How to obey the 7 commandments for spoken dialogue?

Emiel Krahmer, Jan Landsbergen, Xavier Pouteau
IPO, Center for Research on User-System Interaction
P.O.Box 513
NL-5600 MB, Eindhoven, The Netherlands
{krahmer/landsbrn/pouteau}@ipo.tue.nl

Abstract

We describe the design and implementation of the dialogue management module in a voice operated car-driver information system. The literature on designing 'good' user interfaces involving natural language dialogue in general and speech in particular is abundant with useful guidelines for actual development. We have tried to summarize these guidelines in 7 'meta-guidelines', or commandments. Even though state-of-the-art Speech Recognition modules perform well, speech recognition errors cannot be precluded. For the current application, the fact that the car is an acoustically hostile environment is an extra complication. This means that special attention should be paid to effective methods to compensate for speech recognition errors. Moreover, this should be done in a way which is not disturbing for the driver. In this paper, we show how these constraints influence the design and subsequent implementation of the Dialogue Manager module, and how the additional requirements fit in with the 7 commandments.

keywords: spoken dialogue management, error-prevention, error-recovery, design issues

1 Introduction

There are many good reasons why spoken language might be a main in- and output device for a user-interface. One of them is that in certain situations it is difficult for a user to operate a system in another way, because (s)he is involved in a task with heavy manual requirements. Consider the case of a car-driver: the current generation of driver information systems (usually involving HiFi equipment, but also route-guidance computers, traffic-messaging (RDS/TMC) and mobile telephone (GSM)) is getting more and more complex, and operating these devices is becoming a significant task as well. Since the driver's visual and gestural channels are heavily involved in the main, driving task, it seems worthwhile to study the possibilities of a spoken interface for such driver information systems, and this is the main objective of *VODIS*, a European project dedicated to the design and implementation of a vocal interface to an existing driver information system.

Even though the state of the art Speech Recognition (SR) modules perform well (see e.g., Cole *et al.* 1996), speech recognition errors cannot be precluded. For the current application, the fact that the car is a notorious acoustically hostile environment is an additional complication. This means that special attention should be paid to effective methods to compensate for SR errors. Moreover, this should be done in a way which is not disturbing for the driver. This is one of the central tasks of a Dialogue Manager module. In general, the Dialogue Manager module can be seen as an intermediate agent between user and application, helping the former in maintaining a good representation of the latter. Relevant literature points out that there is no general *theory* for the development of a Dialogue Manager (henceforth DM). On the other hand, a lot of *guidelines* for the development of 'good' vocal interfaces exist.

In this paper, we describe some of the methods used for the DM module in the VODIS project, with the focus on error-prevention and error-handling. A recurrent theme in our description of the DM module is the relation between the design and the many guidelines found in the literature. To facilitate the discussion, we have tried to summarize these guidelines into a limited number of 'meta-guidelines': the 7 commandments for spoken language dialogues (section 2). Most of these com-

mandments can be related to general recommendations about user-interfaces (as found in e.g., Shneiderman 1992:72-73 and Karis & Debroth 1991:578), but here the emphasis is on *spoken* user-interfaces. The 7 commandments may sound obvious and general, although hard to obey in real life. We contend that this is a basic property of commandments. Be that as it may, we feel that it is worthwhile to present these 7 commandments, if only to give the reader an impression of the kind of things that have to be kept in mind when designing and implementing a DM module. The 7 commandments are given in section 2.[1] In section 3 we describe the main generic methods used within the DM to compensate for speech errors in VODIS and in section 4 we briefly describe how they are implemented. Finally, in section 5 there is some discussion on the applicability of the commandments and the generalizability of the DM in VODIS.

2 The 7 commandments for spoken language dialogues

I. THY SYSTEM SHALL MAINTAIN CONSISTENCY

A system should assign the same response to the same input (Lea 1994: 26). However, one should balance consistency with commandment V (adaptability): be consistent but not rigid (cf. Grudin 1989), e.g., enable reduced dialogues (Leiser 1993:287).

