) *Solution:* System formulations and keywords were eventually adjusted, especially when user problems were encountered with the understanding of keywords. Also many phrases were reformulated over time to make them clearer and more unambiguous.

7. (a) *Problem:* The user is offered non-wanted information or ditto options which they have to respond to.

(b) *Violation* of the DSD (7) design commitment: Avoid superfluous or redundant interactions with users (relative to their contextual needs), i.a. by reusing information.

(c) *Justification:* Need for non-superflous interaction with the system.

(d) *Examples:*

7.1. In WOZ (5) all users, independently of their interest in special fares, were informed on special fare options during flight time decision-making on the Reservation task. In WOZ (6) this has been avoided through the addition of a new sub-dialogue which can be side-stepped by those who do not want the special fare option.

This problem was discovered through design analysis.

7.2. Information: Prices: A conclusion is missing to this information sub-task, which would allow users to ask for more information without returning to the main menu and go back down from there (WOZ 5). WOZ 6 has a conclusion to Information sub-tasks which allows users to immediately ask for more information. However, since information includes prices as well as travel conditions and times, an even more ideal solution would be to allow users to stay within, e.g., the time information domain if they so wish. This is currently not possible.

This problem was discovered through design analysis.

(e) *Solution:* More examples could be quoted of this type of user problem. Many of these are in fact trade-off problems which do not have simple solutions by appeal to one single design commitment. The reason is that adding more choice options to the graph has the negative effect of burdening with more dialogue turn-takings those users who do not want additional options but want to conclude the dialogue or return to other basic tasks as quickly as possible. This problem can only really be solved, it seems, by improving the language understanding capabilities (or the intelligence) of the system and in fact mirrors problem (3) above in which users cannot access information when they want to.

8. (a) The system talks too much during individual dialogue turns.

(b) *Violation* of the DSD (7) design commitment: Reduce system talk as much as possible during individual dialogue turns.

(c) Justification: Users get bored and inattentive from too much uninterrupted system talk.

Since the system already expressed itself rather succinctly on the topics it addresses, the solution does not lie in simply revising its formulations. Instead, the designers undertook a number of more or less hazardous trade-offs between the new design commitment and existing ones, as the following examples show.

(d) Examples:

8.1. Information: Travel Conditions: The designers removed an admonition concerning prohibited luggage contents, regarding it as being 'relatively irrelevant'. The trade-off is the following:

Option 1: Remove admonition concerning prohibited luggage contents.

Option 2: Retain admonition concerning prohibited luggage contents.

Conflicting commitments involved:

(a) Reduce system talk as much as possible during individual dialogue turns.

(b) Complete task domain information.

Resolution: Option 1. Justification: This admonition is relatively insignificant and self-evident. No third option was found which might remove the conflict between the two commitments.

8.2. Reservation: The designers removed the information to users that these would receive an invoice to be paid at the post office:

Option 1: Remove the information to users that they will receive an invoice to be paid at the post office.

Option 2: Retain the information to users that they will receive an invoice to be paid at the post office.

Conflicting commitments involved:

(a) Reduce system talk as much as possible during individual dialogue turns.

(b) Complete task domain information.

Resolution: Option 1.

Justification: This information is irrelevant to business customers who have an already agreed-upon method of payment. Other customers would 'naturally' expect the method of payment described. This justification was not found to be convincing. As a result, a third option was invented which resolves the conflict between the commitments above:

Option 3: When customers obtain their customer number (from a human travel agent), the method of payment is agreed upon.

8.3. Information: The designers decided to split the information on special fares and their conditions between the Price sub-task and the Travel Conditions sub-task. As a result, customers would have to carry out two different Information sub-tasks to obtain full information on special fares:

Option 1: Provide full information on special fares under the two different sub-tasks Prices and Travel Conditions.

Option 2: Provide different parts of the information on special fares under the two different sub-tasks Prices and Travel Conditions.

Conflicting commitments involved:

(a) Reduce system talk as much as possible during individual dialogue turns.

(b) It should be possible for users to fully exploit the system's task domain knowledge when they need it.

Resolution: Option 2. Justification: Criterion (a) was assigned the stronger weight. This was not convincing, however, and therefore a third option was found which might resolve the conflicting commitments:

Option 3: Introduce a direct link between Prices and Travel Conditions via an additional dialogue turn. The system would ask: "Do you want to be informed on the corresponding Prices/Travel Conditions?" depending on what sub-task the user is currently engaged in.

8.4. Information: Check-In Time: The designers decided to only inform about special check-in times for travellers who accompany infants or are in need of travel personnel support. This information was not also provided under Information: Travel Conditions: Infants, and under Information: Travel Conditions: People in need of travel personnel support. As a result, customers would have to carry out two different Information sub-tasks to obtain full information:

Option 1: Provide full information on check-in times under the two different sub-tasks Information: Travel Conditions: Infants, and Information: Travel Conditions: People in need of travel personnel support.

Option 2: Provide different parts of the information on check-in times under the two different sub-tasks (1) Information: Check-In Time and (2) Information: Travel Conditions: Infants, and Information: Travel Conditions: People in need of travel personnel support.

Conflicting commitments involved:

(a) Reduce system talk as much as possible during individual dialogue turns.

(b) It should be possible for users to fully exploit the system's task domain knowledge when they need it.

Resolution: Option 2.

Justification: Commitment (a) was assigned the stronger weight. No third option was found which might remove the conflict between the commitments. In applying the new design commitment on avoidance of too much system talk, the designers were led to reconsider a number of turn-taking sequences, trying to remove some of their elements:

8.5. Change of Reservation: An extra system question was removed which made sure that the user really wanted to cancel a reservation:

Option 1: Provide feed-back to users that a cancellation is about to be done and ask them to confirm their choice.

Option 2: Make the cancellation when this has been asked for by the user.

Conflicting commitments involved:

(a) Reduce system talk as much as possible during individual dialogue turns.

(b) Avoid superflous or redundant interactions with users (relative to their contextual needs).

(c) Ask confirmation from users before important commitments are entered into the database (new candidate design commitment).

Resolution: Option 2.

Justification: Commitments (a) and (b) were assigned the stronger weight. However, a third option was found which resolves the conflict between the actual or possible commitments (a) to (c). A general 'Correct' function was introduced which always allows users to backtrack one step in the dialogue in order to make changes to the decisions made (see below).

8.6. Reservation and Information: Users were no longer asked if they knew the Flight Number as an alternative to their knowing the departure or arrival time:

Option 1: Ask users about the Flight Number as an alternative to the departure or arrival time.

Option 2: Do not ask users about the Flight Number as an alternative to the departure or arrival time. Conflicting commitments involved:

(a) Reduce system talk as much as possible during individual dialogue turns.

(b) Avoid superflous or redundant interactions with users (relative to their contextual needs).
(c) Take users' relevant background knowledge into account.

Resolution: Option 2.

Justification: It is very uncommon for users to know about Flight Numbers. No third option was found which might remove the conflict between the commitments.

Solution: All the design solutions described involved some kind of trade-off. Such solutions which are improvements-at-a-price should be carefully tested to observe the users' reactions and hence to evaluate whether the solution was a good one.

9. (a) *Problem:* Risk of lacking anticipation of users' relevant background knowledge.

(b) *Violation* of the DSD (7) design commitment: Take users' relevant background knowledge into account.

(c) *Justification:* Need for adjustment of system utterances to users' relevant background knowledge and inferences based thereupon. This is to prevent that the user does not understand the system's utterances or makes unpredicted remarks such as, e.g., questions of clarification, which the system cannot understand or answer.

(d) *Examples:*

9.1. In the system's confirmation of a reservation made, the day of the week is not always mentioned in addition to the date in the early WOZ iterations. The user may want to know which day of the week it is. This has actually been said earlier in the dialogue but it is not evident that the user will be able to remember that information.

This problem was discovered through design analysis.

9.2. Information on 'red' (cheaper) departures on a given day. If a certain 'red' departure is fully booked it is not offered by the system. This might appear all right, but the risk is that

the user knows one of the fully booked departure times as being a 'red' one and asks why it has not been offered. The system will probably not be able to understand this question. The problem could be solved by inserting the phrase 'not fully booked' in the system's formulation of the information. The same problem of course applies for other departures as well (green and blue).

This problem was discovered through design analysis.

