Deliverable D1.5

Working Paper on Dialogue Management Current Practice

May 1998

Esprit Long-Term Research Concerted Action No. 24823

Spoken Language Dialogue Systems and Components: Best practice in development and evaluation.

DISC

TITLE	**D1.5 Working paper on dialogue management current practice.**
PROJECT	DISC (Esprit Long-Term Research Concerted Action No. 24823)
EDITORS	Niels Ole Bernsen and Laila Dybkjær, The Maersk Institute
AUTHORS	Niels Ole Bernsen, Laila Dybkjær (MIP) Uli Heid, Jan van Kuppevelt (IMS) Niels Dahlbäck, Patrik Elmberg, Arne Jönsson (Linköping)
ISSUE DATE	25 May 1998
DOCUMENT ID	wp1d5
VERSION	0.2
STATUS	Draft
NO OF PAGES	195
WP NUMBER	1
LOCATION	deliverables/MIP-DM-25.5.98.ORIG
KEYWORDS	WP1, dialogue management, current practice

Document Evolution

Version	Date	Status	Notes
0.1	10/05/98	Draft	First draft published for review from partners.
0.2	25/05/98	Draft	First draft published for review.

Working Paper on Dialogue Management Current Practice

Abstract

This paper summarises key aspects of current practice in dialogue manager development and evaluation for commercial and research spoken dialogue systems (SLDSs). Dialogue management is arguably the core functionality of SLDSs because the dialogue manager tends to have the controlling function of the system as a whole and maintains a model of the evolving dialogue context. The paper is based on in-depth analyses of the following dialogue manager exemplars: the Daimler-Benz Dialogue Manager, the dialogue manager in the Danish Dialogue System, the dialogue manager in Railtel/ARISE, the dialogue manager in Verbmobil, and the dialogue manager in Waxholm. These analyses are presented in the Appendix. The rich diversity exhibited by those five exemplars has served as the basis for the two main synthetic contributions of the paper: the introduction to dialogue management current practice in Chapter 2 and the introduction to dialogue manager development and evaluation in Chapter 3. Both chapters have been iterated over the exemplar analyses to produce what the authors hope are views on how to design, develop and evaluate appropriate dialogue managers, that are more general than those which are currently available. Current views on dialogue management tend to be heavily biased towards a particular SLDS, a particular interactive task, and a particular dialogue management solution for that task. The reason is that few, if any, dialogue manager developers have extensive hands-on experience from developing different dialogue managers for a wider range of SLDSs and interactive tasks. Hands-on experience remains a major source of knowledge about dialogue management, given the fact that the generalisations that are required in the field cannot yet be found in the literature. If considered as general truths, those views may induce sub-optimal solutions to the development of SLDSs and dialogue managers which, by their nature, require a different approach. On this background, the authors believe that Chapters 2 and 3 contain as yet unexploited material for establishing much needed generalisations concerning best practice in the development and evaluation of dialogue managers. In the second year of DISC, the paper will serve as a basis for developing a draft best practice methodology for dialogue manager development and evaluation.

The intended readership of the paper is specialists in SLDSs in general and in dialogue management in particular. The "packaging" for different audiences of the evolving DISC understanding of best practice in SLDSs development and evaluation will be done later in the project. The occasional advice provided to the dialogue manager developer in this paper, is intended for industrial and academic developers aiming to produce systems for real or "simulated" end-users, following current dialogue engineering practice. The advice is not intended for developers who, for experimental or other reasons, aim to develop systems that are more complex than needed for the task and the domain.

Contents

1. Introduction

Dialogue management is arguably the core functionality of spoken language dialogue systems (SLDSs). Figure 1 helps explain why this is the case. The figure shows the logical architecture of SLDSs as organised in a series of layers called performance, speech, language, control and context, respectively. In some order which is dependent on the task and the SLDS at hand, a user interacting with an SLDS produces speech input (speech layer) and receives speech output (speech layer) from the system. In addition, the regular user might figure out more abstract properties of the system's behaviour, such as how co-operative it is during dialogue, the distribution of user and system dialogue initiative, or how the system attempts to influence the user's dialogue behaviour through its choice of words and in other ways (performance layer). The rest of the system's workings are hidden from inspection by the user. Figure 1 classifies these workings in terms of a series of headers, such as 'user utterances' or 'domain model', and elements subsumed by each header. The elements are high-level references to the functionality that may be present in today's SLDSs. Most of today's SLDSs do not have all of the functionalities referred to in Figure 1, but all SLDSs have some of the functionalities.

Dialogue management is primarily located in the control and context layers in Figure 1. When the user inputs an utterance to the system, the dialogue manager may receive a representation of the meaning of what the user said from the speech and language layers. It is the task of the dialogue manager to handle that representation properly, eventually producing an appropriate output utterance to the user. Depending on the complexity of the task which the system helps the user solve, the dialogue manager may have to make use of most or all of the elements listed in the control and context layers in Figure 1. For instance, to determine if the user has provided a new piece of information which the system needs in order to complete the task, the dialogue manager may have to check the history of sub-tasks performed so far (context layer, task history). Or, to decide if shortcuts in the dialogue can be made because this particular user does not need a certain piece of information or advice, the dialogue manager would check if the user belongs to the class of experienced users (context layer, user group).

When addressing the state-of-the-art in dialogue management, it is sometimes difficult to distinguish clearly between issues to do with dialogue management *functionality* and dialogue management *usability*. Representing the core functionality of SLDSs, dialogue management cannot be designed and developed without keeping the intended users in mind. Still, the present paper primarily addresses the issues of functionality to be considered by the dialogue management developer. For the usability issues, the reader is referred to Williams et al. 1998.

This paper only addresses issues to do with the development of dialogue managers for *task-oriented* SLDSs. These SLDSs are being developed to help the user solve a particular task, or several tasks, and they standardly do so in a dialogue with the users in which speech input and output plays a major part. The alternative to task-oriented SLDSs is *conversational* SLDSs which aim to perform intelligent conversation with the user more generally, for instance in order to pass some Turing test. Existing experimental systems of this kind are not being considered in what follows.

The issue of dialogue management can be addressed at several different levels of abstraction. In this paper, we shall mostly ignore low-level implementation issues such as programming

languages, hardware platforms, software platforms, generic software architecture, database formats, query languages, data structures which can be used in the different system modules, etc., except when these issues come up during as part of particular grid of life-cycle analyses (these terms are explained below). Generic software architectures for dialogue management are still at a very early stage, and low-level implementation issues can be dealt with in different ways with little to distinguish between these in terms of efficiency, adequacy etc. Good development tools would appear to be of more relevance at this point. Tools will be discussed, and the following text aims to provide the basics needed to interpret specific dialogue manager architectures and flow descriptions and use them as a source of information for building SLDSs in general and dialogue managers in particular.

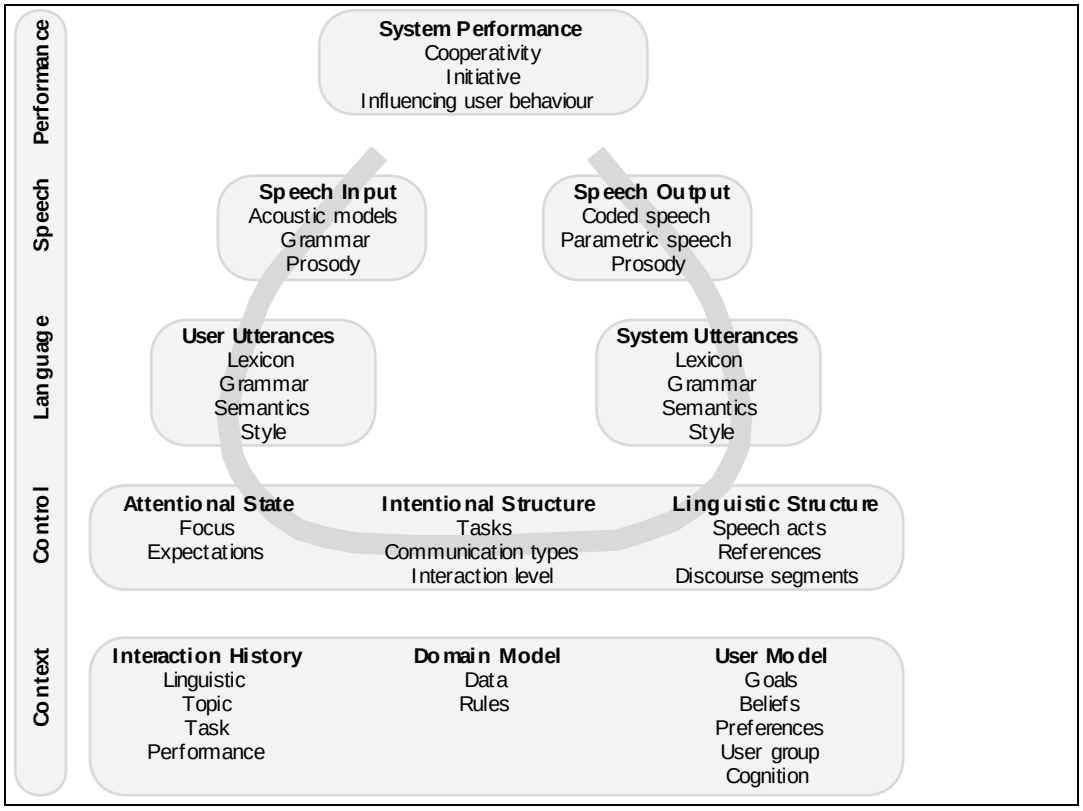


Figure 1. Elements of an interactive speech theory. The grey band and grey boxes reflect the logical architecture of SLDSs.

Dialogue management is but one of the six different aspects of SLDSs investigated in DISC. The five other aspects are: speech recognition, speech synthesis, language understanding and generation, human factors and systems integration. Each of these aspects are being investigated from two different perspectives, called the 'grid' and the 'life-cycle', respectively. Accordingly, we speak of 'grid properties' and 'life-cycle properties' of SLDSs and their components. *Grid properties* are factual properties of SLDSs and components as well as interrelationships among such factual properties. *Life-cycle properties* are characteristics of the development process of an SLDS or component. This paper represents a first synthesis of a series of analyses which has been performed in DISC of the grid and life-cycle properties of several dialogue managers from research and industry. The analyses themselves are presented in the Appendix.

Chapters 2 and 3 have a dual job to do. Firstly, they should make intelligible the analyses of

individual dialogue managers presented in the Appendix. In other words, Chapters 2 and 3 serve to explain the grid and life-cycle properties attributed to the dialogue managers presented in the Appendix. Second, Chapters 2 and 3 serve as an overview of current practice in dialogue management development and evaluation as observed in the analyses of the individual dialogue managers presented in the Appendix. It seems likely that the members of the DISC consortium who have been looking at dialogue management, have had a unique opportunity to do in-depth analysis of a variety of dialogue managers. Dialogue management theory is still relatively uncharted territory. The material presented and synthesised in this paper, if properly analysed, might form the basis for sketching a first general theory of dialogue management. During the preparation of the paper, it has become clear that its authors have sometimes different perspectives and theoretical views on the subject of dialogue management. In most cases, the different perspectives can be accounted for by reference to the fact that the authors have different mental models of dialogue managers based on their different, and always limited, experience from actual dialogue manager development. The different theoretical views can be seen to be due to, among other things, different views on the relevance of human-human spoken dialogue theory to the current state-of-the-art in dialogue manager design and development. Our inter-author discussions are far from concluded yet, fortunately, because these discussions hold the promise of making additional progress towards a joint and consolidated view of dialogue management.

2. Designing an Appropriate Dialogue Manager

A dialogue manager should be designed based on careful consideration of its task(s) and users. What actually happens is that the designer (or developer) designs a system based on both *design-time* and *run-time* considerations. Some design decisions only affect design-time work whereas many others affect run-time operation as well. The system's task, for instance, is fixed at design-time and cannot be modified at run-time. On the other hand, what the user actually inputs at some point, cannot be predicted at design-time and must be handled at run-time. The distinction between design-time and run-time considerations is inherent to the following account and the reader will no doubt be able to decide when a particular comment applies to design-time only or to run-time as well.

Goal of the Dialogue Manager

2.1 Efficient task performance

The central goal of an SLDS is to enable the interactive task to be done efficiently and with a maximum of usability. As we will not be considering usability aspects in this paper (see (Williams et al. 1998), from the point of view of the present paper this goal reduces to efficient task performance. Efficient task performance means that the system always performs appropriate processing of the user's input and always provides appropriate output at a given stage during the dialogue. This are the - admittedly ideal - goals that SLDS developers should strive for, and these goal subsume a number of sub-goals which are common in discussions of SLDSs and their components, such as robustness, naturalness etc. (see Chapter 3).

System Varieties

2.2 Multimodal systems including speech

Appropriate output is not always *spoken* output. For some tasks, the system's output may be system actions which the user can perceive, such as connecting the user with the person the user asked to speak to. Some tasks require the system to output information which is most effectively presented in, e.g., textual or pictorial form. Long lists of flight connections, for instance, are much more efficiently conveyed by text, and it is well-known that the contents of many pictures are virtually impossible to render through verbal means. A bicycle repair guide, for instance, is not much worth to the novice without ample pictorial illustration. In such cases, the SLDS needs to be enhanced through other modalities of (output) information presentation, such as text or images presented on a screen. This is the case in the Waxholm system which also uses an animated graphical speaking face whose lip movements help the user disambiguate the system's synthetic speech and whose eye movements help the user focus on newly added information on the screen. Through its user speaks/interrupts input button, Waxholm also illustrates the fact that SLDSs do not need to only take speech as input.

More generally speaking, there is an important question here about when to use speech input

and/or speech output for a particular application, when not to use speech, and when to use speech in combination with which other input/output modalities of information representation and exchange. The solution to this question about the functionalities, or roles, of modalities in particular cases does not only depend on the task but on many other parameters as well. A tool is being developed in DISC to support the decision on when (not) to use speech for particular applications (Bernsen and Dybkjær, to appear).