II. THOU SHALT BE AWARE OF THE PROFOUND INFLUENCE OF BOTH CONTENT AND FORM OF PROMPTS

This commandment essentially says that the system should be a good dialogue partner. To achieve this, the system should first of all pay attention to the way prompts are formulated. They should be as brief as possible without being compendious; wordy prompts (*system*: "I heard you say ...") lead to confusion (Fraser 1994:137,Lea 1994:15). Consistency is also relevant here: use a consistent linguistic style (Fraser 1994:137).

Second, prompts should fit in with the ongoing dialogue. Thus, the system should ensure that, where possible, each prompt finishes with an explicit question or command; proceed with the discourse rather than looking back to verify the past (Fraser 1994:137, Lea 1994: 15).

Third, different kinds of prompts can be used to *mark* different contexts. E.g., different voices can be used as the auditive counterparts of different 'active windows' in a windows-based operating system. However, one should use such distinctions carefully and ensure that each voice serves an intuitively different purpose (Fraser 1994: 137, Lea 1994: 31, Leiser 1993: 287).

Fourth, when a speech-recognition-error occurs, re-prompt in such a way that the user receives extra guidance on how to behave in the desired way (Fraser 1994:137). E.g, repeat the attempt containing an error *once*, so that the user can recognize the error, and at the same time error-loops are avoided (Lea 1994:32).

III. THY SYSTEM SHALL BE EASY TO COMPREHEND

Put differently: the system should have a low threshold for actual usage. Use progressive disclosure of information. *Structure tasks into small pieces*, so that the user does not have to remember too many things at a given point (Lea 1994:28). Keep the user informed about the currently available options (Lea 1994: 28, Leiser 1993:287).

IV. THOU SHALT MAKE THY SYSTEM 'GOOF-PROOF', FOR TO ERR IS HUMAN, BUT TO FORGIVE DESIGN

This commandment, based on an old adage (cf. Lea 1994: 18, Hix & Hartson 1993), subsumes *error-prevention* (IV.a) and *error-handling* (iv.b).

Ad IV.a: keep the user informed about the current situation (Leiser 1993: 287). One way to achieve this is by providing a clear response after every spoken input from the user, so the user knows that the system received input and can determine which interpretation is assigned to the input (Lea 1994: 31). In general: use visual and/or auditory cues to indicate the current interaction context, and emphasize switches from one application to another (Leiser 1993: 287). Another means to avoid errors is to define phonetically distinct words or phrases for allowed inputs, and make 'erroneous' choices unavailable (compare the different shading of unavailable menu options or icons in a windows-based operating system) (Lea 1994:31). For potentially 'dangerous' or 'expensive' actions (i.e., undoing them is relatively costly/time-consuming), include a validation step . Such a validation strategy should not be used for 'harmless' actions; that would slow down the interaction unnecessarily.

Ad IV.b: If an error *does* occur, let the system take the blame (e.g., *system*: "I didn't understand your utterance."). Do not blame the user (thus not *system*: "What you did was illegal!"). Focus on *re-*

[1] This list of 7 commandments is primarily based on the guidelines found in Fraser (1994), Lea (1994), and Leiser (1993), the two first mentioned references sum up a lot of the relevant literature. Lea comes to a list of seven 'cardinal rules' that partially overlaps our 7 commandments. Leiser is specifically concerned with speech interfaces in the car.

covering the error. One important element is the presence of a vocal 'undo' command. If possible, allow correction of local errors: avoid the necessity to re-enter the entire command (Lea 1994: 32).

V. THY SYSTEM SHALL BE ADAPTABLE
Do not force interaction, rather make the user aware of currently available options on a take-it-or-leave-it basis (Leiser 1993: 286). Only interrupt the ongoing dialogue in 'urgent' situations, and justify the interruption. Distinguish novice and expert users, and adapt to their levels.Where possible guide the naive user, but also allow the expert user to initiate actions and use short-cuts. (Lea 1994:30). Support interruption and recovery: use the 'normal manners' for interrupting the user in his current activities, i.e., only interrupt in 'critical' or 'urgent' situations, and provide the user with a justification for the interruption. Also, reassure the user that the system is robust against sudden interruptions (e.g., by using synthesized speech; the user will feel less social urgency to respond when he or she is aware of the fact that the dialogue partner is a computer (Leiser 1993); contrast this with commandment VI).