9.3. Change of departure time: "At what date do you want to go back?" The system does not also ask for the day of the week. The user may know the day of the week but not the date and may ask if that will do instead, which the system will not understand. This example caused us to have a discussion on the comparative semantics of 'date' and '(week-) day' (in the Danish language) which probably couldn't have been resolved by looking into the linguistics literature. Our conclusion was the hypothesis that 'day (of the week)' is more easily understood as covering dates as well than is the covering of 'day (of the week)' by 'date'. In consequence, it was decided to use the term 'day (of the week)' throughout in the questioning of users and to use both weekday and date in informing users.

This problem was discovered through design analysis.

9.4. Time: No distinction was made between asking users about the time (of day) and the exact time of day, and only the latter was used in the questioning of users. Users may know the time but not the exact time when they want to depart or arrive. The distinction was then introduced.

This problem was discovered through design analysis.

(e) *Solution:* There should be enough flexibility in the system to accept two common but different interpretations of the same formulation (as in the day/date and time/exact time examples). Otherwise a clear and unambiguous formulation should be found.

10. (a) *Problem:* The possibility is not taken into account that users may make (possibly erroneous) inferences by analogy from related task domains.

(b) *Violation* of the DSD (7) design commitment: Take into account possible (and possibly erroneous) user inferences by analogy from related task domains.

(c) *Justification:* Need for adjustment to users' background knowledge and inferences based thereupon. Users may otherwise fail to understand the system.

(d) *Examples:*

10.1. During reservation users are offered information on various types of special fare but not on stand-by fares. The reason is that stand-by tickets cannot be reserved on domestic flights. However, the user might know that stand-by reservations can be made on international flights (within 24 hours of departure). As a result, questions might be posed which the system cannot understand. This problem has not been solved at the time of writing.

This problem was discovered through design analysis.

This type of user problem seems to occur less frequently than most other types discussed in this chapter.

(e) *Solution:* This type of problem is very difficult to address. It certainly requires an unusually good imagination to take into account such user inferences by analogy. We did not dis-

cover any problems of this type which we considered important enough to address in the dialogue model. The problem does not seem to be overwhelmingly important and the best solution probably is to consider each case when it occurs.

11. (a) *Problem:* Non-separation between novice users who need introductory information about what the system can and cannot do and intermediate and expert users who may not or do not need such information and for whom listening to it would only delay task performance.

(b) *Violation* of the DSD (7) design commitment: Separate whenever possible between the needs of novice and expert users (user-adaptive dialogue): Introduction to system is optional for expert users.

(c) *Justification:* There are major differences between the needs of novice and expert users, one such difference being that expert users already possess the information needed to understand system functionality.

(d) *Examples:*

11.1. Introduction (WOZ 7): A new question to users was added: "Do you know this system?" First-time users may obtain additional information about the functionality of the system and about how to communicate with it. Other users may proceed directly with their task.

This problem was discovered from user problems. Users complained that the system talked too much. Consideration of this complaint led to the described design improvement.

No further ways of differentiating between the needs of novice and expert users have been discovered so far.

(e) *Solution:* In WOZ 7 it was made optional to listen to the introduction to the system. However, there was a number of other situations in which shortcuts would also have been desirable, in particular when users wanted information on more than one topic. Users very quickly got used to the system and from thence it became inefficient to go back to the main menu after each task. There obviously is a demand for shortcuts which perhaps could be introduced by allowing a number of keywords at certain points in the dialogue. Probably an expert user would be able to handle this. However, keywords represent a non-optimal solution in spoken language systems and would probably require users to have access to a written manual for the system.

12. (a) *Problem:* The system provides insufficient instructions to users about how to interact with the system.

(b) *Violation* of the DSD (7) design commitment: Provide clear and sufficient instructions to users on how to interact with the system.

(c) *Justification:* Risk of communication failure in case of unclear or insufficient instructions to users on how to interact with the system. Users may become (or remain) confused about the functionality of the system.

(d) *Examples:*

12.1. During the Introduction: Information to first-time users: "In addition, you may use the two special commands 'Repeat' and 'Correct' in order to have repeated or to correct the latest piece of information." This explanation of the functions 'Repeat' and 'Correct' is too brief to be immediately comprehensible to all users.

This problem was discovered through design analysis.

12.2. Introduction: Possibly unclear and not sufficiently emphatic request to the user to respond briefly to the system's questions one at a time in order that the system may understand the user (WOZ 5). In the earliest WOZs (WOZ 1 and 2), no instruction was given to users that they should be brief in addressing the system. The results were many violations of the DSD (7) design commitment: Use single sentences (or max. 10 words). WOZ 6 has improved the information to first-time users in this respect. They are told that they will only be understood by the system if they respond to its questions briefly and one at a time. In fact, users already addressed the system briefly enough but the designers believed that the final system needed a more emphatic admonition to users.

This problem was discovered through design analysis.

(e) *Solution:* System formulations were adjusted until they appeared to serve their purpose. However other parameters were also changed during the development process. For example, the dialogue was made increasingly system-directed.

13. (a) *Problem:* Lack of explicitness in communication of user commitments.

(b) *Violation* of the DSD (7) design commitment: (i) Be fully explicit in communicating to users the commitments they have made and (ii) Provide feedback on each piece of information provided by the user.

(c) *Justification:* Users need feedback from the system on commitments made.

(d) *Examples:*

13.1. Confirmation: The system's description of the reservation made is not fully explicit. The exact hour and possible 'red'/'green' (reduced fare) conditions are missing (WOZ 5). In WOZ 6 a more explicit formulation is provided of the reservation made.

This problem was discovered through design analysis.

(e) *Solution:* The solution is to put more information in the system's phrases, formulate these clearly and in all cases provide sufficient feedback to users.

System understanding

14. (a) *Problem:* The system cannot understand names.

(b) *Violation* of the DSD (7) design commitment: Understand user utterances in task domain.

(c) *Justification:* Need for principled limitations on natural system performance.

(d) *Examples:*

In the first four WOZ generations the wizard accepted names although it was known that the final prototype would not be able to do that. Later, numbers were introduced to identify customers and travellers (see (e) below). This numbering system is sufficient in the task domain. However, it has caused several design problems:

14.1. Why should singles, in contrast to, e.g., companies and heads of families, have both a customer number and an id-number? There is no clear reason for that but the problem has not been solved. The straightforward solution is to have the travel agent inquire if a customer is single and if that person would be happy having only a customer number. This would make the system conform to the design commitment: Avoid superflous or redundant interactions with users (relative to their contextual needs), without any apparent trade-offs having to be made. However, the designers seem to have put more emphasis on a design commitment which has nothing to do with usability, namely, the simplicity and consistency of the numbering system.

14.2. Users are not being told during the system's introduction to first-time users that they need (one or) two numbers in order to make a reservation. Instead, they are asked for their numbers after three dialogue turns on the Reservation task. If they do not have these numbers, they have wasted their time on the system and have to call a travel agent. The straightforward solution is to include information on number requirements in the system's introduction to first-time users. This would make the system conform to the design commitment: Avoid superflous or redundant interactions with users (relative to their contextual needs). However, the designers seem to have put more emphasis on the design commitment: Reduce system talk as much as possible during individual dialogue turns, supported by the assumption that the number of users who would meet (only once!) with this problem will be relatively small. In addition, whereas the system's request for an id-number was originally put to the user at the start of the Reservation task, this request was subsequently moved till later in the Reservation task for a reason which has nothing to do with usability, namely, that this request 'naturally' belonged to the Persons sub-task which was located three dialogue turns down into the Reservation task.

14.3. If, during the Change of Reservation task a customer cannot remember the relevant numbers or dates, what happens? Or, which number or numbers should the customer be able to remember in order to be allowed to change a reservation? At the moment, the customer is being refused if the only number (including the date of departure) remembered is the customer number. This is not a technological problem, but the relevant design commitments are presently unclear and no satisfactory solution has been found.