2.3 Multilingual systems

Nor is appropriate output always spoken output *in the same language* as the spoken input. Spoken translation systems, for instance, such as Verbmobil, translate spoken input in one language into spoken output in another language. These systems are basically different from standard SLDSs in that they do not conduct any dialogue with the user about the application domain: rather, they mediate dialogue between users who speak different languages. At most, spoken translation systems conduct meta-communication dialogues with their users (see 2.15). Still, such systems share many issues of dialogue management with standard SLDSs.

Another obvious possibility is that an SLDS accepts spoken input in different languages for the same task(s) and responds in the language in which it is being addressed.

2.4 One or several tasks?

In any case, what is appropriate output primarily depends on the interactive task for which the system is being built. In what follows we shall focus on the design of an SLDS for a single task. Already today, however, systems are being built to handle several more or less independent tasks, such as consulting one's diary and answering email over the phone. These applications increase an already existing need for task and domain independent dialogue managers. Task and domain independent dialogue managers would help facilitate the rapid prototyping of SLDSs for new tasks and new domains. We shall return to this issue towards the end of this chapter (Section 24).

Input Speech and Language

2.5 Are the speech and language layers OK?

Suppose that we have identified a certain task, T1, for which we consider building an SLDS. Suppose, in addition, that T1 is already known not to generate problems which cannot be solved at the speech and language layers (cf. Figure 1). In other words, our speech recogniser can be expected to possess the necessary robustness for the application, the vocabulary will not be infeasibly large, we are able to develop the grammar and parser required, language generation and speech generation of sufficient quality can be developed, etc.

2.6 Do the speech and language layers need support from the dialogue manager?

Things in this world often do not come for free. It may be that conditions have to be imposed on the dialogue manager in order to guarantee feasibility in the speech and language layers. The feasibility conditions may derive from, e.g., the need for real-time performance or for achieving a sufficient recognition rate in a large-vocabulary system. Such conditions could be that the dialogue manager must:

- constrain the search space of the speech recogniser by constraining the set of words which are likely to occur in the next utterance (sub-vocabulary prediction);

- constrain the search space of the keyword or phrase spotting component by delimiting its search space to the most probable keywords or phrases (keyword prediction);
- constrain the search space of the syntactic analysis component by narrowing the set of applicable grammar rules to a specific and probable subgrammar (subgrammar prediction);

- constrain the search space of the parser by narrowing the set of semantically meaningful units it should be working with (semantic prediction);
- constrain the search space of the semantic analysis component by, e.g., delimiting the set of possible antecedents in cross-sentence anaphora resolution;
- control the prosody of the spoken output based on control layer information about the message to be produced, such as speech act information;
- control the lexical variation of dialogue expressions to be produced by the language generation component;
- control the grammar of the dialogue utterances to be produced by the language generation component;
- control the style of the dialogue utterances to be produced by the language generation component.

In other words, the dialogue manager may have to actively support the processing done in the speech and language layers. It is worth noting, however, that many existing systems use no dialogue manager support for the speech and language layers. The first four of the above conditions concern input prediction. The fifth condition concerns input language processing control. The last four conditions concern output control. *Input prediction* is possible, i.a., if the task has some structure to it (see 2.8 and 2.11). *Input language processing control* becomes increasingly important as the user's input utterances grow in length as well as in lexical, grammatical and linguistic context-determined complexity. *Output control* is an important means of controlling the user's dialogue behaviour (see 2.9). In the future, we will see many more forms of input prediction, input language processing control and output control than those listed above.

Whether or not conditions such as the above obtain are likely to depend primarily on (a) real-time requirements on the application (see 2.7) and (b) task complexity (see 2.8).

2.7 Real-time requirements

It may be assumed that most or all SLDSs need to work in real-time or close-to-real-time, for such is the nature of spoken dialogue. In some cases of spoken human-human dialogue, we do tolerate waiting for a response from our dialogue partner, especially when the partner has to search for information or do complex calculations or inferences before responding. Users may be expected to tolerate the same from machines but, generally speaking, close-to-real-time operation is a very important goal in SLDS design. So let us look at the other factor, task complexity.

Getting the User's Meaning

2.8 Task complexity

Ideally, we would like to develop SLDSs which simply let the users speak their minds and then

do whatever is necessary to complete the task. For some tasks, this is clearly feasible. Suppose, for instance, that our T1 has to ask the user if the user wants to accept a collect call and, depending on the user's answer, simply routes or does not route the call to the user. In this case, the risk of letting the users speak their minds probably is as small as it ever gets. However, suppose that T1 is about ordering VIP dinner arrangements at a large restaurant including date, time, duration, choice of rooms, many-course meals, special diets, wines, flowers, timing for speeches, entertainment, seating arrangements, payment arrangements etc. It is easy to imagine that some users will have quite a bit to say about that. They may have so much to say, in fact, that even the dialogue manager support to the speech and language layers described in 2.6 above will not be sufficient to guarantee that the system will always be able to provide appropriate output. Task complexity, therefore, probably is the single most important factor to consider for the dialogue management developer.

At this point, task complexity cannot be measured in any objective way. Still, several guidelines may be useful, such as:

(a) *Volume of information:* how many pieces of information need to be exchanged between the user and the system to complete the task? In many cases, one or two pieces of information will suffice. The collect call SLDS above is a case in point. The Operetta telephone directory assistance system is another example. In other cases, exchange of something in the order of 4-6 pieces of information suffice to complete the task. The RailTel/ARISE train time-table information systems are cases in point. In these systems, the user has to provide the system with just enough information about departure station, arrival station, date, and possibly time of day, that the system can compute the missing information to be offered to the user. It is feasible today with such systems to let the users speak their minds, have the system realise that one or two data points may be missing from the user's input, ask for these, and then compute the answer to the user's query. The Danish Dialogue System, on the other hand, requires so much information from the user that letting the users speak their minds may be counterproductive. The likelihood that the system fails to recognise and understand what the users say becomes too great. The same is true of the dinner arrangement system mentioned above. In such cases of relatively large task complexity, it may be necessary to adopt strategies for controlling the user's *input* so that it may have a good chance of being recognised and understood by the system. Such strategies will be reviewed below (2.9).

(b) *Task structure:* what is the task structure, if any?

Some tasks are ill-structured or have no structure at all. Consider, for instance, an SLDS whose database contains all existing information on flight travel conditions and regulations for a certain airline company. This information tends to be quite comprehensive both because of the many practicalities involved in handling differently composed groups of travellers and their luggage, but also because of the legal ramifications of travelling which may surface if, e.g., the flight crashes. Some users may want many different individual pieces of information from the database whereas others may want only one piece. Which piece(s) of information a user wants is completely unpredictable. We call such user tasks *ill-structured tasks:* the user may want one, two or several pieces of information from the database, and the order in which the user may want the information is completely arbitrary as seen from the system's point of view. The system must be prepared for everything from the user all the time.

One way to try to reduce the complexity or large ill-structured tasks is to use, or invent, *domain structure.* That is, the domain may be decomposable into a number of sectors which themselves may be hierarchically decomposed, etc. So the systems asks, for instance: "Do you want to know about travelling with infants, travelling with pets, travelling for the elderly and the handicapped, hand luggage, or luggage for storage?" And if the user selects hand luggage,

the system asks: “Do you want to know about volume of permitted luggage, electronic equipment, fragile items, or prohibited luggage?” Etc. However, no user will accept to navigate through many hierarchical levels prompted by the system in order to find a single piece of information at the bottom of some deep domain hierarchy. Making the hierarchy more shallow will often make matters even worse. No user will accept to have to go through a shallow but broad hierarchy prompted by the system in order to find a single piece of information at the end of it. Just imagine a Danish time table inquiry system which asks the user: “Do you want to go from Aabenrå, Aalborg, Aarhus ...”, mentioning the country’s 650 or so train stations in alphabetical order and in chunks of, say, ten at a time.

Other tasks have some structure to them. Consider the flight ticket reservation task handled by the Danish Dialogue System. This task is at least a partially ordered one. It would normally make little sense, for instance, to ask for one of the morning departures until one has specified the date; and it normally makes little sense to expect the system to tell whether flights are fully booked or not on a certain date until one has indicated the itinerary. Moreover, ordinary users know that this is the case. Task structure is helpful if the task complexity makes it advisable to control the user’s input. It important to note, however, that partial-order-only is what one is likely to find in most cases. For instance, some users may want to know which departures are available at reduced fares before wanting to know about departure times, whereas others do not care about fare reductions at all. Moreover, even the partial order that there is, may be by default-only. If somebody simply wants to leave town as soon as possible, for instance, the itinerary matters less than the departure time of the first outbound flight which has a free seat.

(c) *Negotiation tasks:* does the task require substantial negotiation? Some task may involve a relatively low volume of information and may in addition have some structure to them. Still, they may be difficult to manage if they involve a considerable amount of negotiation. The Verbmobil meeting scheduling task is an example. To fix a meeting simply requires fixing a date, a time or a time interval, and possibly a place, so the volume of information is relatively low. Unless one knows the date, it can be difficult to tell if one is free at a certain time, so the task has some structure to it. And for busy people, the date-time pair may matter more than the exact venue. The problem inherent to the Verbmobil task is that fixing meetings may require protracted, and ill-structured, negotiations of each sub-task. In addition, Verbmobil does nothing to impose structure on the (human-human) dialogue for which it provides translation support, allowing the dialogue to run freely wherever the interlocutors want it to go. In such cases, it makes little sense to judge the task complexity in terms of the volume of information to be exchanged or in terms of the task structure that is present, because the real problem lies in negotiating and eventually agreeing to meet. Had Verbmobil been a *human-machine* dialogue system product, the machine representing the diary of someone not present to the conversation, state-of-the-art engineering practice would probably dictate much stronger system control of the dialogue (see 2.9). Note also that negotiation is a natural human propensity. It seems likely that most SLDSs can be seen to involve an element of negotiation. Verbmobil just involves a considerable amount of negotiation. In the case of the Danish Dialogue System, for instance, the system may propose a list of morning departures and ask if the user wants one of them. If not, the joint search for a suitable departure time continues. So the reason why negotiation is less obvious or prominent in, e.g., the dialogue conducted with the Danish Dialogue System when compared to the dialogue conducted with Verbmobil, is not that the task of the Danish system by itself excludes negotiation. Rather, the reason is that the system’s dialogue behaviour has been designed to constrain the way in which negotiations are done.

Summarising, when developing SLDSs we want to let the users speak their minds. However, if

the task is a complex one because it either requires transfer of large quantities of information from user to system, is large and ill-structured, or requires significant negotiation, the SLDS project either has to be given up as a commercial undertaking, turns into a large-scale research project, such as Verbmobil, or requires some or all of the following strategies: (a) input prediction, (b) input language processing control, (c) output control and (d) control of user input. To the extent that they are applicable, all of (a) through (d) are of course useful in any system but their importance grows with increasing task complexity. (d) subsumes (c) as shown in the next section. Obviously, (a) through (d) can also be used for very simple task, making dialogue engineering more easy to do for these tasks than if the users were permitted to speak their minds freely.

2.9 Controlling user input

The dialogue manager can do a lot to control the user's input in order to keep it within the system's technical capabilities of recognition and understanding. Among task-oriented SLDSs, the extreme in terms of lack of user input control is a system which merely tells the user about the task it can help solve and then advises the user to go ahead. For any but the most low-complexity tasks, clear signs of unconstrained user input are: very long sentences, topics raised, or sub-tasks addressed, in any order, and any number of topics being addressed per user utterance. Even for comparatively simple tasks, the handling of unconstrained user input is a difficult challenge.

Some of the user input control mechanisms available to the SLDS developer are:

- explicit instructions to users on what the system can and cannot do. For all but the simplest SLDSs, it is advisable that the system clearly and succinctly tells the user up-front things like what is its domain, what is its task, etc. This helps "tailoring" the user's expectations, and hence the user's input, to the knowledge the system actually has, thereby reducing the task of the dialogue manager. Instructions to users do not always have to be given through speech. Waxholm, for instance, provides these instructions on the screen as well as in synthetic speech. If the instructions are given through speech, it is important in most cases that they are being expressed briefly and clearly because users tend to loose attention very quickly in that situation;

- explicit instructions to users on how to address the system. For instance, the Danish Dialogue System tells it users that it will not be able to understand them unless they answer the system's questions briefly and one at a time;

- feedback on what the system has understood, so that, throughout the dialogue, the user is left in no doubt as to what the system has understood (see 2.19);

- processing feedback to the user. When the system processes the received information from the user and hence may not be speaking for a while, processing feedback keeps the user informed on what is going on and helps avoid unwanted user input (see 2.19);

- output control (cf. 2.6). The aim of output control is to "prime" the user through the vocabulary, grammar and style adopted by the system. Humans are extremely good at (unconsciously) adapting their vocabulary, grammar and style to those of their partners in dialogue or conversation. It is therefore useful to make the system provide output which only uses the vocabulary and grammar which the system itself can recognise, parse and understand, and to make the system use a style of dialogue which induces the user to provide input which is terse and to the point. The unconscious adaptation performed by the users ensures that they still feel that they can speak their minds without feeling hampered by the system's requirements for recognisable vocabulary, simple grammar, and terse style. A particular point to be aware of

in this connection is that if the system's output to the user includes, e.g., text graphics, then the textual output should be subjected to the same priming strategy as adopted for the system's spoken output. It is not helpful to carefully prime the user through the system's output speech and then to undercut the purpose of the priming through a flourishing style of expression in the textual output;

- focused output combined with system initiative (2.10). If the system determines the course of the dialogue by having the initiative all or most of the time, for instance through asking questions of the user or providing the user with instructions that the user has to carry out, a strong form of user input control becomes possible. The system can phrase its questions or instructions in a focused way so that, for each question, instruction, etc., the user only has to choose between a limited number of response options. If, for instance, the system asks a question which should be answered by a 'yes' or 'no', or by a name drawn from a limited set of names (of persons, airports, etc.), then it exerts a strong influence on the user's input. Dialogue managers using this approach may be able to handle even very large-complexity tasks in terms of information volume (2.8). Note that system initiative in itself is not sufficient for this kind of input control to take place. If, for instance, the system says "ADAP Travels, can I help you?", it does, in principle, take the initiative by asking a question. However, the question is not focused at all and therefore does not restrict the user's input. An open or unfocused system question may therefore be viewed as a way of handing over the dialogue initiative to the user. Note also that not every task can be handled through focused output combined with system initiative. Some tasks do not lend themselves to system initiative, and large unstructured tasks cannot be handled through focused output combined with system initiative in a way which is acceptable to the users.