VI. THY INTERFACE SHALL BE TRANSLUCENT
Allow inputs which perform several steps, or which allow jumping from one point to another (Lea 1994:30). Use natural speech output (as opposed to synthesized speech) for prompts, to avoid focus on the quality of the machine voice (Lea 1994: 25).

VII. THOU SHALT COGITATE BEFORE THOU COMMENCETH
Last but not least, the necessity of a design phase should not be underestimated, and this is where commandments I to VI are useful. Also, always keep the added value of speech in mind (Lea 1994:15).

3 On the design of the DM module

How to obey these 7 commandments when designing a DM module? As usual with commandments, some are conceptually clearer and easier to obey than others. The best way to follow commandments is to take them as a source of inspiration and not follow them to the letter. In fact, obeying all guidelines subsumed by the 7 commandments is effectively impossible, since —as the reader will have noticed— they contain some inconsistencies.

While living by all these commandments when designing a system to be used in 'normal situations' is effectively impossible, to obey them when designing for in-car systems might appear to be even more difficult. One reason for this is that the interaction with the system must never interfere with the user's primary task (the actual driving). Moreover, since the car is an acoustically hostile environment, the limits of speech recognition have to be taken special care of. In this section, we look in more detail at the design of the DM module within VODIS, with special attention to the specific conditions posed by the vehicle-context and the relation with the 7 commandments. In the section hereafter we discuss the actual implementation in more detail.

3.1 The VODIS project

The main objective of the VODIS project is to integrate and further develop the technologies which are required for the design and implementation of voice-based user-system interfaces. More concretely, the project aims at developing a vocal interface to an existing driver information system (namely the *Berlin RCM303A* of Robert Bosch GmbH), which integrates a tuner, an amplifier, a CD changer, a cassette player, a navigation computer and a GSM telephone. The vocal interface is speaker independent, and is developed for German and French. The project consists of two stages: for the first stage a basic command & control language is defined consisting of about 70 keywords, which essentially encompasses the functionalities of the current system (selecting a device: "navigation", "tuner", choosing a destination, making phone calls, etc.), as well as dialogue control keywords ("no", "OK", "abort", etc.). Experimental evaluations of the first prototype will be the input to design and development of the second prototype, which also aims at broadening the range of possible user's input by allowing spontaneously spoken database queries for the navigation task. The reader can visit `http://www.is.cs.cmu.edu/VODIS` for more details.

Figure 1 depicts the general architecture of the VODIS system. As said, the purpose is to design and implement a voice interface to the *Berlin* driver information system. A *controller module* provides a software interface to the Berlin system: it can modify the state of the Berlin system, and it can retrieve information from it. If the user wants to say something, (s)he can indicate this by pressing a button, located near the steering wheel (the *Push-To-Talk* (PTT) button). The result of a PTT push action is that the *speech recognition unit* is activated. The *DM module* fills the gap between the speech recognition and the controller. The DM can provide information to the user via Text-To-Speech (TTS) synthesis and via a small display.

This architecture can already be related to some commandments. The PTT button allows the user to take the initiative: interaction is not forced, the sys-

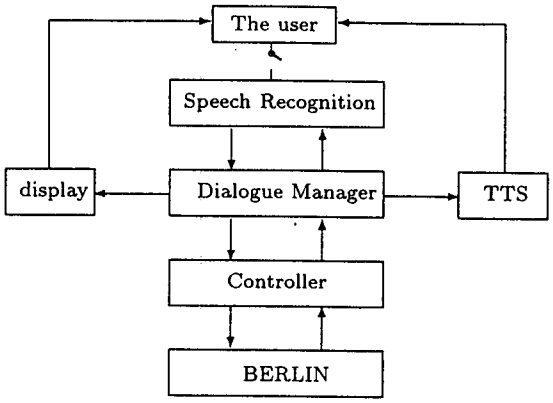

Figure 1: The VODIS architecture

tem just presents the user with his/her options, and by pressing the button the user requests attention of the speech recognition unit (cf. V). Additionally, TTS is used instead of pre-recorded natural speech (V/VI). This choice is more or less forced upon us, since there is no fixed vocabulary from the system's point of view. For instance, each user has a personal phone book stored in his GSM telephone, and to pronounce the names in this phone book the system can only use TTS.