(e) *Solution:* Current speech recognition systems have no possibility of recognising the multitude of different names that persons have. Person individuation is crucial to billing, issuing of individual tickets, security and emergency management. People or institutions should be billed for the air travels made. Tickets are issued in the names of the individuals who will use them. And even if a traveller does not need a ticket, such as an infant accompanied by an adult, their identities should be registered somewhere when they go on a flight. Finally, individual tickets should be registered as such, i.a. because one and the same person may have booked several flights in the database. Also, flights have numbers, but these are only used internally by the system. Even though current speech recognition systems cannot handle spoken names, they can handle spoken numbers. They can also handle spelling. It would have been a possibility to ask users to spell names and addresses but this would be a tedious and inefficient solution especially when a user is booking tickets for several travellers. The solution was to introduce numbers. Each customer was assigned a customer number and each potential traveller an id-number. These number would act as keys to a database containing the relevant information on customers and travellers:

A customer number is used for mailing bills. When given the number, the system retrieves the relevant name and address from the database. This number is assigned to individuals or institutions through their contacting a travel agent.

An id-number is used to identify an individual traveller by name and age. Every traveller has to have such a number. When given the number, the system retrieves the relevant name and age from the database. The number is assigned to individuals through their contacting a travel agent.

A reference number is used to identify each individual ticket which has been reserved. This number is generated by the database and is only relevant when customers want to change their reservations.

15. (a) *Problem:* The system does not reuse information elicited for another task during the same dialogue.

(b) *Violation* of the DSD (7) design commitment: Avoid superfluous or redundant interactions with users (relative to their contextual needs), i.a. by reusing information.

(c) *Justification:* It is boring and inefficient to ask a user to repeat the same information more than once, in particular when it is not just one single piece of information.

(d) *Examples:*

15.1. It is not possible to obtain a price for tickets which have just been reserved (within the same dialogue) without repeating all but the same information that was used for the reservation. There is no coupling between the reservation and the price computation which makes performance on this task seem very inefficient.

(e) *Solution:* Within the duration of a dialogue all information provided by the user should be stored and continuously consulted to see if the needed information is already there before asking the user to provide it. When in doubt the system should check with the user if a given piece of information can be reused.

User communication

16 (a) *Problem:* Missing repair and other support mechanisms, i.e. how to provide natural interactive user support during user-system dialogue?

(b) *Violation* of the DSD (7) design commitment: Ability to communicate that system or user understanding has failed (*correct* and *repeat*). *Possible new design commitments:* Add the functions *help*, *wait*, and *restart* to the list of meta-communication possibilities.

(c) *Justification:* Without repair and support mechanisms, tasks cannot be satisfactorily performed when something has gone wrong and the user would have to start all over again.

(d) *Examples:*

16.1. In WOZ 7 a user believed that it was also possible to book train tickets. The departure she tried to book was sold out and instead she attempted to book a train ticket. She never mentioned the word ‘train’ during the dialogue but went three times through the same piece of dialogue trying to find out how to book a train ticket. The only result she obtained was to be told that the flight departure she tried to book was sold out. She only told in an interview after the dialogue what her intention had been. In this case an appropriate help function was needed but not available.

16.2. In WOZ 7 a user wanted to obtain information on discount types. The system mentioned four types of discounts and asked which one of these the user would like to have information on. The user had not caught the keywords for the different types and just said something which the system did not understand. The system merely repeated its question and did not state what the four types of discount were, so the user tried yet another couple of formulations until she hit upon a word ('red') which the system understands. So it was not enough just to let the system repeat its last question. It should have repeated all the phrases from its last turn.

16.3. The user cannot suspend the dialogue for a while (e.g. by saying 'just a moment' or 'wait'). This was criticised by some of the secretaries who acted as subjects in WOZ 7 and really missed this function. People often drop in to ask the secretary about something, also when s/he is in the middle of a telephone call.

16.4. The only way in which to stop a dialogue before it comes to its natural end is to hang up (an example is found in WOZ 6). This did not often happen during the WOZ experiments since users were very cooperative and tried hard to perform the tasks given to them. However, real life users will probably be more impatient. If a dialogue is going really wrong or the user thinks s/he has followed a wrong path, it will be easier and more efficient to just start all over again at once rather than wait until the system closes the dialogue and asks if the user wants to do another task. And just hanging up requires a new call to the system.

(e) *Solution:* Repair mechanisms are very important particularly in spoken dialogues where misrecognitions are common. Ideally, interactive user support functions should allow users to obtain contextually relevant help at any time, move freely around within the task domain covered by the system, have system phrases repeated at will and change whatever commitments they have made. Unfortunately, not all of this is possible given current scientific and technological constraints on spoken language dialogue systems. During the early WOZ generations the user-system dialogue was relatively free and natural and the wizard was able to offer an almost ideal interactive user support. Accordingly, the first attempt to add a set of interactive user support functions came close to the ideal just described. Users were offered an 'Interrupt' function (which they sometimes misinterpreted) subsuming the following support functions: 'Correct', 'Change Subject' and 'Help'. However, the 'Help' and 'Change Subject' functions never came close to being operationally specified and were subsequently abandoned.

Adjusting to technological constraints and designer preferences, the 'Interrupt' function was replaced by the two rather basic, context-dependent support functions 'Repeat' and 'Correct' in WOZ 7 (the latter having been part of the 'Interrupt' function). 'Repeat' makes the system repeat its latest utterance (dialogue turn) if, e.g., users believe not to have heard or understood it correctly. 'Correct' enables users to repeat their own latest utterance if, e.g., this turns out to have been misrecognised or misunderstood by the system. A third support function, 'Local Help' will not be implemented in the first prototype for general feasibility (resource) reasons. Local Help, being context-dependent, would make the system explain its latest utterance to users by using different terms.

In WOZ 7 all key information from the user, such as dates, times and destinations were confirmed by the system in order to make it possible for the user to repair at once in case of misunderstanding. A help function would be a desirable means of informing the user on the options available at a certain point during dialogue. In cases where the system keeps misunderstanding the user's phrases, graceful degradation would be useful. In such cases, the user is asked to indicate the information more directly (e.g. by using only a single word). If

this does not help, the user may be asked to spell the word and, finally, if this does not work either, the user may be advised to talk to a travel agent. In the same way as it is possible to have repair mechanisms trigged by keywords such as 'repeat' and 'correct', it should be possible for the user to restart a dialogue whenever s/he wants to by saying e.g. 'restart'. It is difficult to predict how users will react in case of communication failure. It is our experience that subjects sometimes do not react even when the system confirms something else than what they asked for. This might be different in real life practice but it probably would be desirable to build in some checks and warnings to the user if, e.g., s/he has booked two tickets for the same person and the difference between the times of departure is, say, just a few hours.

Feasibility constraints

17. (a) *Problem:* Because of limitations in the number and variation of the user scenarios as well as in the number of simulated dialogues, there are limitations to the vocabulary material obtained through the Wizard of Oz method. This points to one of the limitations of the Wizard of Oz methodology itself. As such, the problem is a basic one which cannot be remedied by any known methodology. This raises the following problem:

(b) *Violation* of the DSD (7) design commitment: Large enough task-related vocabulary.

(c) *Justification:* Need for principled limitations on natural system performance. If only part of the terms natural to users can be used in addressing the system, there is an unprincipled limitation on natural system performance, which is contrary to the DSD (7) design commitment: Intelligible, practicable and principled limitations on natural system performance.

(d) *Examples:*

17.1. Reservation: The system asks: "At what time of day?" (WOZ 6). There is no effective way of ensuring that we have identified all the possible phrases which users might use for stating qualitative times on a given day.

This problem was discovered through design analysis.

17.2. Information on travel times: "At what time of day?" (WOZ 6). The problem is the same as in 17.1.

This problem was discovered through design analysis.

(e) *Solution:* One might collect a number of possibilities for how to express time of day by asking subjects to write down as many possibilities as they can think of. A larger set of scenarios which are designed so that users cannot immediately copy the vocabulary used in them would provide more information. Furthermore, the test of P1 is supposed to provide additional information on users' vocabulary.

18. (a) *Problem:* The total vocabulary size exceeds 500 words.

(b) *Violation* of the DSD (7) design commitment: 500 words vocabulary.

(c) *Justification:* Because of limited project resources the vocabulary has been limited to about 500 words.

(d) *Examples:*

18.1. Examples 17.1 and 17.2 fit here, too. It is obvious that there are many ways in which to formulate an answer to the question "At what time of day?".