Even if the above measures have been taken to the extent necessary for the application, the users may still speak their minds "out of order", such as when the system is speaking or processing received input. The likelihood of this happening varies from one application to another and this point should be considered by the dialogue manager developer. In some applications, it may even be desirable that the users can speak freely among themselves whilst the system is processing the spoken input. Still, barge-in technology is advisable for many SLDSs, so that the system is able to recognise and process user input even if it arrives when the system is busy doing something other than just waiting for it. In Waxholm, the system does not listen when it speaks. However, the user may barge in by pressing a button to interrupt the system's speech. This is useful when, for instance, the user already feels sufficiently informed to get on with the task. For instance, users who are experts in using the application can use the button-based barge-in to skip the system's introduction. The Danish Dialogue System does not allow barge-in when the system speaks. This turned out to cause several transaction failures, i.e. dialogues in which the user did not get the intended result. A typical case is the one in which the system provides feedback to the user (see 2.19) which shows that the user has been misunderstood. During the system's following output speech, the user says, e.g.: "No, Saturday". The lack of reaction from the system is interpreted by the user as if the system had received the error message, which of course it hasn't, and couldn't have. As a result, the user will later receive a flight ticket which is valid for the wrong day.

For specific purposes, such as when the users are professionals and will be using the SLDS extensively in their work, user input control can also be exerted through textual material which the user is expected to read in connection with using the system. This remedy will not work for walk-up-and-use systems unless the system includes a screen (cf. Waxholm) or other text-displaying peripherals.

2.10 Who should have the initiative?

Dialogue management design takes place between two extremes. From the point of view of technical simplicity, one might perhaps wish that all SLDSs could conduct their transactions with users as a series of questions to which the users would have to answer 'yes' or 'no' and nothing else. Simpler still, 'yes' or 'no' could be replaced by filled and unfilled pauses, respectively, between the system's questions. From the point of view of natural dialogue, on the other hand, users should say exactly what they want to say and when they want to say it, without any restrictions being imposed by the system. Both extremes are unrealistic, of course. If the task complexity is low in terms of information volume and the task does not involve negotiation, then the users may be allowed to say what they want (2.8). This does not alter the fact that, in most current SLDSs for low-complexity tasks, the users are *not* being allowed to freely express what they want but rather have to join into a system directed dialogue. When the task complexity becomes larger in terms of information volume, it begins to matter who has the initiative during the dialogue:

As long as the task is one in which the system requires a series of specific pieces of information from the user, the task may safely be designed as one in which the system preserves the initiative throughout by asking focused questions of the user. This *system-directed approach* would work even for very large-complexity tasks in terms of information volume and whether or not these tasks are well-structured. The difficult case is when the task is large in terms of information volume and both the user and the system need information from one another. This requires a *mixed-initiative approach*. Whilst asking its questions, the system must, always or sometimes, be prepared that the user may ask a question in return instead of answering the system's own question. For instance, the user may want to know if a certain flight departure allows discount before deciding whether that departure is of any interest. Finally, the *user-directed approach* is mainly for ill-structured tasks in which there is no way for the system to anticipate which parts of the task space the user wants to address on a particular occasion. The flight condition information task (2.8) is a case in point, as is the email operation task (2.4).

Initiative distribution among user and system is often a matter of degree. In some cases, it may even be difficult to classify an SLDS in terms of who has the initiative. If the system opens the dialogue by saying something like: "Welcome to service X, what do you want?" - one might say that the system has the initiative because the system is asking a question of the user. However, as the question the system asks is a completely open one, one might as well say that the initiative belongs to the user. In other words, only focused questions clearly determine initiative. The same holds true of other directive uses of language in which partner A tells partner B to do specific things, such as in instructional dialogues.

In practice, most SLDSs have to be mixed-initiative systems for the simple reason that user-initiated repair meta-communication is mandatory. The user must have the possibility of telling the system, at any point during dialogue, that the system has misunderstood the user or that the user needs the system to repeat what it just said (see 2.15). Only simple systems which lack meta-communication altogether can avoid that. Similarly, even if the user has the initiative throughout, the system must be able to take the initiative to do repair or clarification of what the user has said. When thinking about, or characterising, SLDSs, it may be useful, therefore, to distinguish between two issues: (a) who has the initiative in domain communication? and (b) who has the initiative in meta-communication? (More in 2.15).

From a dialogue engineering perspective, it may be tempting to claim that system-directed dialogue is generally simpler to design and control than either user-directed dialogue or mixed-initiative dialogue, and therefore should be preferred whenever possible. This claim merits several comments. Firstly, it is not strictly *known* if the claim is true. It is quite possible that

system-directed dialogue, for all but a small class of tasks, is *not* simpler to design and control than its alternatives because it needs ways of handling those users who do not let themselves be fully controlled but speak out of turn, initiate negotiation, ask unexpected questions, etc. Secondly, products are already on the market which allow mixed-initiative dialogue for relatively simple tasks, such as train time-table information. And it is quite likely that users generally tend to prefer such systems because they let the users speak their minds to some extent.

2.11 Input prediction/prior focus

In order to support the system's speech recognition, language processing and dialogue management tasks, the dialogue manager developer should investigate if selective prediction of the user's input is possible at any stage during the dialogue (2.6). This is clearly possible if, e.g., the system asks a series of questions each requesting specific pieces of information from the user. If the task has some structure to it, it may be possible to use the structure to predict when the user might ask questions of the system, thus facilitating the mixed-initiative approach (2.10) through selective prediction. I.a. the Daimler-Benz dialogue manager and the Danish Dialogue System use various forms of input prediction. Another way of describing input prediction is to say that the dialogue manager establishes a (selective) *focus of attention* prior to the next user utterance.

Input prediction can be achieved in many different ways. It may be useful to distinguish between two general approaches. In *knowledge-based input prediction,* the dialogue manager uses a priori knowledge of the context to predict one or more characteristics of the user's next utterance as illustrated in the preceding paragraph. In *statistical input prediction,* the dialogue manager uses corpus-based information on what to expect from the user. Given a corpus of user-system dialogues about the task(s) at hand, it may be possible to observe and use regularities in the corpus, such as that the presence of certain words in the user's input makes it likely that the user is addressing a specific subset of the topics handled by the system, or that the presence of dialogue acts DA5 and DA19 in the immediate dialogue history makes it likely that the user is expressing DA25. Waxholm uses the former approach, Verbmobil the latter.

Useful as it can be, input prediction may fail because the user does not behave as predicted. In that case, the dialogue manager must initiate appropriate meta-communication (see 2.15).

2.12 Sub-task identification

It is useful for the dialogue manager developer to consider the development task from the peculiar point of view of the dialogue manager. The dialogue manager is deeply embedded in the SLDS, is out of direct contact with the user and has to do its job based on what the speech and language layers deliver. This happens in the context of the task and whatever output and input control the dialogue manager may have imposed. Basically, what the speech and language layers can deliver to the dialogue manager is some form of meaning representation. Sometimes the dialogue manager does not receive any meaning representation from the speech and language layers even though one was expected. But even if a meaning representation arrives, there is no guarantee that this representation adequately represents what the user had in mind because the speech and language layers may have gotten the user's intended meaning wrong. Still, whatever happens, the dialogue manager must be able to produce appropriate output to the user.

Current SLDSs exhibit different approaches to the creation of a meaning representation in the speech and language layers as well as to the nature of the meaning representation itself. An

important point is the following: except for really low-complexity tasks, the fact that a meaning representation arrives with the dialogue manager is not sufficient for the dialogue manager to carry on with the task. *First, the dialogue manager must identify to which sub-task(s), or topics, if any, that meaning representation provides a contribution.* In other words, most task-oriented SLDSs require the dialogue manager to do sub-task identification or topic identification.

The task solved by most SLDSs can be viewed as including one or several sub-tasks or topics to be addressed by user and system. One or several of these sub-tasks have to be solved in order that the user and the system succeed in solving the task. Other sub-tasks may be optional, i.e. their solution is sometimes, but not always, needed. One example of optional sub-tasks is meta-communication sub-tasks (see 2.15): if the dialogue proceeds smoothly, no meta-communication sub-tasks have to be solved. Another example is optional sub-task loops, i.e. sub-tasks which some, but not all, users need solved, such as asking the system about a particular piece of information in order to be able to appropriately answer a system question.

For the dialogue manager, the fact that the user's input may be addressing several different sub-tasks means that it becomes paramount to identify the sub-task(s) which the user is actually addressing in a particular input utterance. Until the system knows, of believes it knows, that, it cannot do anything with what the user has said (see 2.13).

Basically, dialogue managers can be built so as to be in one of two different situations with respect to sub-task identification. In the first case, the dialogue manager has good reason to assume that the user is addressing a particular domain sub-task; in the second case, the dialogue manager does not know which domain sub-task the user is addressing.

- the dialogue manager may have good reason to assume that the user's utterance addresses a specific sub-task, such as providing the name of an employee in the organisation hosting the system. Depending on the task and the dialogue structure design, there can be many different reasons why the dialogue manager knows which sub-task the user is currently addressing: there may be only one sub-task, as in an extremely simple system; the task may be well-structured; the system just asked the user to provide input on that sub-task, etc. Generally speaking, this is a good situation for the dialogue manager to be in, as in the Danish Dialogue System. It knows (or has good reason to believe) that the user is either addressing a specific domain sub-task or that the user has initiated meta-communication (2.15). Since it knows which sub-task the user is addressing, the task of the dialogue manager reduces to that of finding out exactly what is the user's contribution to that sub-task (or one of those sub-tasks, if we count in the possibility that the user may have initiated meta-communication). In such cases, the system has a *local focus;*

- the dialogue manager does not know which of several possible sub-tasks the user is addressing. The main reason why the dialogue manager does not know which sub-task the user is currently addressing, is that the dialogue manager has given the user the initiative, for instance by asking an open question in response to which the user may have addressed any number of possible sub-tasks. Alternatively, the user unexpectedly took the initiative. In such situations, the dialogue manager has to do sub-task identification, or topic identification, before it can start processing the user's specific contribution to the sub-task. Sub-task identification is crucial in systems such as RailTel/ARISE, Waxholm and Verbmobil. Waxholm uses probabilistic rules linking semantic input features with topics. For sub-task identification, Verbmobil uses weighted default rules to map from input syntactic information, keywords, and contextual information about which dialogue acts are likely to occur, into one or several dialogue acts belonging to an elaborate taxonomy of 54 speech (or dialogue) acts. The mapping can be done in a 'shallow processing' mode as well as in a 'deep processing' mode.

An important additional help in sub-task identification is support from a *global focus,* for instance when the dialogue manager knows that the task history (see 2.21) contains a set of not-yet-solved sub-tasks. These tasks are more likely to come into local focus than those which have been solved already. Another form of global focus can be derived from observation of dialogue phases. Most task-oriented dialogues unfold through three main phases, the introduction phase, the main task-solving phase, and the closing phase. Sometimes it may be possible to break down the main task-solving phase into several phases as well. If the system knows which phase the dialogue is in at the moment, this knowledge can be used for sub-task identification support. Knowing that can be a hard problem, however, and this is a topic for ongoing research. For instance, the absence of certain discourse particles called topic-shift markers and the presence of certain dialogue acts may suggest that the user has not changed dialogue phase.

In both cases, the dialogue manager must start from the semantic representations that arrive from the speech and language layers, look at the semantically meaningful units, and seek to figure out which sub-task the user is addressing.

Generally speaking, the more possible sub-tasks the user might be addressing in a certain input utterance, the harder the sub-task identification problem becomes for the dialogue manager. When doing sub-task identification, the dialogue manager may follow one of two strategies. The simpler strategy is to try to identify one sub-task only in the user's input even if the user may have been addressing several sub-tasks, and continue the dialogue from there as in Waxholm. The more difficult strategy is to try to identify each single sub-task addressed by the user as in RailTel/ARISE. The latter strategy is more likely to work when the task complexity is low in terms of volume of information.

Depending on what arrives from the speech and language layers, and provided that the dialogue manager has solved its sub-task identification task, the dialogue manager must now determine the users' specific contribution(s) to the sub-task(s) they are addressing (see 2.13). Following that, the dialogue manager must do one of five things as far as communication with the user is concerned: (a) advance the domain communication (see 2.14), (b) initiate meta-communication with the user (see 2.15 and 2.18), (c) initiate other forms of communication (see 2.16), (d) switch to a fall-back human operator, or (e) end the dialogue (see 2.20). Advancing the domain communication means getting on with the task. Initiating meta-communication means starting a sub-dialogue (or, in this case, a meta-dialogue) with the user in order to get the user's meaning right before advancing the domain communication any further. No SLDS probably can do without capabilities for (a) and (b). If (b) fails repeatedly, some systems have the possibility of referring the user to a human operator. Otherwise, calling off the dialogue is the only possibility left.

In parallel with taking action vis-à-vis the user, the dialogue manager may at this stage take a smaller or larger series of internal processing steps which can be summarised as: (f) updating the context representation (see 2.21) and (g) providing support for the speech and language layers to assist their interpretation of the next user utterance (2.6).

2.13 Advanced linguistic processing

The determination of the user's contribution to a sub-task requires more than, to give just an example, the processing of semantic feature structures. Processing feature structures often implies value assignment to slots in a semantic frame even though these values cannot straightforwardly be derived from the system input in all cases. If the system has to decide on *every* contribution of a user to a sub-task, something which few systems do at present,

advanced linguistic processing is needed. It may involve, among other things, cross-sentence co-reference resolution, ellipsis processing, the processing of discontinuous user input involving large gaps in the expected sequence of dialogue acts, and the processing of indirect dialogue acts. In nearly all systems, the processing of most of these phenomena is controlled and carried out by one of the natural language components - by the parser in the Daimler-Benz dialogue manager, by the semantic evaluation component in the Verbmobil translation system - but never without support from the dialogue manager.