3.2 Coping with the limitations of speech recognition

Good results from speech recognition is a *conditio sine qua non* for any spoken dialogue system. A system with bad results from speech recognition makes it impossible to satisfy many of the commandments (how could a user judge a system as flexible, consistent, adaptive, simple etc., if (s)he is often misunderstood by the system?).

Commandment IV stresses the importance of error-prevention (IV.a) and error-handling (IV.b). With regard to IV.a, several techniques are used within VODIS to prevent SR errors. First of all, a lot of attention is paid to optimizing the speech recognition unit 'off line', e.g., by noise reduction. Fortunately, the kind of noise in the car (engine rotation, tires, wind, etc.) is rather specific, and highly correlated to the driving speed, which is available all the time, which means that distortion can be compensated effectively. Moreover, the recognition unit is trained on the basic command and control language developed for the first phase of the project. A third way to optimize speech recognition is based on the fact that not all the keywords need to be available all the time. Since these keywords embrace the functionalities of the original Berlin system, they are partitioned in a more or less comparable way (thus, when the interaction is dealing with HiFi, the user cannot enter a destination for route-guidance). This makes it possible to partition the language by means of a number of *sub-grammars* (roughly speaking: there is a default-set of always active keywords, and each mode is associated with its own grammar, thus one could speak of a HiFi-subgrammar, a navigation-subgrammar etc.). The DM module decides which sub-grammar(s) should be active at any point in the dialogue, and sends this information to the speech recognition unit. As a consequence, the branching factor (= the number of available keywords at a given point) is always significantly *less* than the total number of key-words, which further decreases the chance of speech recognition errors.[2]

Nevertheless, SR errors cannot be precluded. The lowest error rate for speaker independent recognition achieved up to now on a task with a perplexity comparable to the one in VODIS is around 4% (Cole *et al.* 1996). And it is unlikely that recognition *in the car* will lead to *better* results. In other words: recognition errors will occur and this means that a method has to be developed to *handle* them. Each time the user utters something, the SR unit sends an n-best list of results to the DM. When the top element of this list is different from the user's actual utterance, we are facing a SR error. In general, the system cannot decide whether the first candidate of the list is:

1. the right candidate (as it will be in most cases),
2. an error due to confusion within the SR unit, or
3. an error due to the user, e.g., because a phrase was uttered outside the currently active vocabulary.

The only way to detect and solve an error is via

[2] A disadvantage of this partitioning is that there is a certain risk of the user uttering a keyword which does not correspond to the current state of the system, and since the speech recognition unit will not be able to recognize the user's utterance in that case only a relatively non-specific warning can be given (e.g, *system*: "The system cannot interpret your utterance."). Thus, this choice might lead to a reduction in 'user-friendliness' of the system. However: as noted above, good results of speech recognition is the basic requirement for a voice interface. Thus, the actual partitioning is a compromise between commandment IV on the one hand, and commandments I, III, V and VI on the other. Notice that this compromise puts extra emphasis on the marking of the current interaction context (cf. III), since a user which is well aware of the current state of the system is less likely to perform an input which is not allowed at that point.

a *validation process* (more of which below). When the first candidate of speech recognition is rejected by the user, the system has to initiate a recover strategy. It would be a bad strategy to systematically request a repetition from the user, as users are known to vary their pronunciation during subsequent attempts (volume, pitch, rate) as they would do when a human dialogue partner made a 'speech recognition' error, which has the undesired side effect of deteriorating speech recognition results. These two considerations imply that the handling of speech recognition results by the DM should be a *system controlled* strategy, which is applied to all results given by the speech recognition. Figure 2 shows a strategy developed in VODIS for that purpose.