18.2. In WOZ 7 one of the subjects experimented with different ways in which to express answers, e.g. by using different synonyms for 'reserve' (book, order) to test if the system would understand them. Although none of the words were unusual, allowing a complete number of synonyms for all the key expressions probably will imply exceeding the 500 words boundary. This points to a usability problem which cannot be solved until the vocabulary can be extended beyond 500 words.

(e) *Solution:* A sub-language vocabulary of about 500 words has been defined on the basis of WOZ 6 and 7 but since the vocabularies used by users in WOZ 6 as well as in WOZ 7 did not clearly converge, it is unlikely that 500 words are sufficient for the task domain. Thus there was clearly too little overlap between the WOZ 6 and WOZ 7 vocabularies. Experiments with P1 are expected to help answering the question of how large a vocabulary is needed.

19. (a) *Problem:* The average length of user utterances exceeds 3-4 words.

(b) *Violation* of the DSD (7) design commitments: (i) Use short sentences (average 3-4 words) and (ii) Close-to-real-time response.

(c) *Justification:* The recognition error rate is larger for longer sentences and because it takes more time to recognise an utterance the longer it is, there is a risk of communication failure in case of delayed response.

(d) *Examples:* Many sentences clearly exceeded the 3-4 words average. In WOZ 1 the average user utterance length was 12 words. This average was eventually brought down. In WOZ 6 an average of 3-4 words was reached [Dybkjær and Dybkjær 1993].

(e) *Solution:* The way to reduce average utterance length was (i) to let the system take more and more of the initiative by turning user questions into system questions. (ii) More information was put into the system phrases, partly to avoid user questions and partly to invite the user to provide only the absolutely necessary information. (iii) The dialogue style was terse and non-polite (but not impolite!) since polite system formulations seem to invite long and polite user replies [Zoltan-Ford 1991]. (iv) Users were asked, as part of the system's introduction, to use brief utterances. Finally (v), resolution of many of the problems described in this chapter has no doubt contributed to preventing users from asking questions of clarification of the system.

20. (a) *Problem:* The maximum length of user utterances exceeds 10 words.

(b) *Violation* of the DSD (7) design commitments: (i) Use single sentences (or max. 10 words), (ii) Close-to-real-time response and (iii) Understand user utterances in task domain.

(c) *Justification:* The recognition error rate is larger for longer sentences and because it takes more time to recognise an utterance the longer it is, there is a risk of communication failure in case of delayed response.

(d) *Examples:*

20.1. In WOZ 1 the system's opening phrase which asked for information on day, hour and destination, invited a long reply which users also produced in most cases.

20.2. In the early WOZ generations many user utterances exceeded 10 words probably because the conversation style was very much that of human-human dialogue.

(e) *Solution:* The way to bring down the number of long utterances is approximately the same as the one followed in decreasing average utterance length (cf. problem 19 above). The decrease in average utterance length resulted in an increase in the number of turns used to perform a task. From WOZ 4 onwards the number of long utterances clearly decreased. In WOZ 7 the wizard simulated limited understanding of long utterances (of which there were very few). If they started with a 'yes' or a 'no' only this word was understood.

21. (a) *Problem:* More than 100 active words at a time are required implying that the system cannot respond in close to real time.

(b) *Violation* of the DSD (7) design commitment: Close-to-real-time response.

(c) *Justification:* Risk of communication failure in case of delayed response. Especially as regards dialogues over the telephone where the interlocutors cannot see each other, it is important to avoid long pauses because they make the other interlocutor unsure of what to do: Should s/he say something? Is there no connection? Is something wrong?, etc.

(d) *Examples:*

21.1. The opening phrase in WOZ 1 asked for information on day, hour and destination, which normally resulted in a long answer from the user the contents of which could not easily be predicted. Sometimes the user did not provide an answer to all the information asked for and sometimes the answer contained information not asked for. Probably the system asked for too much information at a time and users often decided to provide all the information they had almost as if they were talking to a human.

21.2. When a task is finished the system asks "Do you want more?". Here it is rather unpredictable what the user will answer.

(e) *Solution:* The problem was eventually solved by asking more explicit questions and hence for less but more well-defined information at a time. In WOZ 7 it was decided only to accept a yes/no answer to the question "Do you want more?". In situations where it is known that an answer cannot be given immediately, a solution could be to let the system say 'just a moment' (or play a tape with the sound of somebody using a keyboard). However, this possibility has not been implemented in P1.

3 Communication Problems in the Running System

In Chapter 2 a number of problem types in dialogue design were presented and it was described how to prevent each problem type. However, no matter how carefully the prevention of problems has been done, communication problems cannot fully be avoided. This chapter presents a systematic approach to how to address problems in the communication between system and user by allowing meta-communication (cf. Section 1.8).

In principle, both interlocutors have equal possibilities of managing communication problems but in reality this is only true to the extent that they also have the same possibilities of taking the initiative (cf. Section 1.4). Due to technological constraints dialogue systems cannot just leave the initiative to the user without any restriction. In systems like P1 almost all initiative lies with the system. The user only has the possibility of taking the initiative by saying *correct* or *repeat* (cf. Chapter 1). There are other systems, however, and P2 is intended to be among them, which to some larger extent can leave the initiative to the user. The following discussion will focus on meta-communication problems and how to address (1) the system's problems and (2) the user's problems. P1 forms the basis of the discussion, but the chapter has a more general aim and hence is not restricted to fully system-directed dialogue.

3.1 System problems

When a system has been designed on the principles discussed in Chapter 2 it should work well as long as the user provides cooperative utterances which are understood by the system. Cooperative utterances are utterances which a user has a right to expect the system to be able to understand. It is up to the system to inform users on the system's understanding capabilities and limitations. Cooperative utterances must conform to this information. See also Appendix B.

Table 3.1 shows how an ideal dialogue would generally proceed with the system asking the questions. Note that whenever the user has provided a piece of information it is checked for validity and consistency by the system. This check may in case of inconsistency give rise to new sub-dialogues concerning pieces of information which the user has already provided. In particular, when the user wants to correct a piece of information which is not the immediately preceeding one, inconsistencies are likely to appear with other already given information.

However, even in systems which have been carefully developed to prevent communication problems, the system may fail to understand what the user says (or the user may fail to understand the system, see Section 3.2). Although today's dialogue systems are based on the assumption of cooperative users in the sense that users are supposed to say something relevant so that it is possible to make valid predictions [Bilange 1991, Eckert and

Intention of system question	**Cooperative user answer**	**System reaction**
Obtain a new value	The value asked for is provided.	If OK the value is recorded and a confirmation is output to the user. Otherwise the user is told that the value is wrong and why, which may be because it does not exist or is inconsistent with previous information.
Check a value with the user.	Cooperative user answers: 1. Confirmation. 2. Denial. 3. Confirmation + value. 4. Denial + value.	System reactions to the four types of user answers: 1. Record the value. 2. Ask for a new value. 3. If the value provided by the user is the same as the one the system has it is recorded. Otherwise the user is told that there is an inconsistency. 4. If the value provided by the user is different from the one the system has and is otherwise OK it is recorded and a confirmation is output to the user. Otherwise the user is told that the value is wrong and why, which may be because it is the same value as the system already has, does not exist or is inconsistent with previous information.

McGlashan 1993]. Users do not always act cooperatively. This may be, e.g., because they want to experiment with the system.

Table 3.1: Intention of system questions, cooperative user answers and corresponding system reactions.

However, even when users are cooperative the system may fail to understand them. This may, e.g., be due to their use of words or grammar not covered by the system's vocabulary and grammar, or it may be due to a user speaking dialect or having a speech defect. Non-cooperative users will not be dealt with here but of course it should be ensured that users cannot make the system break down. On the other hand, a system should be robust enough to handle most cooperative user input which has not been understood. However, cases such as pronunciational deviations may be hard to handle reasonably as they concern the appropriateness of the system's basic signal processing capacity. Robustness is here taken to mean that the system should be sufficiently flexible, e.g. by accepting more or other relevant information than was asked for, and able to cope with understanding failure by using meta-communication for clarification and repair (cf. Section 1.8).

Table 3.2 shows the possible cases of input understanding in a dialogue system. The top row describes the system's understanding of information related to the current task. The left-hand column describes the system's understanding of information related to other tasks than the current one. The system may have understood nothing, part of the information or all the information provided by the user. Note that what has been understood is not necessarily the same as what the user actually said, which implies that the user must have the possibility of making corrections (see Section 3.2).