In case of cross-sentence co-reference and ellipsis processing, the natural language system component in charge is supported by the dialogue manager which provides a representation of contextual information for the purpose of constraining the relevant search space. The contextual information is part of the dialogue history (see 2.21). Dialogue history information consists in one or several representation structures that are being built up incrementally to represent one or more aspects of the preceding dialogue. In principle, the more aspects of the preceding dialogue are being represented in the dialogue history, the more contextual information is available for supporting the processing done in the language layer, and the better performance can be expected from that layer.

Discontinuous user input ('input gaps'), where the user unexpectedly "jumps" ahead or backwards in the dialogue model, forms a problem for most systems depending on gap size and, more basically, on whether the user's input is controlled in such a way that the occurrence of the phenomenon is avoided as much as possible. In case that the occurrence of discontinuous input is not controlled by the system, this may give rise to problems in case of relatively large input gaps. In Verbmobil, for instance, dialogue act prediction by the keyword spotting component fills in missing dialogue acts in case of small input gaps. However, the technique used by this component is insufficient for gaps that are larger than two dialogue acts.

Finally, advanced linguistic processing also includes the processing of indirect dialogue acts. The central problem for the system is to identify the "real" dialogue act performed by the user and disguised as a dialogue act of a different type. In contrast to direct dialogue acts, indirect dialogue acts cannot be determined on the basis of their surface form which makes the frequently used keyword spotting techniques used for the identification of direct dialogue acts almost useless in these cases. Clearly, the processing of indirect dialogue acts calls for less surface oriented processing methods involving semantic and pragmatic information associated with input sequences. This is a hard problem.

Communication

2.14 Domain communication

The primary task of the dialogue manager is to advance the domain communication based on a representation of the meaning-in-task-context of the user's input (cf. 2.12 and 2.13). Obviously, what to do in a particular case depends on the task and the sub-task context. The limiting case is that the dialogue manager simply decides that it has understood what the user said and takes action accordingly, such as connecting the caller to a user who has been understood to want to accept a collect call, replaying an email message, or displaying a map on the screen. Some other cases are that the dialogue manager:

(a) inserts the user's input meaning into a slot in the task model, discovers that more information is needed from the user, and proceeds by eliciting that information;

(b) looks up the answer to the user's question in the database containing the system's domain knowledge and sends the answer to the language and speech generation components (or to the

screen, etc.). All the systems described in the Appendix include a database containing domain facts and rules;

(c) inserts the user's input meaning into a slot in the task model, verifies that it has all the information needed to answer the user's query, and sends the answer to the language and speech generation components (or to the screen, etc.);

(d) makes an inference based on the user's input meaning, inserts the result into a slot in a database and proceeds with the next question. For instance, as in Waxholm, the system completes the user's 'on Thursday' by inferring the appropriate date, and replaces qualitative time expressions, such as 'this morning', by well-defined time windows, such as '6 AM - 12 AM'. Verbmobil does inferencing over short sequences of user input, such as that a counterproposal (for a date) implies the rejection of a previous proposal; a new proposal (for a date) implies the acceptance of a (not incompatible, less specific) previous proposal; and a change of dialogue phase implies the acceptance of a previous proposal (for a date).

It may be noted that such domain-based inferences abound in human-human conversation. The dialogue manager has no way of replicating the sophistication of human inferencing during conversation and dialogue. Most current systems are able to process only relatively simple inferences. The dialogue manager developer should, therefore, focus on enabling all and only the inferences that are strictly necessary for the application to work. Even that can be a hard task and it may often be uncertain which inferences really are necessary. For instance, should the system be able to perform addition of small numbers or not? Simple as this may appear, it adds a whole new chapter to the vocabulary, grammar and rules of inference that the system should master. For instance, in travel booking applications, some users would naturally say things like "two adults and two children"; or, in travel information applications, some users may want to know about the 'previous' or the 'following' departure given what the system has already told them. The developer has to decide how important the system's capability of understanding such phrases is to the successful working of the application in real life. In most cases, this requires empirical investigation of actual user behaviour. In addition, the developer must try to prevent the user from requiring the system to do inferences that are not strictly needed for the application;

(e) discovers that the user's input meaning is likely to make the system produce too much output information and produces a request to the user to provide further constraints on the desired output;

(f) discovers that the user's input meaning is inconsistent with the database, infeasible given the database, inconsistent with the task history, etc. The system may reply, e.g., "There is no train from Munich to Frankfurt at 3.10 PM ....", or "The 9 o-clock flight is fully booked already ....";

(g) translates the user's input meaning into another language.

Among the examples above, (a), (b), (c), (d) and (g) illustrate straightforward progression with the task. (e) and (f) illustrate domain sub-dialogues. Meta-communication sub-dialogues are described in Section 2.15.

The system may, in fact, do something more than just advancing the domain communication as exemplified above. As part of advancing the domain communication, the system may provide feedback to the user to make sure that what the user just said has been understood correctly (see 2.19).

2.15 Meta-communication

Meta-communication, although secondary to domain communication, is crucial to proper dialogue management. In addition, meta-communication is complex and potentially difficult to design. In meta-communication design, it is useful to think in terms of distinctions between (a) system-initiated and user-initiated meta-communication and (b) repair and clarification meta-communication. One of the partners in the dialogue initiates meta-communication because that partner has the impression that something went wrong and has to be corrected.

System-initiated repair meta-communication is needed whenever the system has reason to believe that it did not understand the user's meaning. Such cases include, i.a.:

- nothing arrived for the dialogue manager to process although a user's meaning input was expected. To provide appropriate output in such cases, the dialogue manager must get the user to input the meaning once more. It is worth noting that this can be done in many different ways, from simply saying, e.g., "Sorry, I did not understand" or "Please repeat" to asking the user to speak louder or more distinctly. The more the system knows about the probable cause of its failing to understand the user, the more precise its repair meta-communication can be. Any such gain in precision increases the likelihood that the system will understand the user the next time around (see also 2.18);

- something arrived for the dialogue manager to process but what arrived was meaningless in the task context. For instance, the user may be perceived as responding 'London' to a question about departure date. To provide appropriate output in such cases, the dialogue manager may have to ask the user to input the meaning once more. However, as the system actually did receive some meaning representation, it might tell the user what it did receive and that this was not appropriate in the task context. This is done by, e.g., Waxholm and the Danish Dialogue System. For instance, if the Danish Dialogue System has understood the user to want to fly from Aalborg to Karup, it will tell the user that there are no flight connections between these two airports. Note, however, that, e.g., RailTel/ARISE would take the user's 'London' to indicate a change to the point of departure or arrival (see below, this section). Verbmobil uses a statistical technique to perform a kind of constraint relaxation in case of a contextually inconsistent user input dialogue act (cf. 2.13).

A core problem in repair meta-communication design is that the user input that elicits system-initiated repair may have many different causes. The closer the dialogue manager can get to correctly inferring the cause, the more informative repair meta-communication it can produce, and the more likely it becomes that the user will provide comprehensible and relevant input in the next turn.

System-initiated clarification meta-communication is needed whenever the system has reason to believe that it did understand the user's meaning which, however, left some kind of uncertainty as to what the system should produce in response. Such cases include, i.a.:

- a representation of the user's meaning arrived with a note from the speech and/or language processing layers that they did not have any strong confidence in the correctness of what arrived. The best approach for the dialogue manager to take in such cases probably is to get the user to input the meaning once more rather than carry on the basis of dubious information, which may easily lead to a need for more substantial meta-communication later on. However, as the system actually did receive some meaning representation, it might instead tell the user what it received and ask for the user's confirmation;

- a representation of the user's meaning arrived which was either inherently inconsistent or inconsistent with previous user input. In cases of inherent inconsistency which the system judges on the basis of its own domain representation, the system could make the possibilities

clear to the user and ask which possibility the user prefers, for instance by pointing out that 'Thursday 9th' may be either 'Thursday 8th' or 'Friday 9th'. Cases of inconsistency with previous user input are much more diverse and different response strategies may have to be used depending on the circumstances;

- a representation of the user's meaning arrived which was (semantically) ambiguous or underspecified. For instance, the user asks to be connected to Mr. Jack Jones and two gentlemen with that name happen to work in the organisation; or the user wants to depart at "ten o-clock", which could be either AM or PM. In such cases, the system must ask the user for information that can resolve the ambiguity. The more precisely this can be done, the better. For instance, if the system believes that the user said either 'Hamburg' or 'Hanover', it should tell the user just that instead of broadly asking the user to repeat. Experience indicates that it is dangerous for the system to try to resolve ambiguities on its own by selecting what the system (designer) feels is the most likely interpretation. The designer may think, for instance, that people are more likely to take an airplane at ten AM than at ten PM and may therefore assign the default interpretation "ten AM" to users' "ten o-clock". If this approach is followed, it is advisable to explicitly ask the user for verification through a "yes/no" feedback question (see 2.19).

Whilst user meaning inconsistency and (semantic) ambiguity are probably the most common and currently relevant illustrations of the need for system clarification meta-communication, others are possible, such as when the user provides the system with an irresolvable anaphor. In this case, the system should make the possible referents clear to the user and ask which of them the user has in mind. As the above examples illustrate, system-initiated clarification meta-communication is often a 'must' in dialogue manager design.

In general, the design of system clarification meta-communication tends to be difficult and the developer should be prepared to spend considerable effort on reducing the amount of system clarification meta-communication needed in the application. However, as so often is the case in systems design, this piece of advice should be counterbalanced by another. Speaking generally, users tend to loose attention very quickly when the system speaks. It is therefore no solution to let the system instruct the user at length on what the system really means or wants on every occasion where there is reason to believe that the user may go on to say something which is ambiguous or (contextually) inconsistent. In other words, careful prevention of user behaviour which requires system-initiated clarification meta-communication should be complemented by careful system clarification meta-communication design. One point worth noting is that, for a large class of system-initiated clarifications, yes/no questions can be used.

User-initiated repair meta-communication is needed whenever the system has demonstrated to the user that is has misunderstood the user's intended meaning. It also sometimes happens that users change their minds during dialogue, whereupon they have to go through the same procedures as when they have been misunderstood by the system. In such cases, the user must make clear to the system what the right input is. Finally, users sometimes fail to get what the system just said. In this case, they have to ask the system to repeat just as when the system fails to get what the user just said. These three (or two) kinds of user repair meta-communication are mandatory in many systems. Examples are Waxholm, the Daimler-Benz dialogue manager and the Danish Dialogue System.

User-initiated repair meta-communication is difficult to design. Ideally, we would like the user to just speak their minds whenever they have been misunderstood by the system, changed their minds with respect to what to tell the system, or failed to get what the system just said. Some systems do that, such as Waxholm, but with varying success, the problem being that users may initiate repair in very many different ways, from "No, Sandhamn!" to "Wait a minute. I didn't

say that. I said Sandhamn!" Other systems require the user to use specifically designed keywords for this purpose, again with varying success. Thus, in the Danish Dialogue System, users are asked to use the keyword 'change' whenever they have been misunderstood by the system or changed their minds, and to use the keyword 'repeat' whenever they failed to get what the system just said. Keywords are simpler for the system to handle than unrestricted user speech but users sometimes fail to remember the keywords they are supposed to use. A third approach is being used in RailTel/ARISE. This approach is similar to using an eraser: one erases what was there and writes something new in its place. For instance, if the system gets 'Frankfurt to Hanover' instead of 'Frankfurt to Hamburg', the user simply has to repeat 'Frankfurt to Hamburg' until the system has received the message. No specific repair meta-communication keywords or meta-dialogues are needed. The system is continuously prepared to revise its representation of the user's input based on the user's latest utterance. Whilst this solution may work well for low-complexity tasks, it will not work for tasks involving selective input prediction (2.11) and may be difficult to keep track of in high-complexity tasks.

User -initiated clarification meta-communication is probably the most difficult challenge for the meta-communication designer. Just like the user, the system may output, or appear to the user to output, inconsistent or ambiguous utterances, or use terms which the user is not familiar with. In human-human conversation, these problems are easily addressed by asking questions such as: "What do you mean by green departure?" or "Do you mean scheduled arrival time or expected arrival time?" Unfortunately, most current SLDSs are not being designed to handle such questions at all. The reasons are that (a) this is difficult to do and, often more importantly, (b) that the system developers have not discovered such potential problems in the first place. If they had, they might have tried to avoid them in their design of the system's dialogue behaviour. Thus, they would have made the system explain the green departure before the user was likely to ask what it is, and they would have made the system explicitly announce when it was speaking about scheduled arrivals and when is was speaking about expected arrivals. In general, this is one possible strategy to follow by the dialogue manager developer: to remove in advance all possible ambiguities, inconsistencies and terms unknown to users, rather than to try to make the system handle questions from users about these things. A tool is being developed in DISC to support the design of co-operative system dialogue (Bernsen and Dybkjær, to appear). Part of the purpose of this tool is to avoid situations in which users feel compelled to initiate clarification meta-communication.

There is an obvious alternative to the strategy recommended above, i.e. to generally try to prevent the occurrence of user-initiated clarification meta-communication. The alternative is to strengthen the system's ability to handle user-initiated clarification meta-communication. It seems likely that the nature of the task is an important factor in determining which strategy to emphasise. Consider, for instance, users inquiring about some sort of yellow pages commodity, such as electric guitars or used cars. Each of these is a relatively complicated subjects. In addition, the inquiring users are likely to differ widely in their knowledge about electric guitars and cars. A flight ticket reservation system may be able to address its domain *almost* without using terms which are unknown to its users, whoever these may be. Not so with a used cars information system. As soon as the system mentions ABS brakes, racing tyres or split back seats, some users will be wondering what the system is talking about. In other words, there seems to be a large class of potential SLDSs which can hardly start talking without speaking gibberish to some of their intended users. In such cases, the dialogue manager developers have better prepare for significant user-initiated clarification meta-communication. It is no practical option for the system to explain all the domain terms it is using as it goes along.