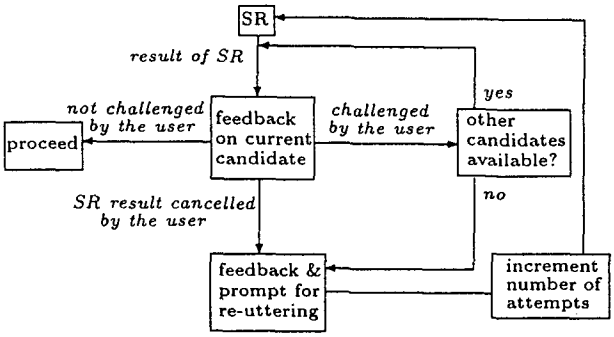

Figure 2: The handling of SR results

Let us illustrate this view diagram via an example. One place where SR errors might arise is in the recognition of names in the user's personal phone book. Suppose that the user's phone book contains two nicknames: "Phil" and "Bill". Now the user utters "Call Bill". The SR unit returns an ordered tuple of results to the DM: ⟨ "Call Phil", "Call Bill" ⟩. Thus, "Call Phil" is assigned a higher *confidence score* than the designated, second candidate "Call Bill". The DM now proposes the first candidate of speech recognition via a validation feedback, e.g., the system says "Call Phil?". At this stage, the user can do three things:

1. cancel the SR results,
2. challenge the first candidate, or
3. accept it.

In the current example, the user can be expected to go for the second option. Then the DM proceeds with the the next candidate ("Call Bill?"), which corresponds with the actual utterance. Again, the user can do three things, but now we may assume that the user will not challenge this proposed candidate, as it corresponds with the actual input from the user. The advantage of such a routine is that it applies to all inputs from the user in a uniform way, and does not put a too heavy burden on the user's attention. Naturally, the user has to 'know' what is expected from him, and this puts high demands on feedback and prompt design.

Summarizing, the basic mechanism sketched in figure 2 applies to every spoken input of the user in the same way, which complies with commandment I (be consistent). Whenever an error occurs, the error-handling part of commandment IV is obeyed as well: no blame is assigned, the focus is on recovering the error and there is an undo option ("abort").

3.3 Feedback and prompt design

In general, feedback aims at helping the user in keeping a good mental representation of the system (commandments II-IV), and a good representation generally increases the efficiency of the communication (e.g., the chances of out-of-vocabulary input are reduced). The DM can give feedback to the user via two modalities: *sound* and *vision*. In general, it is difficult to decide which modality should be chosen to present a given piece of information (witness e.g., Kariagiannides *et al.* 1995). However, two practical guidelines apply for VODIS:

1. Since a relatively short visual message might be missed (vision is employed for driving), essential information should at least (see 2.) be conveyed via sound,

2. Since speech is transient, information which the user may need to consult more than once should be available via vision.

A central ingredient of the procedure which handles user's inputs, sketched above, is the system's validation feedback on the current SR candidate. Leiser (1993:276) mentions two extremes regarding to validation: 1. assume that no errors have occurred, and leave it to the user to recover from any inopportune change of state as a result, and 2. always ask the user to explicitly confirm. The first alternative ignores error-handling (IV.b) and is obviously in conflict with the underlying philosophy of the handling of user's input. The second alternative, however, disobeys commandments II (looking back too much) and IV (forcing unnecessary validation), and both violate V (by not being adaptive). An adequate balance between both strategies, together with implicit validation, would greatly improve the situation. This is the strategy chosen in VODIS. Suppose, for example, that the n-best list of SR results contains "radio" as the first candidate.

This candidate is presented for validation to the user via a feedback message which tells the user that the system will switch to the radio and start playing the last selected radio station, e.g., "Switching to radio station BBC 1". Once this message has been synthesized, the user can do various things. The user can challenge the prompt by explicitly saying "no" or "abort". But the user can also validate it, either *explicitly* by saying "yes" or "OK", or *implicitly* by keeping silent ('those who don't speak, agree'), or by proceeding with the discourse via the utterance of a new command (e.g., "play KISS FM").