The system determines whether is has understood part or all of the information by checking if the information is sufficient for the current task. If, for example, the system has asked the user to indicate a route but only has understood one airport then the system may assume that it only understood part of the information provided by the user. Here it does not really matter whether the system actually did not understand part of the provided information or whether the user did not provide the missing part. The system's reaction will be the same, namely to ask for the missing piece of information.

information concerning current task --- **information concerning other task(s)**	**none**	**partial**	**full**
none	1. degradation	2. finish current task (by asking for missing information)	3. feedback on the acceptability of information on current task
partial	4. finish other task(s) when possible, then return to current task	5. finish current task, then finish other task(s)	6. feedback on the acceptability of information on current task; when the task is finished then finish other task(s)
full	7. feedback on the acceptability of information on other task(s); when the task(s) is finished, then return to current task	8. finish current task, then feedback on the acceptability of information on other task(s)	9. feedback on the acceptability of information on current task; when the task is finished then feedback on the acceptability of information on other task(s)

Table 3.2: Input understood by the system (top row and lefthand column) and system reactions to this input.

When input could be understood it means that it was within the system focus set (cf. Section 1.6). When nothing has been understood, one possible explanation is the occurrence of a non-cooperative answer, e.g. that the user input is not within the system focus set. Other possibilitties are that, e.g., the recogniser or grammar or vocabulary failed.

Each cell of Table 3.2 has been enumerated for ease of reference and briefly indicates the system's reaction to input. As a general rule, the system reaction should be to finish the current task first unless there is no information on it but only about other tasks. Partial understanding requires exchanges to obtain complete information. When partial or complete information has been understood the system should check this information with its database and with already provided information. If no conflicts are found the system should provide feedback on what it has understood and otherwise an error message should be given to the user. In cell 1 understanding has totally failed and the method to be used in this situation is graceful degradation which is described below. In cell 2 partial understanding of information concerning the current task has been achieved by the system, e.g., only the arrival airport was understood but not the departure airport of a route. Cell 3 corresponds to Table 1 which is the ideal situation. The user has provided the input cooperatively and the system has understood it. The input may have to be completed by the system. For instance, in P1 it is acceptable to indicate the date of departure as "On monday" or "On the 23rd". Such information is turned into a full date by the system on the basis of rules corresponding to those commonly used by humans. For instance, the 23rd will be taken to refer to the 23rd day of the present month if this day still lies ahead and otherwise to the 23rd day of the next month. Cells 4 and 7 express that the user has provided information on other tasks in system focus than the current one. There may be dependencies between tasks so that it is necessary to finish one task before another. For instance, it does not make sense to determine the hour of departure before a route has been determined. Cells 5, 6, 8 and 9 express that the user has provided information not only concerning the current task but also concerning one or more other tasks in system focus. In this case the current task should be finished before proceeding to (one of) the other task(s). An example of cell 9 could be that the current task is about finding a date of departure and the user provides the date as well as the hour.

Levels and graceful degradation

There may be several reasons why understanding fails and these are not necessarily due to a non-cooperative user, as already mentioned. If due to difficulties in recognising a user's pronunciation of certain words a first reaction could be to ask the user to repeat the utterance. However, if due to, e.g., an overly complicated utterance a simple repetition will not help. In this case it is necessary to make the user express the information in a simpler way. It will probably not be possible to detect exactly why understanding has failed in each case so a general method to handle any situation involving system understanding problems is needed. We propose to use some kind of graceful degradation [Heisterkamp 1993].

Graceful degradation is a method by which the system will explicitly ask the user to provide the missing information in increasingly simple terms. The degradation of communication level will continue until either the system has understood the user's input or no further degradation is possible (see also the degradation algorithm below).

We have found it appropriate to distinguish between five different levels of communication (cf. Chapter 1.6). These five levels involve the five different types of

question shown in Table 3 (questions 1-5) along with an explanation of cooperative user answers. The sixth type of question mentioned in Table 3.3 does not correspond to any particular level. Rather such explicit clarification questions are level-preserving and may be needed at any level. For instance, the system can ask the user to repeat any level.

The first four types of question (levels) and the sixth one in Table 3.3 are initiative-preserving which means that they do not offer the user the initiative. If the system already has the initiative then it keeps it if the user answers its questions cooperatively and does not actively try to take over the initiative, e.g. by asking for repetition or clarification. If the user had the initiative just before the system took over in order to handle a communication problem, then the initiative may be returned to the user when the problem has been solved. The fifth type of question (corresponding to the fifth level) offers the user the initiative.

In a graceful degradation process, a first step could be to ask the user to repeat whenever an utterance was not understood and before asking for the information in a new way. This would imply the insertion of the repeat possibility at each of the three steps in the degradation algorithm below. To ask for repetition is the least possible step in the direction of trying to repair an understanding problem. However, it does not always make sense to ask for repetition. In cases of partial understanding, such as when the system has only understood the arrival airport and not the departure airport, it would not be reasonable to ask for repetition.

System question	**Cooperative user answer**
1. Questions which require the user to spell the answer.	The user spells.
2. Yes/no questions (e.g., Do you want to x?).	The user answers yes or no.
3. Multiple choice questions (e.g., Do you want to a, b or c?).	The user selects one of the offered choices a, b or c.
4. Focused questions (e.g., when, where, which, how many, ...).	The user answers by clearly giving the desired piece of information.
5. Open questions (e.g., Can I help you?/What do you want?/ ...).	Answers like “I would like to reserve a ticket to Aalborg” or “Yes, I want to make a reservation” which are within the task domain and to be expected at the given point in dialogue.
6. Explicit clarification questions (e.g., Please repeat, I did not understand; Are you still there? (timeout)).	When asked to repeat the user repeats the answer, perhaps in a simplified version. The user finally answers after a period of silence.

Table 3.3: Types of system questions and cooperative user answers.

A tentative method for carrying out graceful degradation when system understanding fails could be the following 'degradation algorithm' involving three steps :

1. Initialisation: If the system does not already have the initiative then it should take the initiative and ask the user a question concerning what it thinks the user is about to tell the system. This may be determined on the basis of the system focus set. If understanding fails again proceed to step 2, otherwise stop degradation.

   An example could be that the system believes that the user wants to make a reservation but has no information on the reservation yet, and therefore asks "Where does the travel start?".

2. Explicitness iteration: Make explicit to the user what was implicit in the system's original question. This step may be iterated a number of times. If understanding still fails, proceed to step 3, otherwise stop degradation.

   An example could be to stress that the user's answer is supposed to mention one of three possibilities offered by the system. There are other possibilities of making questions explicit, however, not all of them involving repetition of the last system question together with an indication of the type of answer expected. If the system has understood part of the user's input, e.g., when the user was asked and has indicated a route and the system only understood the arrival airport, then one way of making explicit the next question would be to confirm the arrival airport and ask for the departure airport.

3. Level iteration: Ask a question which can be answered in a different and simpler way to provide the same information, i.e. degrade to the next relevant level and proceed to step 2 if understanding fails, otherwise stop degradation. When no lower level exists a bottom stop condition should be activated. This could be to stop the dialogue with a reference to a person who may help the user achieve his/her goal or if possible the system could decide to postpone the acquisition of the piece of information causing trouble.

   An example of degradation to lower levels could be that the system has asked for a departure airport. P1 is a small prototype and only includes three destinations so it would be possible to ask the same question as a multiple choice question (i.e. make level iteration) and then return to step 2 if understanding fails. If at step 2 it turns out that a new degradation of this question is necessary (i.e. a return to step 3) then it could be asked as a yes/no question. The lowest level would be to ask the user to spell the name of the airport. If this fails and a new visit to the second step, i.e. making explicit the question does not help either, then when returning to step 3 again the bottom stop condition should be activated.

During graceful degradation it is not always a solution to degrade to the level immediately below the current one. In the example of asking for an arrival airport it would not make sense to formulate the question as a multiple choice question if there were, e.g., ten destinations. In this case the next relevant level would be to ask the user to spell. So the problem is how to decide the next relevant level. This may be done as follows: For each piece of information to be obtained, all five levels could be indicated along with a grammar telling how to ask for that information. If it does not make sense to use a certain level, no

grammar should be indicated, and if it only sometimes makes sense the grammar should be conditioned. An example of a conditioned multiple choice formulation could be:

If the number of relevant airports which can be referred to is

1-3: a grammar should be indicated.