To summarise, the dialogue manager is several steps removed from direct contact with the

user. As a result, the dialogue manager may fail to get the user's meaning or it may get it wrong. Therefore, both the system and the user need to be able to initiate repair meta-communication. Even at low levels of task complexity, users are able to express themselves in ways that are inconsistent or ambiguous. The system needs clarification meta-communication to handle those user utterances. Much user clarification meta-communication should be prevented rather than allowed but for in tasks, user clarification meta-communication plays a large role in the communication between user and system.

2.16 Other forms of communication

Domain communication and meta-communication are not the only forms of communication that may take place between an SLDS and its users. Thus, the domain-independent opening of the dialogue by some form of greeting is neither domain communication nor meta-communication. The same applies to the closing of the dialogue (see 2.20). These formalities may also be used in the opening and closing of sub-dialogues. Another example is system time-out questions, such as "Are you still there?", which may be used when the user has not provided input within a certain time limit.

If the SLDSs task-delimitation is not natural to users, they are likely to sometimes step outside the system's limited conception of the task. By the system's definition, their communication then ceases to be domain communication. For some tasks, out-of-domain communication may happen too often for comfort for the dialogue manager developer who may therefore want to do something about it. Thus, Waxholm is sometimes able to discover that the user's input meaning is outside the domain handled by the system. This is a relatively sophisticated approach because the system must be able to understand out-of-domain terms. The approach may be worth considering in cases where users may have reason to expect that the system is able to handle certain sub-tasks which the system is actually unable to handle. When the users address those sub-tasks, the system will tell them that, unfortunately, it cannot help them. In the Verbmobil meeting scheduling task, users are prone to produce reasons for their unavailability on certain dates or times. Verbmobil, however, whilst being unable to understand such reasons, classifies them nevertheless and represents them in the topic history (see 2.21).

2.17 Expression of meaning

Once the system knows what to say to the user, this meaning representation must be turned into an appropriately expressed output utterance. In many cases, this is being done directly by the dialogue manager which, having done its internal processing jobs, for instance:

(a) selects a stored audio utterance and causes it to be played to the user by sending a message to the player;

(b) concatenates the utterance from stored audio expressions or phrases and causes it to be played to the user by sending a message to the player;

(c) selects or fills an output sentence template and causes it to be synthesised it to the user.

Strategies (a) and (b) are closely related and are both used in, e.g., the Danish Dialogue System. Waxholm and the Daimler-Benz dialogue manager use (c). (c) is compatible with relatively advanced use of control layer information for, e.g., determining the prosody of the spoken output. This can also be done in the (a) approach but is difficult to do in the (b) approach because of the difficulty of controlling intonation in concatenated pre-recorded speech.

(d) A more sophisticated approach is to have the dialogue manager produce the *what,* or the

meaning, of the intended output and then have the output language layer determine the *how,* or the form of words to use, in the output. In this approach, the how is often co-determined by accompanying constraints from the dialogue manager's control and context layers, such as that the output should be a question marked by rising pitch at the end of the spoken utterance.

2.18 Error loops and graceful degradation

An additional issue for consideration by the dialogue management developer is the possibility that the user simply repeats the utterance which caused the system to initiate repair meta-communication. The system may have already asked the user to speak louder or to speak more distinctly but, in many such cases, the system will be in exactly the same uncomprehending situation as before. The system may try once more to get out of this potentially infinite loop but, evidently, this cannot go on forever. In such cases, the system might either choose to fall back on a human operator or close the dialogue. To avoid that, a better strategy in many cases is that the system attempts to carry on by changing the level of interaction into a simpler one, thereby creating a 'graceful degradation' of the (meta-) communication with the user. Depending on the problem at hand and the sophistication of the dialogue manager, this can be done in many different ways:

- the user may be asked focused questions one at a time instead of being allowed to continue providing one-shot input which may be too lengthy or otherwise too complex for the system to understand. This is a common approach used in, i.a., RailTel/ARISE, Waxholm and the Daimler-Benz dialogue manager;

- the user may be asked to re-phrase the input or to express it more briefly, for instance when the user's answer to a focused question is still not being understood;

- the user may be asked to produce a complete sentence rather than grammatically incomplete input, as in Waxholm;

- the user may be asked to answer a crucial question by 'yes' or 'no';

- the user may be asked to spell a crucial word, such as a person name or a destination.

It is important to note that the levels of interaction/graceful degradation approach can be used not only in the attempt to get out of error loops but also in combination with system-initiated clarification meta-communication. So, generalising, whenever the system is uncertain about the user's meaning, graceful degradation may be considered. The Daimler-Benz dialogue manager standardly accepts three repetitions of a failed user turn before applying the graceful degradation approach.

2.19 Feedback

System feedback to users is essential to successful dialogue management. The user must have the opportunity to verify that the system has understood the user's input correctly. In general, the user should receive feedback on each piece of information which has been input to the system. The feedback needs not be communicated through speech. Waxholm, for instance, provides a textual representation on the screen of what the system has recognised as well as of the system's output response. The important thing is that the user can perceive the feedback and verify if what the system did was what the user intended the system to do by providing a certain input. So the system's feedback might consist in presenting a particular map on the screen, or a table packed with information of some kind or other, or in playing a certain voice mail which it believes that the user has asked for. In many cases, however, the feedback will be speech produced by the system. Imagine the following dialogue:

Dialogue 1

S1: “ADAP Travels, can I help you?”

U1: “When is the first morning train from Frankfurt to Hamburg tomorrow morning?”

S2: “5.35 AM”.

U2: “Many thanks. Goodbye.”

S3: “Goodbye”.

The conversation looks OK, the only problem being that the system misrecognised ‘Hamburg’ as ‘Hanover’! Dialogue 1, therefore, is a transaction failure (see Chapter 3). A better solution would be:

Dialogue 2

S1: “ADAP Travels, can I help you?”

U1: “When is the first morning train from Frankfurt to Hamburg tomorrow morning?”

S2: “The first train from Frankfurt to Hanover on 3rd May 1998 leaves at 5.35 AM”.

U2: [Initiates repair meta-communication.]

S3: ....

The system feedback in the example above is relatively straightforward to design. One thing to consider is whether the system should repeat *twice* some or all of its core information to the user, such as in saying “... 5.35 AM, 5.35 AM”. However, the feedback strategy in Dialogue 2 cannot always be trusted. What may happen is the following transaction failure:

Dialogue 3

S1: “ADAP Travels, can I help you?”

U1: “When is the first morning train from Frankfurt to Hamburg tomorrow morning?”

S2: “The first train from Frankfurt to Hanover on 3rd May 1998 leaves at 5.35 AM”.

U2: “Thank you.”

S3: “Do you need any other information?”

U3: ....

In Dialogue 3, the user simply ignores the system’s feedback and walks away with what is likely to be false information, just as in Dialogue 1. If the users turn out to be too prone to make the error illustrated in Dialogue 3, the strategy shown in Dialogue 4 might be considered:

Dialogue 4

S1: “ADAP Travels, can I help you?”

U1: “When is the first morning train from Frankfurt to Hamburg tomorrow morning?”

S2: “You want to go from Frankfurt to Hanover?”

U2: [Initiates repair meta-communication.]

S3: ....

Experience has shown that the strategy illustrated in Dialogue 4 is more robust than the strategy illustrated in Dialogue 3. The price to be paid for adopting the Dialogue 4 strategy is that the user and the system have to spend more dialogue turns to solve the task. The Dialogue 4 strategy can be considered an early step in graceful degradation (2.18).

However, as often happens when task complexity increases, difficulties emerge. If the system, e.g., needs a larger volume of information than can be conveyed by the user in one utterance, it is advisable to give the users feedback before they are to input the next piece(s) of information. Otherwise, the user may have to correct something that happened several dialogue turns earlier, which can be difficult. The safest but also most intrusive way for the system to provide per-turn feedback is to say, e.g., "You want to go from Frankfurt to Hanover?" as in Dialogue 4 above. The user must answer with a 'yes' or 'no' before getting on with the task. A less intrusive feedback strategy is for the system to say, e.g.: "From Frankfurt to Hanover, when do you want to leave?" Unfortunately, this strategy can be as unsafe as the strategy illustrated in Dialogue 2 above. Sometimes users will ignore the feedback from the system and focus on answering its next question instead. The consequences will be as bad as if the system had not provided any feedback at all.

The amount and nature of the feedback the system should give to the user also depends on factors such as the cost and risk involved in the user-system transaction. Feedback on important bank transfers or costly travels are obviously more critical than feedback on which email the system should now be reading to the user. Current opinion probably is that the dialogue manager developer should prefer the safer among the two most relevant feedback options. Even travel information, if the user gets it wrong, can have serious consequences for that user.

In addition to the feedback discussed above, which might be called *information feedback,* SLDS dialogue manager developers may also consider to provide process feedback.

Process feedback is meant to keep the user informed that the system is "still in the loop", that it has not gone down but is busy processing information. Otherwise, the user may, e.g., believe that the system has gone down and hang up, wonder what is going on and start asking questions, or believe that the system is waiting to receive information and start inputting information which the system does not want to have. Process feedback is still at an early stage for SLDSs. It is quite possible for today's dialogue manager designers to come up with new, ingenious ways of providing the needed feedback on what the system is up to when it does not speak to the user and does not wait for the user to speak. The best feedback need not be spoken words or phrases but might be grunts or ehm's, tones, melodies, or appropriate earcons. Waxholm tells the user, for instance, "I am looking for boats to Sandhamn." In addition, Waxholm uses its screen to tell the user to wait while the system is working.

2.20 Closing the dialogue

Depending on the task, the system's closing of the dialogue may be either a trivial matter, an unpleasant necessity or a stage to gain increased efficiency of user-system interaction.

Closing the dialogue by saying, e.g., "Thank you. Good bye.", is a trivial matter when the task has been solved and the user does not need to continue the interaction with the system. Users often hang up without waiting for the system's farewell.

Closing the dialogue is a dire necessity when the system has spent its bag of tricks to overcome repeated error loops (2.18) and failed, or when the system hands over the dialogue to a human operator. In the former case, the system might ask the user to try again.

In some cases, however, when the user has solved a task, the dialogue manager should be

prepared for the possibility that the user may want to solve another task without interrupting the dialogue. This may warrant asking the user if the user wants to solve another task. Only if the user answers in the negative should the system close the dialogue.

The possibility that the user may want to close the dialogue at any point in time is often neglected in dialogue manager design, probably because the user can always simply hang up or walk away.

History, Users, Implementation

2.21 Histories

As soon as the task complexity in terms of information volume exceeds one piece of information, the dialogue manager may have to keep track of the history of the interaction. 'Dialogue history' is a term that covers a number of different types of dialogue records which share the function of incrementally building a dialogue context for the dialogue manager to use or put at the disposal of the language and speech layers. Note that a 'dialogue history' is *not* a log file of the interaction but a dedicated representation serving some dialogue management purpose. Note also that a 'dialogue history' may be a record of some aspect of the entire (past) dialogue or it may be a record of only a part of the dialogue, such as a record which only preserves the two most recent dialogue turns. In principle, a dialogue history may even be a record of several dialogues. This may be useful for building performance histories (see below) and might be useful for other purposes as well.

Most applications need a *task history,* i.e. a record of which parts of the task have been completed already. The task history enables the system to focus its output to the user on the sub-tasks which remain to be completed; to avoid redundant interaction; to have a global focus (2.12); and to enable the user to selectively correct what the system has misunderstood without having to start all over with the task. The task history does not have to preserve any other information about the preceding dialogue, such as how the user expressed certain things, or in which order the sub-tasks were resolved. If an output screen is available, the task history may be displayed to the user, as in Waxholm. If the user modifies some task parameter, such as altering the departure time, it becomes necessary to remove all dependent constraints from the task history.

A *topic history* is more complex. It is a record of the topics which have come up so far during the dialogue and possibly in which order they have come up. Even low-complexity systems can benefit from partial or full topic histories, for instance for detecting when a miscommunication loop has occurred (2.18) which requires the system to change its dialogue strategy, or for allowing users to do input repair arbitrarily far back into the preceding dialogue. Another thing which a topic history does is to build a context representation during dialogue, which can be much more detailed than the context built through the task history. In spoken translation systems, such as Verbmobil, the context representation provided by the topic history is necessary to constrain the system's translation task.

A *linguistic history* builds yet another kind of context for what is currently happening in the dialogue. The linguistic history preserves the actual linguistic utterances themselves (the surface language) and their order, and is used for advanced linguistic processing purposes (2.13). Preserving the linguistic history helps the system interpret certain expressions in the user's current input, such as co-references. For instance, if the user says: "I cannot come for a meeting on the Monday", then the system may have to go back through one or more of the user's previous utterances to find out which date 'the Monday' is. The Daimler-Benz dialogue

manager and Verbmobil, for instance, use linguistic history for co-reference resolution. Compared to the task history and the topic history, a linguistic history is a relatively sophisticated thing to include into one's dialogue manager at present.

A *performance history* is rather different from any of the above histories. The system would build a performance history in order to keep track of, or spot relevant phenomena in, the users' behaviour during dialogue. So a performance history is not about the task itself but about how the user handles the task in dialogue with the system. For instance, if the system has already had to resolve several miscommunication loops during dialogue with a particular user, it might be advisable to connect that user with a human operator rather than continue the agony. One way or another, performance histories contribute to building models of users, whether during a single dialogue or during a series of dialogues with a particular user.

Future systems solving high-complexity tasks are likely to both a task history, a topic history and a linguistic history, as is the case in Verbmobil. For increased usability, they may need a performance history as well.

2.22 Novice and expert users, user groups

The discussion above has focused on the central importance of the task to the dialogue manager developer. However, developers also have to take a close look at the intended users of the application as part of designing the dialogue manager.