For this approach to work, the feedback messages have to meet certain criteria. It is well known that people are not only sensitive to the content of a message but also to the way it is sent.[3] A syntactically marked yes/no question ("Do you want to switch to navigation mode?") or a clear question contour ('high and rising', in the notation of Pierrehumbert and Hirschberg (1990): H* H H%) will cause the user to feel forced to *explicitly* confirm or reject. This indicates why feedback messages should be phrased in a consistent style (I and II). Sometimes, it may be useful to violate these commandments, most notably in non-standard situations, e.g., after an error has been detected. Notice that in such cases, 're-packaging' of the message serves a purpose: the user is provided with extra clues which are significant in that they provide additional information which may help the user in updating his model of the system. Thus: prompts should be short and to the point, and violations of this principle should serve a purpose.[4]

4 On the implementation of the DM

Our ultimate objective is the development of a 'good' DM module as part of the VODIS system, and we believe that designing a dialogue manager which obeys the 7 commandments as far as possible is a first, indispensable step towards that objective. The second step, which is addressed in this section, is *implementing* a DM module based on this design, as part of the first VODIS prototype. Since this prototype will be tested by drivers in the car, it runs on a stand-alone machine. The DM module is a separate *block* in the VODIS architecture, which interacts with other blocks via intercommunication protocols. The DM is written in C++.

The DM receives messages from two sources: the controller (the software interface to the Berlin system) and the SR unit. Messages from the controller concern state-changes of the system. They can lead to an update of the state-representation of the system, or to an interruption (e.g., in the case of an incoming phone call). In the case of an interruption, the DM can support the interruption of the main thread of the dialogue and restore the previous context, employing stack mechanisms. In general, change of status information (as far as it is directly relevant for the user) is handled via specific *system initiated* routines.

The other, from the point of view of this paper, more interesting source of messages received by the DM are mostly the result of a *user initiative*: the user has said something. Whenever the user indicates that (s)he wants to say something by pressing the PTT button, the DM is notified of this PTT event and waits for the actual results of speech recognition. Figure 3 depicts the modules of the DM which are involved in the subsequent handling of the input from the user.

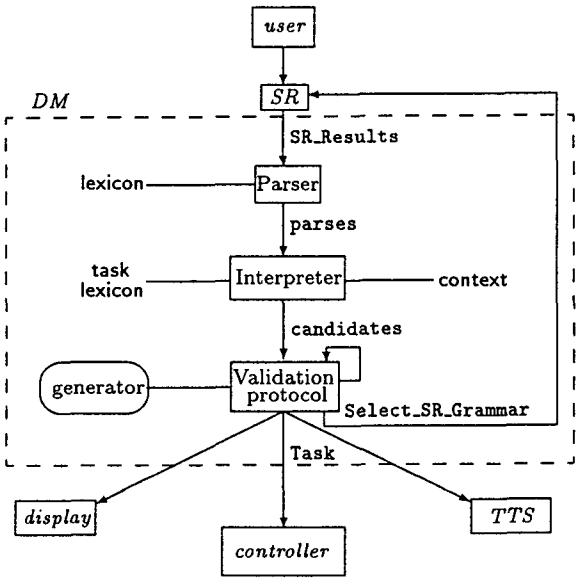

Figure 3: DM software architecture

The DM module has a modular structure. After the user's input has been analysed in the speech recognition unit, the DM receives a message consisting of a list `SR_results`, which contains the recognized phrases and their respective confidence scores. In the DM module, this list is unwrapped, and the phrases are parsed. Each recognized phrase is

[3]This relates to the notion of information packaging (cf. Chafe 1976). Chafe points out that the format of a message is only partially related with the content of the message, "[information packaging has] to do primarily with how the message is sent and only secondarily with the message itself, just as the packaging of toothpaste can affect sales in partial independence to the quality of the toothpaste inside".