> 3: no grammar should be indicated.

If the system has asked for the departure airport and does not know the arrival airport there may be fairly many possible airports for the user to refer to. However, if the departure airport has been determined and the system has asked for the arrival airport the number of possibilities may be reduced to few airports. For instance, if, in P1, the departure airport is not Copenhagen then it is almost certain that the arrival airport is Copenhagen.

It should be noted that graceful degradation was not implemented in P1 but will be considered for P2. In case of communication problems P1 only provides the possibility of asking for repetition for the system as well as for the user. So the current level is preserved. P1 primarily asks questions at levels 2-4, i.e. yes/no, multiple choice, and focused questions. It does not contain the spelling level and only in one situation an open question is asked ("Do you want more?").

3.2 User problems

In Section 3.1 a normative approach was proposed as to how user input may be handled by the system, especially when it did not understand it. This approach was possible because it is the system designers who decide how the system should react. However, as they cannot decide how users react, a descriptive and empirical approach must be adopted to user reactions and understanding of system output. As a first step, one may try to anticipate the user needs and actions which create or express the presence of communication problems in order to hypothesize the kinds of meta-communication needed. It seems obvious that users will need both dialogue clarification and dialogue repair (cf. Section 1.8). The description of possible user needs and actions will serve as a basis for deciding what the system should allow users to do and which cases it should be prepared to handle in order to be sufficiently robust.

User needs and actions	**System reaction**
Silence.	Ask if the user is still there and repeat the question (timeout); if no reaction after some interval (e.g. after a certain number of timeouts) then hang up.
Wait (the user needs time e.g. to think or to talk to somebody).	Wait for a time interval, then hang up if the user did not return. The user is expected to indicate that s/he is ready again and then the system will resume the dialogue.
Repetition (the user needs to have the latest system utterance(s) repeated, e.g. to ensure correctness of understanding).	The most recent system utterance(s) is (are) repeated or explained in more detail.
Correction (the user needs to correct previous information).	The user is prompted for a new answer to the system's request for that information.
Help (the user needs help from the system to get on with the dialogue). Actions such as unexpected, confusing or irrelevant answers or repeated use of the repeat command may be taken to indicate a need for help.	The system provides information on possible user answers. If possible the information should depend on the context, i.e. the current point of dialogue and the previous user input.
Restart (the user needs to start all over again, e.g. because too many things have gone wrong during the dialogue).	The system asks what the user wants (e.g. reservation, change of reservation or information).
Hang up.	Quit and clean up.
Extra information (the user provides information which s/he was not encouraged to give).	The system may take one of the following actions: 1. accept only the part which was asked for; 2. accept all of it if possible; 3. reject.
Question (no answer to the system question is provided; instead the user asks a question for clarification).	1. reject and ask the question again; 2. accept, suspend the present subdialogue if necessary and answer the question.

Table 3.4: Types of user needs and actions and corresponding system reactions.

In the same way as was shown for the system in Table 3.2, the user may have understood nothing, part of or all of the system output. If the user has understood everything and found no reason to disagree, the conversation may continue smoothly. In all other cases the user may take or preserve the initiative and begin dialogue clarification or repair the aim of which is to solve the problem. The more robust and flexible a system is the more user needs and actions it will be able to handle. Table 3.4 presents a number of expressed user needs and actions many of which were observed during the development of P1, together with suggestions for reasonable system reactions. Note that in fully system-directed dialogues all kinds of user questions will cause difficulty because they imply a shift of initiative to the user such that the system no longer has full control of the dialogue.

The need for help has two aspects. The user may need help to continue the dialogue, e.g. by being told what possibilities s/he has at the present point of dialogue. The user may also need domain help, i.e. help e.g. to understand a certain concept such as “red discount”. Help to continue the dialogue has much in common with clarification and graceful degradation.

When, in P1, the user wants to ask for correction or repetition this can be done only by using the keywords *correct* (for dialogue repair) and *repeat* (for dialogue clarification), which is not very flexible. It may work when there are only two such keywords to remember but if there are more than that, one cannot expect that users who are new to the system will grasp all these keywords and how to use them during the introduction to the system. On the other hand, a manual would be against the idea of a walk-up-and-use system. The solution probably is to allow more natural formulations of users’ need for meta-communication. It is probably much easier to remember a number of useful functions than a number of precise keywords which do not necessarily coincide with the words that users find most natural for describing the corresponding function. The system’s problem will of course be that there are many possible ways in which to ask for, e.g., repetition or correction which implies a need for a large active vocabulary.

4 Conclusion

A first theory of task-oriented spoken language dialogue has been outlined in the previous chapters. The theory is based on experience from the development of P1. There is a large literature on human-human dialogue theories, but such theories cannot immediately be tranferred to human-computer dialogue and are often far from being implementable. However, many ideas and concepts from human-human dialogue theory are useful for spoken language dialogue systems development and have been recast above to serve as a basis for the design, representation and implementation of such systems.

A detailed analysis of two topics has been provided in this report. Firstly, it was discussed how to prevent the occurrence of communication problems through design (Chapter 2). A number of problem types illustrated by examples from the development of P1 were presented along with possible solutions. The problem types and their solutions are supposed to be valid not only for P1 but more generally and will be taken into consideration in the development of P2. Secondly, a first analysis has been provided of how to deal with the problems of communication that inevitably occur during dialogue no matter how carefully a system has been designed (Chapter 3). This was done via a normative description of the system's possible actions with focus on meta-communication and graceful degradation. A descriptive approach was taken to users' possible, expressed needs for meta-communication.

Comparison with related work

The SUNDIAL project [Peckham 1991, Peckham 1993] has many aspects in common with the Dialogue project. Both projects concern airline travel information systems and the overall system architectures are much the same. Therefore a comparison with the dialogue theory and dialogue handling used in the SUNDIAL project is of obvious interest. The theory is, like the theory outlined in this report, based on oral task-oriented human-computer considerations rather than on human-human dialogue theories. [Bilange 1991] mentions many of the elements discussed in Chapter 1 such as predictions, initiative, dialogue history, evaluation and meta-communication. The SUNDIAL dialogue handling module mainly consists of three modules Bilange [1991, Eckert and McGlashan 1993]: a task module representing domain knowledge, a dialogue module deciding on the next action to take, making predictions (dialogue allowances) and coping with dialogue history, and a belief module which registers information on user utterances. This information is used for deciding how and about what to address the user. It roughly corresponds to the user status of P1 (cf. Report 6b) which is part of the dialogue description. The database of P1 roughly corresponds to the SUNDIAL task module, and the P1 dialogue description corresponds to the SUNDIAL dialogue module plus its belief module. In many respects the two dialogue handling modules seem to be quite similar. However, the SUNDIAL system clearly has a more robust handling of communication problems through the use of graceful degradation (cf. Chapter 3 and [Heisterkamp 1993]).

Another important and interesting work is the Circuit-fix-it Shop by [Smith 1991] who views task performance as theorem proving within the domain and claims that "the role of language is to supply missing axioms to complete these proofs." He further defines a sub-dialogue as "all the language interaction pertaining to one task step." This approach provides a simple and elegant framework for the system structure, and a good distinction between domain and dialogue descriptions. However, the feasible or possible proof sequences do not necessarily match a dialogue structure natural to humans. Smith manages this potential dilemma by introducing an interruptible theorem prover which is able to suspend proof sequences while initiating or resuming others. The actual control resides in the dialogue part which decides whether to follow the directions from the theorem prover, follow the user's commands or take an entirely different action (e.g., of clarification).

The approach of [Smith 91] provides clear insights on the respective roles of domain, dialogue and language, and is useful as a guidance for developing the operational aspects of dialogue theory as well as during implementation. However, he has little to say on how to design smooth dialogues and how to handle dialogue problems (though much of the technical apparatus seems to be present). These two points are precisely those elaborated in Chapters 2 and 3 above. Moreover, his dialogue theory primarily concerns the domain representation and the underlying theorem prover. The emphasis is not on the dialogue situation and on naturalness. So the dialogue theory will probably only be well-suited in cases where the structure of the theorem prover is in accordance with what would be a natural and appropriate dialogue structure.