An important issue common to many different dialogue management tasks is the difference between novice and expert users. This is a difference continuum, of course, and it may sometimes be important to the dialogue manager developer that there are, in fact, two different distinctions between novice and expert users. In particular, someone may be an expert in the domain of the application but a novice in using the system itself. Depending on for which of four user groups (system expert/domain expert, system expert/domain novice, system novice/domain expert, system novice/domain novice) the system is to be developed, the dialogue manager may have to be designed in different ways. For instance, if the target group is domain and system experts only, the developer may be able to impose strict task performance order, mandatory command keywords, etc., and support use of the system through written instructions, all of which makes the dialogue manager design much easier to do. If the target group is walk-up-and-use users who can be expected to be novices in using the system, a much more user-tailored design may be required. This is the case for all the systems described in the Appendix (except for the fact that Verbmobil is speaker-adaptive and hence only meant for novice users but not for walk-up-and-use users). The need for elaborate design increases even further if the system novices are also domain novices, so that any technicality either has to be removed or explained at an appropriate point during dialogue. For instance, even though virtually every user comes close to being a domain expert in time-table travel information, many users do not know what a 'green departure' is and therefore have to be told.

The 'downside' of doing elaborate dialogue design for walk-up-and-use users can be that (system) expert users rightly experience that their interaction with the system becomes less efficient than it might have been had the system included special shortcuts for expert interaction. Given the simplicity of current SLDSs, users may quickly become (system) experts, which means that the short-cut issue is a very real one for the dialogue manager developer to consider. The Danish Dialogue System, for instance, allows (system) expert users to by-pass the system's introduction and avoid getting definitions of green departures and the like. Waxholm allows its users to achieve the same thing through its barge-in button. The

acceptance of unrestricted user input in RailTel/ARISE means that experienced users are able to succinctly provide all the necessary information in one utterance. Novice users who may be less familiar with the system's exact information requirements, may provide some of the information needed and be guided by the system in order to provide the remaining information.

The dialogue manager developer may have to consider user groups other than novices and experts, such as the visually impaired or users whose dialects or accents create particular problems of recognition and understanding. In the case of dialects or accents, performance history information might suggest that the dialogue manager make use of the graceful degradation approach (2.18).

2.23 Other relevant user properties

If the task to be solved by the dialogue manager is above a certain (low) level of complexity, the dialogue manager designer is likely to need real data from user interactions with a real or simulated system in order to get the design right at design-time. Important phenomena to look for in this data include:

- what are the users' *standard goal(s)* in the task domain? If the users tend to have specific standard goals they want to achieve in dialogue with the system, there is a problem if the system is only being designed to help achieve some, but not all, of these goals. Strict user input control (2.9) may be a solution, but do not count on it to work in all possible conditions. Deep-seated user goals can be difficult or impossible to control. Of course, another solution is to increase the task and domain coverage of the system;

- do the users tend to demonstrate that they have specific *beliefs* about the task and the domain, which may create communication problems? It does not matter whether these user beliefs are right or wrong. If they tend to be present, they must be addressed in the way the dialogue is being designed;

- do the users tend to have specific *preferences* which should be taken into account when designing the dialogue? These may be preferences with respect to, for instance, dialogue sub-task order;

- will the dialogue, as designed, tend to impose any strong *cognitive loads* on users during task performance? If this is the case, the design may have to be changed lest the cognitive load makes the users behave in undesirable ways during dialogue. One way to increase users' cognitive load is to ask them to remember to use specific keywords in their interaction with the system; another, to use a feedback strategy which is not heavy-handed enough;

- other cognitive properties of users that the dialogue manager developer may want to know about include the *response package* phenomenon. For instance, users seem to store some pieces of information together, such as "from A to B". Asking them from where they want to go therefore tends to elicit the entire response package. If this is the case, then the dialogue manager should make sure that the user input prediction enables the speech and language layers to process the entire response package.

2.24 Implementation issues

(a) Architecture and modularity: There is no standard architecture for dialogue managers, their modularity, or the information flow between modules. Current dialogue manager architectures differ, among many other things, with respect to their degree of domain and task independence. The functionality reviewed above may be implemented in any number of modules, the modularity may be completely domain and task dependent or relatively domain and task independent, and the dialogue manager may be directly communicating with virtually

every other module in an SLDS, as illustrated in the Appendix. As to the individual modules, Finite State Machines may be used for dialogue interaction modelling, and semantic frames may be used for task modelling. However, other approaches are possible. The ones just mentioned are among the simplest approaches. The Appendix provides examples of dialogue manager architectures and descriptions of the flow of information among dialogue managers and other SLDS modules.

(b) Main task of the dialogue manager: If there is a central task which characterises the dialogue manager as a *manager,* it is the task of deciding how to produce appropriate output to the user in view of the dialogue context and the user's most recent input as received from the speech and language layers.

Basically, what the dialogue manager does in order to interpret user input and produce appropriate output to the user is to:

- *use the knowledge of the current dialogue context and local and global focus of attention it may possess to:*
- *map from the semantically significant units in the user's most recent input (if any), as conveyed by the speech and language layers, onto the sub-task(s) (if any) addressed by the user;*
- *analyse the user's specific sub-task contribution(s) (if any);*
- use the user's sub-task contribution to:
- *execute a series of preparatory actions (consistency checking, input verification, input completion, history checking, database retrieval, etc.) usually leading to:*
- *the generation of output to the user, either by the dialogue manager itself or through output language and speech layers.*

The dialogue management activities just described were discussed in Sections 2.12 - 2.20 and 2.22 - 2.23 above. The analysis of the user's specific sub-task contribution is sometimes called 'dialogue parsing' and may involve constraints from most of the elements in the speech input, language input, context and control layers in Figure 1. In order to execute one or more actions that will eventually lead to the generation of output to the user, the dialogue manager may use, parsing as in Verbmobil, an Augmented Transition Network as in Waxholm, or, rather similarly, decide to follow a particular branch in a node-and-arc dialogue graph as in the Danish Dialogue System.

As the dialogue manager generates its output to the user, it must also:

- *change or update its representation of the current dialogue context; and*
- *generate whatever constraint-based support it may provide to the speech and language layers.*

These dialogue management activities were described in Sections 2.6, 2.9, 2.11, and 2.21 above.

At a high level of abstraction, what the dialogue manager has to do thus is to apply sets of decision - action rules, possibly complemented by statistical techniques, to get from (context + user input) to (preparatory actions + output generation + context revision + speech and language layer support). For simple tasks, this may reduce to the execution of a transformation from, e.g. (user input keywords from the speech layer) to (minimum preparatory actions + dialogue manager-based output generation including repair meta-communication) without the

use of (input or output) language layers, context representations and speech and language layer support. Different dialogue managers might be represented in terms of which increased-complexity properties they add to this simple model of a dialogue manager. Waxholm, for instance, adds semantic input parsing; a statistical topic spotter ranging over input keywords; out-of-domain input spotting; two-level dialogue segmentation into topics and their individual sub-structures; preparatory actions, such as dialogue parsing including consultation of the topic history, and database operations including temporal inferencing; user-initiated repair meta-communication; a three-phased dialogue structure of introduction, main phase, and closing; an output speech layer; advanced multimodal output; and topic history updating.

(c) Order of output to the user: As for the generation of output to the user, a plausible default ordering could be:

(i) if system-initiated repair or clarification meta-communication is needed, then the system should take the initiative and produce it as a matter of priority;

(ii) even if (i) is not the case, the user may have initiated meta-communication. If so, the system should respond to it;

(iii) if neither (i) nor (ii) is the case, the system should respond to any contribution to domain communication that the user may have made;

(iv) then take the domain initiative.

In other words, in communication with the user, meta-communication has priority over domain communication; system-initiated meta-communication has priority over user-initiated meta-communication; user domain contributions have priority over the system's taking the next step in domain communication. Note that feedback from the system may be involved at all levels (2.19). Note also that the above ordering (i) through (iv) is *not* an ordering of the system states involved. As far as efficient processing by the dialogue manager is concerned, the most efficient ordering seems to be to start from the default assumption that the user has made a contribution to the domain communication. Only if this is not the case should (i), (ii) and (iv) above be considered.

(d) Task and domain-independence: The fact that dialogue management, as considered above, is task-oriented, does not preclude the development of (relatively) task independent and domain independent dialogue managers. Task and domain independence is always independence in some *respect* or other, and it is important to specify that respect in order to state a precise claim. Even then, the domain or task independence is likely to be limited or relative. For instance, a dialogue manager may be task independent with respect to some, possibly large, class of information retrieval tasks but may not be easily adapted to all kinds of information retrieval tasks, or to negotiation tasks. Modular-architecture, domain and task independent dialogue managers are highly desirable, for several reasons (cf. 2.4). For instance, such dialogue managers may integrate a selection of the dialogue management techniques described above whilst keeping the task model description and the dialogue model description as separate modules . These dialogue managers are likely to work for all tasks and domains for which this particular combination of dialogue management techniques is appropriate. The Daimler-Benz dialogue manager and the RailTel/ARISE dialogue manager are cases in point. Much more could be done, however, to build increasingly general dialogue managers. To mention just a few examples, it would be extremely useful to have access to a generalised meta-communication dialogue manager component, or to a domain independent typology of dialogue acts.

(e) Dialogue manager development tools: In the absence of a generic dialogue manager for the development task at hand, the second-best thing is likely to be a dedicated dialogue

specification language. Several tools exist for the representation and implementation of dialogue managers. DDLTool is a graphical editor which supports the representation of dialogue management software in the Dialogue Description Language (DDL). CSLUrp is a graphical rapid prototyping environment which in many respects is similar to DDLTool and includes a visual programming language. CSLUrp is a major part of the OGI toolkit. A third example of a tool for representing dialogue management is HDDL [Aust 1996]. In contrast to DDLTool and CSLUrp, HDDL supports textual dialogue representation. A fourth example is GADL/IPSIM [Smith and Hipp 1994] which is a Prolog-based task-oriented dialogue specification language. Common to these languages is that they are event-based and have a range of primitive operations that support the speech and language layers. GADL/IPSIM moreover has support of domain representation. It remains, however, an open question to which extent primitives of any of the languages just mentioned, scale up beyond relatively simple, well-structured tasks.

If the language primitives needed are not provided by a specialised dialogue specification language, it may often be preferable to use a general programming language, such as Lisp, Prolog or Java, which usually also means that the resulting software will be easier to port. Various development environments exist for general programming languages, which may then be used to facilitate dialogue manager development.

3. Dialogue Management Life-Cycle Model

This chapter introduces the DISC life-cycle model for eliciting information on current practice in dialogue manager development and evaluation.

At a high level of abstraction, any standard software engineering life cycle model applies to the development and evaluation of dialogue managers. However, as such models are aimed at describing software development processes in general, they do not specialise to the development and evaluation processes which are specific to particular classes of systems or system components, such as SLDSs or dialogue managers. In addition, general software engineering life cycle models do not include advise on the methods and tools to be used when developing, e.g., dialogue managers.

Figure 1 shows a general software engineering life cycle model which has been slightly specialised to the development and evaluation of SLDSs and their components. The figure shows an overall framework for the development and evaluation process through to installation at the user's site. After this point there may be maintenance of the software, the software may be ported to other platforms, or it may be adapted to new applications in which case a new life cycle starts.

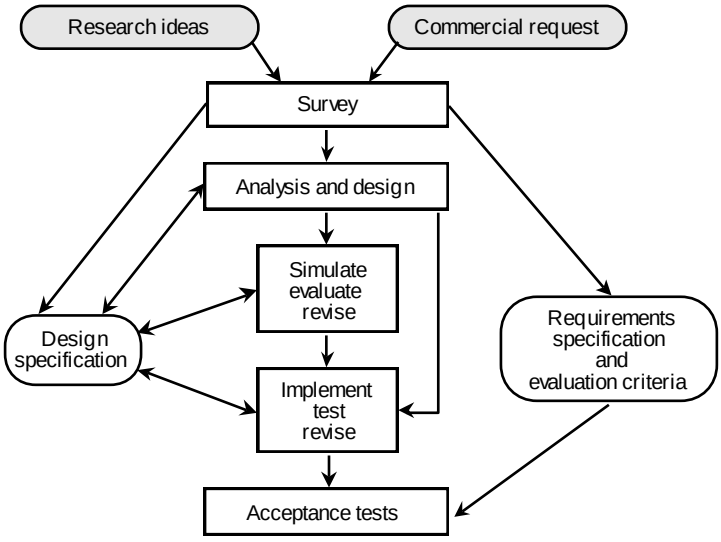


Figure 1. A software engineering life cycle model for the development and evaluation of SLDSs and components. Rectangular boxes show process phases. The development and evaluation process is iterative within each phase and across phases. Arrows linking phases indicate the overall course of the process. The requirements specifications and evaluation criteria, and the design specification (rounded white boxes) are used throughout the development process. Rounded grey boxes indicate that the system being developed may be either a research system or a commercial system.

The DISC dialogue engineering life cycle model draws on a general software engineering life cycle model but is specialised to capturing the process of developing and evaluating SLDSs and components. The DISC life cycle model asks a series of fairly detailed questions which are all related to the model in Figure 1, go beyond that model into, e.g., maintenance, and are specialised to dialogue engineering.

The DISC life cycle model is intended to be used no matter if one describes an entire SLDS or a single component. This is possible because each piece of software has a life cycle and most life cycle issues are relevant independently of the specific nature of the software. The precise contents of the questions to ask on each issue may, however, depend on specific nature of the software. For example, the issue of evaluation is highly relevant to both speech recognisers and dialogue managers. However, the precise tests to carry out are very different. In the following we address the DISC life cycle issues with special emphasis on dialogue management.

3.1 Overall design goal(s)

What is the general purpose(s) of the design process?

Dialogue managers have so far typically been designed as part of a particular (commercial or research prototype) SLDS. Probably for this reason focus has rarely been on dialogue management for its own purpose rather than on dialogue management as one of several components which have to work together to form an SLDS. In some cases focus has actually not been on dialogue management at all. The dialogue manager was built only because it is a necessary part of an SLDS and hence a necessary evil if one wants to study, e.g., speech recognition in an SLDS environment. However, there are now ongoing efforts world-wide to develop dialogue managers which are more general-purpose and which may fairly easily be adapted to a new task domain, such as the Daimler-Benz dialogue manager. In addition to the overall purpose of having a dialogue manager, there may be several different design goals for the dialogue manager, such as to explore a particular approach to dialogue management, study certain aspects of the dialogue management component, ensure multilinguality or focus on modularity in the case of distributed development.