[4]Put differently, what holds for human speakers (cf. Grice 1975), should hold for speaking systems as well.

mapped to a set of representations as found in the lexicon. If parsing one candidate results in a non-singleton set of interpretations, it is ambiguous. The Parser returns a new list, `parses`, to the Interpreter. In this module, the lexical items are mapped to tasks (found in the task lexicon) and related to a context, containing information about the current situation (including application and dialogue). We follow the common treatment of resolving ambiguities using this contextual information.[5] The result of this process is a list `candidates`, the first element of which consists of the task representation of the first disambiguated SR_result. This is the first candidate which is proposed for validation to the user (via TTS and/or the display, depending on the kind of message); an implementation of the validation protocol given in figure 2. The feedback messages are formulated by a *generator* module. In most cases, the first candidate will be the right one (see discussion in section 3). If a proposed candidate is (explicitly or implicitly) accepted by the user, the DM sends a message (containing the validated `Task`) to the control unit requesting modification of the status of the Berlin system according to the user's wishes. Also, the DM sends a message to the speech recognition (using the `Select_SR_Grammar` function) to activate a new set of *sub-grammars* corresponding to the new state of the system and the ongoing dialogue.

5 Discussion: future work

In this discussion section we want to address three issues. The first is evaluation, the second concerns the generalizability of the methods described in this paper, the third the applicability of the 7 commandments.

5.1 Evaluation

The DM module described in this paper will be part of the first VODIS prototype, to be completed in fall 1997. As mentioned, this first prototype will be extensively evaluated by users. For this purpose, the vocal interface to the Berlin system will be placed in a car, and evaluated by French and German drivers, in Paris and Karlsruhe respectively. During the evaluation, attention will be paid to (*i*) the speech recognition performance, and (*ii*) the user-system interface, with the emphasis on security, safety, acceptability and effectiveness. The results of these experiments will constitute the input for the development of the *second* prototype, which also aims at broadening the range of the possible user's input by allowing more 'natural language like' database queries for the navigation task. This raises the question whether the DM methods described in this paper are transmissible to the second prototype.

5.2 How generalizable are the DM methods?

The primary aim of the first VODIS prototype is to build a simple, but robust spoken language system which can be used effectively in the car. The DM methods described in this paper are also intended to be simple, but robust, and that is why the prevention and handling of speech errors plays a central role. Of course, the step from a limited command and control language to more 'spontaneous' speech is a big one, and is likely to affect the DM. However, we would like to claim that the basic DM methodology can remain largely unchanged. To backup this claim, let us first describe the (planned) second prototype in somewhat more detail. The main difference in architecture between the two prototypes is that in the first one the results from the SR unit are directly fed to the DM module, while in the second one the two modules are connected via a *semantic parser* (see e.g., Ward 1994). This parser is trained on a language model, and differs from classical parsers in that it does not (only) use 'syntactic' information, but also domain dependent 'semantic' information. That is: it does not look in the input for NPs and APs, but rather for, say, 'destination phrases' and 'arrival time phrases'. It tries to extract as much information as possible from the received input, simply ignoring input it cannot parse (e.g., interjections and false starts). This entails that the DM will not be confronted with an n best list of recognized key-words, but with a more complex structure; a kind of list of parses annotated with confidence scores from both the SR and the semantic parser. Again, the DM can validate the first candidate in the way described above. Suppose the first candidate of the input list received by the DM indicates that the user wants to go to Bistro *Le pot de terre* in Paris. The DM can put up this first candidate for validation. Essentially, the user will have the same options as for the keyboard based communication (Figure 2), except that (s)he now will have the additional opportunity of clarifying his challenge (user: "No, I don't want to go to Bistro *Le pot de fer*, but to Bistro *Le pot de terre*").[6] On the basis

[5] Of course, the limited control-and-command language will not give rise to many ambiguities. The situation is expected to change when the range of user's inputs is widened at a later stage.

[6] The recognition of proper names (and in particular city names) is a major problem to be tackled for the second demonstrator. For example, the German navigation computer knows 30.000 city names. Plain recog-

of such corrections the DM can perform an update on the original list of annotated inputs. Of course, this is all speculative, but it indicates that the DM methods to deal with SR results presented above can be used for the second prototype as well.