Work in progress and future work

Report 6b describes a concrete representational framework based on the theory outlined in this report. This framework has been used in the specification of P1 which shows that the theory discussed in the present report is sufficient at least for P1. In addition, Report 6b describes the implementation of P1. The ongoing test of P1 is expected to provide input on possible improvements of the theory and hence of the design of P2 which will begin shortly. The design of P2 will include new WOZ experiments and may also require extensions to the dialogue theory described above.

Appendix A: Design Space Development

DSD (Design Space Development) is a notation for representing design space structure, designer commitments and eventually design rationale during artifact design from the point of view of usability engineering. DSD is being developed as part of CCS's participation in the Esprit Basic Research AMODEUS-2 project [Bernsen 1993a]. A series of DSD frames provide a series of 'snapshots' of the developing design process. When combined with a Design Rationale representation, DSD makes explicit the constrained usability reasoning which has gone into designing the artifact. The following DSD frame (7) represents the design commitments related to usability engineering which were made during P1 dialogue model development and succeeds DSD (4) [Bernsen 1993b] and DSD (6) [Bernsen 1993c].

DSD No. (7)

Conventions for shorthands and notation are given in paragraph E at the end of the appendix.

A. General constraints and criteria

Overall design goal:

- Spoken language dialogue system prototype operating via the telephone and capable of replacing a human operator;

General feasibility constraints:

- 20 m/y's available for the first version of the prototype;
- Limited machine power available;

Scientific and technological feasibility constraints:

- Limited capability of current speech and natural language processing;
- Open research questions, e.g. research in dialogue theory;

Designer preferences:

- Use of the Dialogue Description Language (DDL);

Realism criteria:

- The artifact should meet real and/or known user needs;
- The artifact should be preferable to current technological alternatives;
- *The system should run on machines which could be purchased by a travel agency;*
- The artifact should be tolerably inferior to the human it replaces, i.e., it should be acceptable by users (to be expanded in the usability criteria) while offering travel agencies financial advantage;

Functionality criteria:

- Make sure that the artifact *can do* the tasks done by the human it replaces;

Usability criteria:

- Maximize the naturalness of user-interaction with the system;
- Unless a naturalness criterion cannot be met for feasibility reasons, it should be incorporated into the artifact being designed;
- Constraints on system naturalness resulting from trade-offs with system feasibility have to be made in a principled fashion based on knowledge of users in order to be practicable by users;
- Constraints on system naturalness have to be clearly communicated to users;

B. Application of constraints and criteria to the artifact within the design space:

C = Null

O = Null

S = 500 words vocabulary;

Large enough task-related vocabulary;

Natural grammar;

Appropriate semantics;

Natural discourse handling;

Limited speaker-independent recognition of continuous speech;

Close-to-real-time response;

Take users' relevant background knowledge into account;

Take into account possible (and possibly erroneous) user inferences by analogy from related task domains;

Sufficient task domain coverage;

Same formulation of the same question (or address) to users everywhere in the system's dialogue turns;

Terse system language;

Be fully explicit in communicating to users the commitments they have made;

Reduce system talk as much as possible during individual dialogue turns;

Avoid evoking wrong associations in addressing users;

Avoid superfluous or redundant interactions with users (relative to their contextual needs), i.a. by reusing information;

Intelligible, practicable and principled limitations on natural system performance;

I = Spoken telephone dialogue;

T = **User tasks**

Obtain information on and perform booking of flights between two specific cities;

Use single sentences (or max. 10 words);

Use short sentences (average 3-4 words);

Ability to communicate that system or user understanding has failed (*correct* and *repeat);*

It should be possible for users to fully exploit the system's task domain knowledge when they need it.

System tasks

The system's task is to make it possible for the user to obtain information on and perform booking of flights between two specific cities.

Ability to repair if system understanding fails;

Understand user utterances in task domain;

Clear and comprehensible communication of what the system can and cannot do;

Make system limitations clear to users from the outset;

Provide clear and sufficient instructions to users on how to interact with the system;

Provide feedback on each piece of information provided by the user;

U: *Normal adult native Danish speakers;*

E = Novices, walk-up-and-use users;

Ordinary experts who understand the tasks and its vocabulary;

Top experts, i.e. skilled users of the system;

Separate whenever possible between the needs of novice and expert users (user-adaptive dialogue): Introduction to system optional for expert users;

C. Hypothetical issues:

- Sufficient accommodation to experienced users?
- Is a vocabulary of 500 words sufficient to capture the sublanguage vocabulary needed in the task domain ?

D. Documentation:

- *Final user-system interaction graph;*

E. Conventions:

C = Collaborative aspects.

O = Organisational aspects.

S = System aspects.

I = Interface (or more generally: system Image) aspects.

T = Task aspects including task domain aspects.

U = User aspects.

E = User experience aspects.

DSD No. (n) indicates the number of the current DSD specification.

'Null' means that the artifact does not embody a certain aspect of DSD.

Italics indicate new elements in DSD (n) as compared to DSD (n-1).

Note that, in the present design process, the *C* and *O* aspects have been temporarily set to null during design of the first prototype and will be reconsidered during design of the second prototype.

Appendix B: The Principle of Cooperativity

The design commitments listed at the start of Section 2.1 and represented in the Design Space Development (DSD) frame (7) (see Appendix A) specify principles which the designers have agreed to adhere to during a specific design process. The majority of those commitments are expressed such that adherence to them guarantees that various classes of usability problems can be avoided. These commitments will be valid if not sufficient for a wider range of dialogue systems, including P2.

It is interesting to briefly compare our list of design commitments intended to avoid the usability problems discussed in this chapter with Grice's *cooperative principle* [Grice 1975, cf. Nofsinger 1991]. Paraphrased, the principle states that dialogue partners should say just what is needed at any given moment in order to maintain a smooth dialogue. The principle is valid for 'serious, information-oriented conversation' which is precisely the kind of dialogues addressed in P1 and P2. The cooperative principle is constituted by a number of *maxims* which, recast in the context of a *computer* speaking generate the list below. For each maxim the corresponding design commitments are indicated in brackets, cf. Appendix A:

strength: say enough, avoid having the user ask for information when the information can be provided by the system right away (be fully explicit in communicating to users the commitments they have made, provide feedback on each piece of information provided by the user, make system limitations clear to users from the outset);

parsimony: don't tell the user more than necessary, it becomes boring to listen (reduce system talk as much as possible during individual dialogue turns, terse system language);

truth: don't tell the user anything that is false (check all information before giving it to the user). Our commitments list does not have a corresponding commitment because this maxim has been taken for granted. However, one of the worst breakdowns through the WOZ experiments occurred when the wizard came up with an impossible day of the week;

evidence: don't tell the user anything for which there is no evidence. Our commitments list does not have a corresponding commitment because this maxim has been taken for granted. However, the commitments (clear and comprehensible communication of what the system can and cannot do) are relevant at this point and the system actually does check all information before giving it to the user;

relevance: don't tell the user anything that is not of current interest or does not advance the task (avoid superfluous or redundant interactions with users (relative to their contextual needs));

clarity: use language and terminology that is familiar to the users (avoid evoking wrong associations in addressing users, take users' relevant background knowledge into account, take into account possible (and possibly erroneous) user inferences by analogy from related task domains).

It thus turns out that the Gricean maxims are essentially matched by a subset of our design commitments. However, there are many more design commitments relevant to system communication. Some of these deal with the system's understanding of the user and serve to anticipate and constrain the behaviour of cooperative users (see below). The rest mainly fall into the groups of domain and repair mechanisms. In particular, one may wonder why repair mechanisms have been left out by Grice. Even in human-human dialogue, cooperativity does not guarantee the absence of needs for dialogue clarification and repair. Domain completeness, on the other hand, cannot normally be expected by human interlocutors.

Conversely, the Gricean maxims may be applied to the *user* and indeed the user is assumed to behave largely cooperatively, i.e., to adhere to the cooperative principle and its maxims. The system always checks for *truth* and *evidence* and is prepared to ask further questions if the *strength* maxim has not been followed. The three most difficult user maxims for system designers to cope with are perhaps those of parsimony, clarity and relevance. A major aim of the WOZ experiments has been to tailor system vocabulary, grammar and semantics to that required in the domain and hence to make sure that users who follow the *clarity* maxim will be understood by the system. *Parsimony* and *relevance* have been supported by the admonition to users to answer the system's questions briefly and one at a time. This amounts to the explicit addition of an extra constraint on parsimony and relevance as compared to Grice. Finally, repair and clarification mechanisms are extremely important because the system will sometimes fail to understand even the most cooperative user.