3.2 Hardware constraints

Were there any a priori constraints on the hardware to be used in the design process?

Typically there are no a priori hardware constraints on the dialogue manager. The hardware is more likely to be determined by, e.g., what is available at the sites involved, economical considerations as regards the procurer of the SLDS in question, or performance demands on the system (e.g. real-time).

3.3 Software constraints

Were there any a priori constraints on the software to be used in the design process?

Since dialogue managers so far typically have been developed as one-of-a-kind, software constraints have not been dictated by already existing pieces of software which should function as (part of) the dialogue manager. If a dialogue manager already exists and will be adapted to a given application, the only constraint is that the programming language in which it is written must be able to run on the platform on which the dialogue manager will be adapted or further developed.

3.4 Customer constraints

Which constraints does the customer (if any) impose on the system/component? Note that customer constraints may overlap with some of the other constraints. In that case, they should only be inserted once, i.e. under one type of constraint.

For research projects there are typically no real customer. At best the research group has contact to an organisation which potentially could be a real procurer or end-user and which is willing to give relevant information and feedback to the project. The dialogue manager must be able to handle the task(s) specified and thus needs access to relevant task domain information, and it must possess the rules relevant for handling the information. For commercial projects this sometimes causes problems. In some cases, companies are not interested in revealing internal data that is basic to their offer, and the system developers may thus not have access to the entire task structure. This means that part of the task model and the inferences involved have to be made external to the dialogue manager. Other customer constraints may relate to resources, the type of interface, software, protocols, and hardware connections. Currently, there is a strong trend towards PC-based solutions.

3.5 Other constraints

Were there any other constraints on the design process?

Several constraints other than hardware, software and customer constraints may influence the dialogue manager life cycle. Precisely what to mention under other constraints depends on what has already been mentioned under the above constraint categories. For example, cost and development time will often be customer constraints. Other examples of constraints are personpower, development phases, standards conformation, and knowledge in the developer team.

3.6 Design ideas

Did the designers have any particular design ideas which they would try to realise in the design process?

The design ideas underlying the dialogue manager design are typically related to the overall design goal. Thus, if the focus is *not* on dialogue management, the idea may simply be to create a dialogue manager which works sufficiently well to serve as part of an entire SLDS, allowing investigation of issues not directly related to dialogue management. If dialogue management is a focal issue, examples of design ideas could be to create a generic dialogue manager, to explore modularity, to explore co-operativity, or to experiment with different ways in which to control dialogue.

3.7 Designer preferences

Did the designers impose any constraints on the design which were not dictated from elsewhere?

Designer preferences may influence the development process by introducing new constraints. Such preferences may relate to many different issues, such as preferred development platform, preferred programming language, or positive experience in doing things in a certain way and following certain development methodologies. For example, there may be a preference for using rapid prototyping in developing the dialogue manager in order to produce early demonstrations.

3.8 Design process type

What is the nature of the design process?

The dialogue manager design process type has so far mainly been exploratory research, either with a focus on dialogue management or on some other SLDS component of which the dialogue manager forms part. For dialogue managers which form part of commercial systems, the design process type has mostly been that of one-of-a-kind product development. In the cases where an already existing dialogue manager has been used in a new application, the design process type has been, e.g., adaptation, optimisation or redesign.

3.9 Development process type

How was the dialogue manager developed?

Dialogue managers, in particular in research projects, have often been developed through Wizard of Oz (WOZ) prototyping. WOZ is a relatively costly development method. On the other hand, by producing data on the interaction between a (fully or partially) simulated system and its users, WOZ provides the basis for early tests of, i.a., the dialogue model and its feasibility, as well as of the coverage and adequacy of requirements prior to implementation. It is fairly costly to develop dialogue managers, and in research projects dialogue managers are typically rather complex and meant to explore new areas, thus running a high risk that the first design is relatively far from what is technologically feasible or desirable from a user's point of view. In such cases it may be advisable to use WOZ. For simple dialogues and in most industrial settings, WOZ is normally replaced by an implement-test-and-revise approach based on emerging development platforms. Whether or not WOZ is preferable to implement-test-and-revise depends on several factors, such as the novelty of the development objectives relative to the skills, methods and tools available to the developers, the complexity of the interaction model to be designed, the task domain, and the risk and cost of implementation failure. Low complexity speaks in favour of directly implementing the interaction model without interposing a simulation phase, especially if the developers have built similar systems before. High complexity, on the other hand, may advocate iterative simulations.

3.10 Requirements and design specification documentation

Is one or both of these specifications documented? Describe the specifications.

Identified goals and constraints are represented in a *requirements specification* document. The purpose of this document is to eventually list all the agreed requirements which the envisaged dialogue manager should meet. As the dialogue manager is usually part of an SLDS, the requirements specification is normally made for the SLDS as a whole and not separately for the dialogue manager.

The purpose of the *design specification* is to describe and eventually make operational how to build a dialogue manager which will satisfy the requirements specification and meet the evaluation criteria. The design specification, therefore, will be clearly related to, and may simply include, the requirements specification.

Befitting their exploratory purpose, the framework for research systems development is often loosely defined compared to that of commercial systems. The requirements specification does not serve contractual purposes. This means that requirements can be more easily modified later in the development process because there is no procurer who has to approve of the changes

made.

The design specification must be sufficiently detailed to serve as a basis for implementation and, in contrast to the requirements specification, is often expressed in a formal or semi-formal language understood by the systems developers and serving its operational purpose.

To serve its purpose, the design specification must be constantly updated to include the most recent additions and revisions. If this is to be done systematically and coherently, an explicit representation of the changing and accumulating design decisions is needed for keeping track of the development process. This is good engineering practice but difficult to do. If it is not done, it becomes hard or even impossible to (i) keep track of the design decisions that have been made and why they were made, (ii) explain to new developers joining the team what is going on, and (iii) carry out informed maintenance and re-engineering once the project has been completed.

3.11 Development process representation

Has the development process itself been explicitly represented in some way? How?

Dialogue manager development processes differ widely with respect to the extent to which a development process representation has been made and how much of it is available in any systematic fashion. What is typically publicly available are bits and pieces found in scientific papers. Internal reports and working papers may contain more detailed representations of the dialogue manager development process, and meeting protocols, early drafts, diagrams, protocols from experiments and other internal documents may exist as well. Technical reports and working papers may be available from research projects whereas they are usually hard to obtain from commercial projects. However, internal papers and documents are usually inaccessible no matter if they are from research projects or from commercial projects.

The advantages of explicit development process representations are that these can be re-used, possibly in revised form, in new projects and with new developers coming on the team, and can support re-design and maintenance. The main disadvantage is that creating them represents an additional cost.

3.12 Realism criteria

Will the dialogue manager meet real user needs, will it meet them better, in some sense to be explained (cheaper, more efficiently, faster, other), than known alternatives, is the dialogue manager "just" meant for exploring specific possibilities (explain), other (explain)?

Dialogue managers are typically viewed as part of particular SLDSs. Most SLDSs have something to do with real user needs. However, to appropriately address real user needs, the development process often needs to include extended end-user contact, extensive work on domain delimitation, clear up-front performance evaluation criteria, final adequacy evaluation criteria, extended quantitative and qualitative evaluation throughout the development process, an explicit development methodology, etc. In research projects, focus is typically on exploration and realism criteria are not being considered hard constraints, that is, one may deviate from the realism criteria to the extent needed to achieve other, more important project goals.

For dialogue managers which form part of commercial applications, the realism criteria may be related to, for example, cost-savings, making the performance of a task faster and more efficient, or increased quality of service.

3.13 Functionality criteria

Which functionalities should the dialogue manager have (this entry expands the overall design goals)?

The overall function of a dialogue manager is typically to control the dialogue performed by an SLDS. In addition, several other functionalities may be in focus. Examples of functionality criteria are that the dialogue manager should be language independent, be modular, have a sufficient coverage of the selected task domain, be able to assign an interpretation to the input it receives, be able to decide how the dialogue may best continue, and be able to plan an adequate system utterance whenever it is the system's turn to speak.

3.14 Usability criteria

What are the aims in terms of usability?

Since dialogue managers are usually embedded in SLDSs, there are two usability aspects to consider. One aspect is the usability criteria as seen from the *system developer's* point of view. For the system developer, usability criteria typically relate to issues such as how easy it is to embed the dialogue manager component into the SLDS, how easy it is to modify the component, and whether it will be able to run in different environments.

The second aspect is the *users'* point of view. Users will not experience the dialogue manager as a stand-alone component but only as part of an entire system. To a large extent, however, it is the dialogue manager which is responsible for how easy the SLDS is to use, whether it is able to perform a dialogue perceived by the users to be natural, flexible and robust within its domain, whether sufficient meta-communicative facilities are available, etc.

3.15 Organisational aspects

Will the dialogue manager have to fit into some organisation or other, how?

Only if the dialogue manager is developed as a (commercial) "stand-alone" component may organisational aspects be relevant. In this case, the dialogue manager must be useful to the organisation and fit its business programme. For research projects and projects focusing on entire systems rather than on the dialogue manager, organisational aspects are either not applicable or must be seen in relation to the entire SLDS. The purpose of an SLDS will often be to support, partially replace or improve a service which has traditionally been carried out by humans. Still, given the core role of the dialogue manager in SLDSs, organisational aspects may easily affect its design.

3.16 Customer(s)

Who is the customer for the system/component (if any)?

If a generic and stand-alone dialogue manager has been developed, the customer will be the organisation which makes use of the dialogue manager for new applications. So far, however, dialogue managers mostly have been components that were developed for specific SLDSs. For research prototypes there is usually no customer. At best there are contacts to potential customers who can provide input and feedback on the design. For industrial SLDSs, typical customers are telecoms, railway companies, banks and insurance companies, and companies that want, e.g., an automatic switchboard system. This situation is rapidly changing, however, towards a much larger variety of customers who, for instance, may want SLDSs for their Internet pages.

3.17 Users

Who are the intended users of the dialogue manager?

Direct users of dialogue managers are system developers using dialogue managers for SLDS applications. Indirect users (in the sense that they use the entire SLDS and not only the dialogue manager) are often the broad public. For example, the person calling a switchboard system, a telephone service or a train timetable information system may be anybody. Thus many SLDSs are walk-up-and-use systems requiring no particular skills in advance. The main restriction often is the language as SLDSs are not language independent but require the user to speak, e.g., French. The dialogue manager viewed as a separate component may be language independent but not the SLDS.

3.18 Developers

How many people took significant part in the development? What were their backgrounds?

Often, several developers are involved in dialogue manager development simply because of the complexity of the issues involved. The backgrounds of the developers may vary considerably, including engineering, computer science, linguistics and psychology. The reason for this may be found in the fact that dialogue management development can hardly be called a well-established discipline since the area is fairly new. Moreover, it draws on many different resources and skills, including, e.g., knowledge of user behaviour, linguistic knowledge and implementational skills. Thus it may be quite practical to have people with different backgrounds on the development team.

3.19 Development time

When was the dialogue manager developed? What was the actual development time for the component (estimated in person/months)? Was that more or less than planned? Why?

Development of anything but the simplest dialogue manager is fairly time consuming. In many research projects a dialogue manager is being developed over a three year period which is the standard duration of a project. However, much more than three person/years of effort is spent on the development. So far, developers of dialogue managers for advanced SLDSs have had little previous experience to draw upon because the area is so new and there are no guidelines on how to do it efficiently. This means that the development of a dialogue manager includes much trial-and-error experimentation and comparatively little routine-based work.

Dialogue managers for simple systems which are system-driven and based on single word recognition, such as some telephone service systems and switchboard systems, are of course much easier to construct and thus much less time consuming than those for advanced SLDS taking spontaneous spoken input.

3.20 Requirements and design specification evaluation

Were the requirements and/or design specifications themselves subjected to evaluation in some way, prior to dialogue manager implementation? If so, how?

There is often no systematic and explicit evaluation of the requirements and design specifications. In research projects these documents often only exist in some rudimentary form but even in industrial projects where such documents do exist, evaluation is typically not being done. The dialogue manager has to be able to manage the appropriate dialogue structures and otherwise satisfy the specified constraints. However difficult this may be to do in any formal way, it is essential to good development practice to carry out a systematic and explicit

evaluation of whether the goals and constraints are reasonable, feasible and non-contradictory.

3.21 Evaluation criteria

Which quantitative and qualitative performance measures should the dialogue manager satisfy?

In addressing the issue of dialogue manager evaluation, it is important to keep two different situations in mind. The first one concerns evaluation of the SLDS of which the dialogue manager forms a part. The second concerns evaluation of the dialogue manager itself. Dialogue manager evaluation takes place in both situations. However, the first situation obviously makes it harder to exactly distinguish between those aspects of the SLDS's performance which are due to the dialogue manager and which aspects are due to the performance of other system components. Poor speech recognition, for instance, can only to a certain extent be counter-balanced by good dialogue manager design. If the speech recognition is too poor, the users will walk away even if they are dealing with a brilliant dialogue manager. The second situation may be one in which the dialogue manager is being evaluated as part of SLDS development or it may be one in which the dialogue manager itself is the sole development target. In both cases, the evaluation criteria involved are likely to be much more dialogue manager-specific than those involved in evaluating the SLDS as a whole. The reader is encouraged to keep these different possibilities in mind during what follows.

The evaluation criteria that will be used in evaluating the final dialogue manager should in principle be specified along with the requirements specification. The evaluation criteria state the parameters that should be measured or otherwise evaluated and the results that should be achieved for the final dialogue manager to be acceptable. For instance, if the requirements specification includes a quantitative requirement on dialogue transaction success as achieved under certain specified circumstances, then dialogue transaction success should be included among the evaluation criteria and will be among the parameters measured during evaluation of the final dialogue manager.