5.3 How applicable are the 7 commandments?

In the Introduction we noted that the literature does *not* contain a general theory for the development of a DM module, while it *does* contain a lot of practical guidelines. On the one hand, this can be seen as an indication that this is still a relatively immature area of research. On the other hand, it also indicates that the characteristics of a Dialogue Manager are largely determined by the kind of application. Of course, the many guidelines found in the literature (summarized in our 7 commandments) are potentially very useful when one designs a DM module. However, we also saw that obeying all commandments is effectively impossible, since some of the guidelines they subsume are inconsistent with each other. This raises an obvious question: how applicable are the 7 commandments? Or more in general: what are useful guidelines? The evaluation of the first VODIS prototype may give some indications in this respect. For example, it might turn out that users do *not* like to proceed with the discourse, but would prefer explicit validation of each input. In our opinion, it would be very interesting to find out which specific guidelines are useful in which specific situations. However, this research program will not be carried out within VODIS.

6 Conclusions

We have described some aspects of the design of the DM module within the VODIS project, with special attention to the methods the DM can employ to compensate for the limitations of speech recognition. Where possible, we have related our proposals to guidelines found in the literature, summarized in our 7 commandments for spoken dialogue. Of course, the ultimate objective is the development of a "good" dialogue manager module as part of the VODIS project, and we believe that designing a dialogue manager which obeys the 7 commandments as far as possible is an indispensable step towards that objective. The implementation, briefly described in section 4, will be part of the first VODIS prototype, which will be evaluated by users. The results of these experiments will be the input for the development of the second prototype.

nition (user: *Ich möchte nach Himmelpforten fahren*) on such a list will cause problems, for which alternative strategies have to be chosen.

Acknowledgments

This work was funded by the Language Engineering/Telematics Applications Program, project LE-1 2277 (VODIS). Thanks are due to Maddy Janse, Kees van Deemter and Gert Veldhuijzen van Zanten for comments on an earlier version of this paper.

References

Chafe, W. 1976 Givenness, contrastiveness, definiteness, subjects, topics and point of view. In: C. Li (ed.), *Subject and topic*, 25-55, New York, Academic Press.

Cole, R. *et al.* 1996 *Survey of the state of the art in human language technology.* to appear.[7]

Fraser, N., 1994, Interactive dialogue, In: *EAGLES spoken language systems* (draft), ESPRIT.

Grice, H. P. 1975, Logic and conversation. In: *Syntax and semantics 3: speech acts*, P. Cole (ed.) Academic Press, New York

Grudin, J. 1989, The case against user interface consistency. In: *Communications of the ACM*, **32** (10): 1164-1173

Hix, D. & H. Hartson, 1993, *Developing user interfaces*, Wiley, New York

Kariagiannides, C., *et al.* 1995 Media/modalities in intelligent multimedia interfaces. In: J. Lee (ed.) *First international workshop on intelligence and multimodality in multimedia interfaces*, HCRC

Karis, D. & Dobroth, K. 1991, Automating services with speech recognition over the public switched telephone network: human factors considerations. In: *IEEE J. Selected Areas in Commun.*, **9** (4), 574-585

Lea, W., 1994, Developing usable voice interfaces. In: *Journal of the American Voice I/O Society*, **16**

Leiser, R., 1993, Driver-vehicle interface: dialogue design for voice input. In: A. Parkes & S. Franzen (eds), *Driving future vehicles*, Taylor & Francis, 275-294

Pierrrehumbert, J. & J. Hirschberg 1990 The meaning of intonational contours in the interpretation of discourse. In: P. Cohen *et al.* (eds.), *Intentions in communication*, 271-311, MIT Press

Shneiderman, B., 1987, *Designing the user interface - strategies for effective human-computer interaction*, Second edition, Addison Wesley, Reading, MA.

Ward, W., 1994, Extracting information in spontaneous speech. In: *Proceedings of ICSLP 94*, Yokohama

[7] `http://www.cse.ogi.edu/CSLU/HLTsurvey/` for the web-version.