To conclude this brief discussion, current spoken language dialogue systems design involves elaboration and further specification of the Gricean cooperativity maxims as far as both system and user communication is concerned, and requires additional constraints on user cooperativity which have no counterpart in human-human dialogue. In fact, the notion of cooperativity becomes asymmetrical in this context. Whereas system cooperativity is a more elaborated form of Gricean cooperativity, user cooperativity has to obey constraints which have no counterpart in elaborated Gricean cooperativity, such as the requirement to answer the system's questions briefly and one at a time. These findings illustrate a general point of the current report, namely that designers of human-computer communication systems can learn from, but also have to go beyond existing human-human dialogue and discourse theory in developing a dedicated dialogue theory which can support the achievement of their objectives.

References

[Bernsen 1993a] Bernsen, N.O.: The structure of the design space. In Byerley, P.F., Barnard, P.J. and May, J. (Eds.): *Computers, Communication and Usability: Design issues, research and methods for integrated services.* Amsterdam, North-Holland, 1993, 221-244.

[Bernsen 1993b] Bernsen, N.O.: The Structure of the Design Space. CO-SITUE Illustrated by a Study in Early Artifact Design. *CCI Working Papers in Cognitive Science and HCI*, WPCS-93-5, 1993.

[Bernsen 1993c] Bernsen, N.O.: Types of User Problems in Design. A Study of Knowledge Acquisition Using the Wizard of Oz. *Esprit Basic Research project AMODEUS Working Paper* UM/WP 14, 1993. In Deliverable D2: *Extending the User Modelling Techniques.* June 1993.

[Bilange 1991] Bilange, E.: A Task Independent Oral Dialogue Model. In *European ACL*, Berlin, April, 1991.

[Dybkjær et al. 1993] Dybkjær, H., Bernsen, N.O. and Dybkjær, L.: Wizard-of-Oz and the Trade-off between Naturalness and Recogniser Constraints. In *Proceedings of Eurospeech '93*, Berlin 21-23 September, pp. 947-950, 1993.

[Dybkjær and Dybkjær 1994] Dybkjær, H. and Dybkjær, L.: Representation and Implementation of Spoken Dialogues. *Report 6b, Spoken Language Dialogue Systems, CPK Aalborg University, CCS Roskilde University, CST University of Copenhagen.* To appear early 1994.

[Dybkjær and Dybkjær 1993] Dybkjær, L. and Dybkjær, H.: Wizard of Oz Experiments in the Development of a Dialogue Model for P1. *Report 3, Spoken Language Dialogue Systems, STC Aalborg University, CCS Roskilde University, CST University of Copenhagen.* February 1993.

[Eckert and McGlashan 1993] Eckert, W. and McGlashan, S.: Managing Spoken Dialogues for Information Services. In *Proceedings of Eurospeech '93*, Berlin 21-23 September, pp. 1653-1656, 1993.

[Grice 1975] Grice, P.: Logic and conversation. In P. Cole and J.L. Morgan (eds.): *Studies in Syntax* Vol. 3: *Speech Acts.* New York: Academic Press 1975.

[Grosz and Sidner 1986] Grosz, B.J. og Sidner, C.L.: Attention, intentions, and the structure of discourse. *Computational Linguistics* 12, 1986.

[Grosz and Sidner 1989] Grosz, B.J. og Sidner, C.L.: Plans for Discourse. In M. Cohen and M.E. Pollack, (eds.): *Intentions in Communication.* Cambridge MA: MIT Press 1989.

[Heisterkamp 1993] Heisterkamp, P.: Ambiguity and Uncertainty in Spoken Dialogue. In *Proceedings of Eurospeech '93*, Berlin 21-23 September, pp. 1657-1660, 1993.

[Kamp 1981] Kamp, J.A.W.: A theory of truth and semantic representation. In J.A.G. Groenendijk, T. Janssen, and M. Stokhof (eds.): *Formal Methods in the Study of Language.* Amsterdam: Mathematical Center Tracts 1981.

[Larsen et al. 1993] Larsen, L.B., Brøndsted, T., Dybkjær, H., Dybkjær, L. and Music, B.: Overall Specification and Architecture of P1. *Report 2, Spoken Language Dialogue Systems, STC Aalborg University, CCS Roskilde University, CST University of Copenhagen*, February 1993.

[Litman 1985] Litman, D.: Plan Recognition and Discourse Analysis: An Integrated Approach for Understanding Dialogues. *Technical Report TR 170*, University of Rochester, Rochester NY 1985.

[Mann and Thompson 1987] Mann, W.C. and Thompson, S.A.: Rhetorical Structure Theory: Description and Construction of Text Structures. In Kempen, G. (Ed.): *Natural Language Generation: New Results in Artificial Intelligence, Psychology, and Linguistics.* Dordrecht: Nijhoff, 85-95, 1987.

[Mann and Thompson 1988] Mann, W.C. and Thompson, S.A.: Rhetorical Structure Theory: Towards a Functional Theory of Text Organization. *Text,* 8, 3, 243-81, 1988.

[Nofsinger 1991] Nofsinger, R.E.: *Everyday Conversation*. Sage Publications, 1991.

[Peckham 1991] Peckham, J.: Speech Understanding and Dialogue over the Telephone: An Overview of Progress in the SUNDIAL Project. In *Proceedings of Eurospeech '91*, Genova, Italy, September, pp. 1469-1472, 1991.

[Peckham 1993] Peckham, J.: A New Generation of Spoken Dialogue Systems: Results and Lessons from the SUNDIAL Project. In *Proceedings of Eurospeech '93*, Berlin 21-23 September, pp. 33-40, 1993.

[Povlsen and Music 1994] Povlsen, C. and Music, B.: Definition and Specification of the Sublanguage for P1. *Report 4. Spoken Language Dialogue Systems, CPK Aalborg University, CCS Roskilde University, CST University of Copenhagen.* To appear early 1994.

[Searle 1969] Searle, J.R.: *Speech Acts*. Cambridge: Cambridge University Press 1969.

[Sperber and Wilson 1986] Sperber, D. and Wilson, D.: *Relevance. Communication and Cognition*. Oxford: Blackwell 1986.

[Smith 1991] Smith, R.W.: *A Computational Model of Expectation-Driven Mixed-Initiative Dialog Processing.* Ph.D. Thesis, Department of Computer Science, Duke University, Durham, NC 27706, USA, October, 1991.

[Zoltan-Ford 1991] Zoltan-Ford, E.: How to Get People to Say and Type what Computers can Understand. *International Journal of Man-Machine Studies* 34, 527-547 1991.

Project Reports

The following list is a preliminary list of project reports from the research programme *Spoken Language Dialogue Systems.* The final versions may have slightly different titles and authors.

1. Larsen, L.B., Brøndsted, T., Dybkjær, H., Dybkjær, L., Music, B.and Povlsen, C: State-of-the-art of Spoken Language Systems—A Survey. September 1992.
2. Larsen, L.B., Brøndsted, T., Dybkjær, H., Dybkjær, L. and Music, B.: Overall Specification and Architecture of P1. February 1993.
3. Dybkjær, L. and Dybkjær, H.: Wizard of Oz Experiments in the Development of a Dialogue Model for P1. February 1993.
4. Povlsen, C. and Music, B.: Definition and Specification of the Sublanguage for P1.
5. Brøndsted, T. and Larsen, L.B.: Representation of Acoustic and Linguistic Knowledge in Continuous Speech Recognition. January 1994.

6a. Bernsen, N.O., Dybkjær, L. and Dybkjær, H.: Task-Oriented Spoken Human-Computer Dialogue. February 1994.

6b Dybkjær, H. and Dybkjær, L.: Representation and Implementation of Spoken Dialogues.

7. Music, B., Offersgaard, L. and Christensen, D.: The Parsing Module NJAL.
8. Lindberg, B. and Kristiansen, J.: The Speech Recogniser.
9. Evaluation of the Flight Ticket Reservation System.
10. Bækgaard, A.: The ICM/DDL Platform.