Only a few years ago, the field of dialogue management was still so new that evaluation criteria did not exist at all, i.e., nobody had really thought about what to test and no experience from previous development efforts was available. Evaluation criteria were invented ad hoc when the dialogue manager and the SLDS in which it was embedded came up for evaluation. This way of doing things it is still quite common, in particular in research projects, but cannot be recommended because it means that developers have little support during development in terms of criteria that the dialogue manager should fulfil in order to be considered satisfactory and acceptable. The definition, from early on in the development process, of clear, relevant and appropriate evaluation criteria, and the continuous and methodologically sound evaluation of progress with reference to those criteria, should be main characteristics of dialogue manager development and evaluation.

There is still no well-defined set of evaluation criteria to draw upon. Most criteria in current use are purely quantitative. These may be relatively easy to apply but provide insufficient information on quality. Qualitative measures, on the other hand, are difficult to apply when, e.g., they are related to subjective opinions. However, in some cases performance parameters which can be measured quantitatively are also able to express quality. For example, the number of dialogue interaction problems can be measured quantitatively but at the same time the resulting figure expresses something important about the quality of the dialogue manager.

For industrial projects, the main evaluation criteria may be related primarily to cost. The really interesting figures are, e.g., the cost-per-ticket-sold, the cost-per-contract-signed, or the cost-

per-customer-informed. These quantities can be measured for both the human-human and the human-machine version of the dialogue. They capture all aspects of the dialogue because an inadequate system will only get considerably less tickets sold, contracts signed, customers informed or whatever is the task of the system, meaning less customer acceptance. However, these criteria are difficult or impossible to reliably apply before the system has been deployed in the field.

The list below shows a series of evaluation criteria that are relevant to dialogue manager evaluation. Whilst not all of them are applicable to all systems, each criterion may help throw light on the adequacy and performance of some dialogue managers. They need not all be included in the "official" list of evaluation criteria stated in the requirements specification. Even if not included, they may still be relevant for evaluating how good or bad the dialogue manager is and what progress is being made during its development. The list is a long one but, apart from a single criterion, i.e. the one about multimodal input fusion, all the criteria on the list could be applied to the dialogue managers analysed in the Appendix.

- Adequacy of (non-) distinction between novice and expert users.
- Adequacy of dedicated processing of ellipsis.
- Adequacy of domain inferences.
- Adequacy of dialogue manager support for the speech and/or language layers.
- Adequacy of information to the users on how to interact with the system.
- Adequacy of information to the users on the system's domain and task coverage.
- Adequacy of initiative distribution among user and system relative to the task.
- Adequacy of multimodal input fusion, i.e. of combining more or less simultaneous input messages expressed in different modalities into a single semantic representation or sub-task contribution.
- Adequacy of output distribution over speech and other output modalities in multimodal SLDSs.
- Adequacy of operator fallback strategy.
- Adequacy of strategy for identifying and responding to out-of-task and/or out-of-domain input.
- Average time for task completion as compared to other ways of solving the same task.

- Complexity of the interaction model expressed, e.g., in terms of number of nodes if a graph representation is used.
- Conformance of system phrases to the cooperativity guidelines (individually as well as in context).
- Co-reference interpretation success.
- Cost per transaction as compared to other ways of solving the same task.
- Database information sufficiency.
- Degree of utilisation of the knowledge sources available to the dialogue manager.
- Dialogue segmentation adequacy.
- Ease of maintenance/modification of the dialogue manager and/or of its individual modules.

- Feedback strategy sufficiency: information feedback.
- Feedback strategy sufficiency: processing feedback.
- Indirect speech acts interpretation success.
- Naturalness through mixed initiative dialogue.
- Number of dialogue interaction problems.
- Number of turns to complete a task.
- Real-time performance.
- Relevance and success of predictions.
- Re-usability of the dialogue manager and/or of its individual modules.
- Robustness - wrt. error loops and graceful degradation.
- Robustness - wrt. unexpected (user) deviations from the dialogue plan.
- Sub-task or topic identification success.
- Sufficiency of dialogue histories (linguistic, topic, task, performance).
- Sufficiency of meta-communication facilities: system-initiated repair, system-initiated clarification, user-initiated repair, user-initiated clarification.
- Task and domain model coverage: Sufficient? Delineated in a principled and intuitive way?
- Transaction success.
- Translation success of spoken language translation (support) systems.
- User model adequacy.
- User satisfaction and other subjective parameters (explain).

The list provides a good illustration of the complexity of dialogue manager evaluation, reflecting the core "hidden" role of the dialogue manager in SLDSs. Few of the evaluation criteria in the list are straightforwardly quantitative, at least for the time being. And some of the criteria which are quantitative, cannot be applied to the SLDSs performance as a whole but must be applied diagnostically, for instance by inspecting interaction logfiles to see whether, e.g., the dialogue manager adequately supports the performance of correct co-reference resolution or correct ellipsis processing. Very many of the criteria are qualitative. Some criteria, and not the least important ones, are subjective and must be measured through interviews, questionnaires and other contacts with the users to elicit those among their subjective impressions from interacting with the system that may be attributed to the workings of the dialogue manager.

The reason for the existence of so many qualitative criteria on the list is the highly contextual nature of most dialogue managers. A good meta-communication strategy, or a good feedback strategy, for instance, may solve problems arising from the insufficiency of other elements of the dialogue manager or of elements in the speech or language layers as well. If, for instance, the system lacks barge-in, the dialogue manager may have to be designed differently from its design for a similar SLDS which does have barge-in. Similarly, criteria involving terms such as 'adequacy' and 'sufficiency' often conceal one or several implicit conditions, such as 'relative to the task' or 'relative to the intended users'.

Furthermore, the list is incomplete in at least one important respect: it does not include the, often quantitative, criteria which may derive from particular constraints on a given SLDS, such

as that the average user utterance length should be, say, four words maximum. To include such constraints would make the list unwieldly and overly specific. In developing one's dialogue manager, one may measure many different things which it would make no sense to measure in connection with another dialogue manager development project which is subject to very different constraints.

As it stands, the list may be used as a checklist by dialogue manager developers in four iterations, so to speak, as follows:

- in the *first iteration,* the developer selects the criteria which are relevant to the particular dialogue manager to be developed;

- in the *second iteration,* the developer makes the selected criteria more specific and applicable by making explicit the implicit conditions that apply to the development task at hand;

- in the *third iteration,* the developer plans when to apply the specified criteria during the development process;

- finally, in the *fourth iteration,* the criteria are applied in a methodologically sound manner as planned.

Note that the above four-stage model can be used for 'meta-evaluation' of dialogue manager development processes. Central questions to ask during meta-evaluation are:

- did the developers select all the right evaluation criteria?

- did they make these criteria sufficiently specific to their development task?

- did they apply the criteria correctly at all the development stages at which they should have been applied?

- what were the results?

- did the developers take adequate action in view of the results?

The next section will pick up on this point.

3.22 Evaluation

At which stages during design and development was the dialogue manager subjected to testing/evaluation? How (type of test, number of users, etc.)? Describe the results. How was data collection? Was data annotation done? How? Which information has been extracted from the data? What were the results? Is test material/data/test suites (and/or a description of the test conditions) available? Can the test be replicated? Can anybody perform the tests? Is anything stated about comparability of the test(result)s with those of other systems/components of similar scope?

Evaluation is very important and is/should be tightly interwoven with systems development. Evaluation is constantly needed throughout development to measure progress towards the goals which the dialogue manager has to meet. However, evaluation of dialogue managers as well as of entire SLDSs is today as much of an art and a craft as it is an exact science with established standards and procedures for good engineering practice. There is not even consensus on terminology in the field.

Distinction may be made among three types of evaluation which, although clearly not orthogonal, seem to cover the relevant aspects of evaluation and subsume the scopes of other commonly used terms and distinctions. Each of these three types of evaluation may be used at

any stage during dialogue manager development:

- *performance evaluation*, i.e. measurements of the performance of the dialogue manager and its components in terms of a set of quantitative parameters;
- *diagnostic evaluation*, i.e. detection and diagnosis of design and implementation errors;
- *adequacy evaluation*, i.e., how well do the dialogue manager and its components fit their purpose and meet actual user needs and expectations.

Other common terms are 'blackbox' and 'glassbox' tests, and 'progress evaluation'. Blackbox and glassbox tests may be considered forms of diagnostic evaluation but these tests are carried out on implemented components or systems only (se further below). *Progress evaluation* compares two iterations of the same system during development and is a kind of performance evaluation. Progress evaluation is typically used to compare, e.g., Wizard of Oz iterations of a dialogue manager.

Performance, diagnostic and adequacy evaluation should be performed as integral parts of the development process to measure progress towards satisfaction of the requirements specification, evaluation criteria and design specification.

Performance evaluation is made throughout the development process with more or less the same emphasis from one iteration to another.

Diagnostic evaluation is of central importance in the early development process but should require less effort in the final phase by which time most errors should have been removed. During debugging of the implemented dialogue manager, two typical types of test to carry out are the glassbox tests and the blackbox tests. There is no general agreement on the definitions of glassbox and blackbox tests. By a *glassbox test* we understand a test in which the internal system representation can be inspected. The test should ensure that reasonable test suites, i.e. data sets, can be constructed that will activate all loops and conditions of the program being tested.

In a *blackbox test* only input to and output from the program are available to the evaluator. How the program works internally is made invisible. Test suites are constructed on the basis of the requirements specification and along with a specification of the expected output. Expected and actual output are compared when the test is performed and deviations must be explained. Either there is a bug in the program or the expected output was incorrect. Bugs must be corrected and the test run again. The test suites should include fully acceptable as well as borderline cases to test if the program reacts reasonably and does not break down in case of errors in the input. Ideally, and in contrast to the glassbox test suites, the blackbox test suites should not be constructed by the system programmer who implemented the system since s/he may have difficulties in viewing the program as a black box.

Adequacy evaluation of SLDSs typically includes a few general key performance measurements as well as measurement of user satisfaction. If we are talking about adequacy evaluation of a dialogue manager *per se,* there is a need for much more specific evaluation criteria. Adequacy evaluation is used mostly in the later phases of development. This is because a number of adequacy aspects cannot be tested in a sensible way until an implemented and debugged dialogue manager is available for the purpose. For instance, it does not make sense to measure real-time performance on a simulated system.

Another useful distinction is the distinction between objective evaluation and subjective evaluation. *Objective evaluation* addresses objectively measurable performance parameters. *Subjective evaluation* addresses the opinions which users have formed after using the dialogue

manager as imbedded into an SLDS. Performance evaluation and diagnostic evaluation are forms of objective evaluation whereas adequacy evaluation includes both objective and subjective evaluation.

In addition to distinctions between different types of evaluation such as the above, distinction may be made between different types of tests. Test types refer to aspects of the context of the evaluation, such as the users involved, whether scenarios are being used or not, or whether the system being tested is an implemented one or a simulated one. The tests to be mentioned below are controlled tests, field tests and acceptance tests. Roughly speaking, controlled tests are performed during simulation and often also after implementation; field tests are performed after implementation and towards the end of systems development; and the acceptance test is the final test of a system. This notwithstanding, a test may be carried out as a controlled test or as a field test no matter if the system is being simulated or not. Each test typically includes all of the three evaluation types mentioned above (i.e. performance, diagnostic and adequacy evaluation).

In a *controlled test,* the users need not be those who will actually use the final system. However, it is recommended to select the test subjects from the target user group to ensure that they have a relevant background. In the controlled test, the tasks to be carried out (the scenarios) should not be selected by the participants. To ensure reasonable representativity of scenarios with respect to system functionality and task domain coverage, and to bring the controlled test as close to benchmarking as possible, scenario selection should ideally be done by an independent panel according to guidelines on, i.a., who should select the scenarios, their coverage of system functionality and task domain, the number of scenarios per user and the number of users. The panel should include end-users as well as system developers. A *field distribution problem* attaches to all results of controlled tests. The frequency of different tasks across the domain of application may be different in real life from that imposed in the controlled test. This may significantly affect the frequency of the interaction problems encountered in the test.

In a *field test,* the dialogue manager is being tested by real end-users in their appropriate environments. This means that the tasks carried out will be real-life tasks which, however, may not necessarily be representative of the full range of system functionality unless the duration of the field test is very long. The field test option will not always be available for research systems due to the missing customer. It may be preferable to carry out a controlled test before a field test because the controlled test will allow an evaluation which is close to benchmarking.

The *acceptance test* is the final test of the dialogue manager and perhaps also of the system in which it is embedded, before it is accepted for operational use. The test aims to demonstrate that the requirements in the contract (the requirements specification) have been satisfied and the evaluation criteria met. Often the dialogue manager is tested with data supplied by the procurer or in a set-up specified by the procurer. Detected errors must be corrected immediately. In case of larger disagreements with, or omissions in, the requirements specification, developer and procurer must discuss what to do. In the worst case the procurer may turn down the product if the dialogue manager or system does not meet the requirements agreed upon. However, it is not always solely the system developer's fault that the dialogue manager does not exhibit the performance and functionality anticipated by the procurer. In such cases, procurer and developer must negotiate what to do in order to reach an agreement.

3.23 Mastery of the development and evaluation process

Of which parts of the process did the team have sufficient mastery in advance? Of which parts didn't it have such mastery?

In many projects, developers of dialogue managers have little mastery of dialogue management because the field is fairly new and the project itself is meant to serve as a competence-building exercise. However, this does not prevent a team from having good skills in software engineering and other general issues relevant to dialogue management. Other teams have worked extensively on dialogue management and have obtained a good deal of experience in the field.

3.24 Problems during development and evaluation

Were there any major problems during development and evaluation? Describe these.

Problems during development and evaluation may be of many different kinds and both human and technical. Examples are that more than one site is involved in dialogue manager development, which may cause problems in collaboration or technical problems in making the parts fit together; the developers have little experience in dialogue management; one or more developers left the team, resulting in delays; the requirements specification was not properly evaluated to begin with and turns out to be (partially) infeasible, resulting in re-specification and redesign of the dialogue manager.

3.25 Development and evaluation process sketch

Please summarise in a couple of pages key points of development and evaluation of the dialogue manager. To be done by the developers.

This entry is intended to provide a brief overview of the dialogue manager development and evaluation process. The sketch may include information on how requirements were established, how a dialogue model was developed and evaluated, implementation issues, and tests done on the dialogue manager. On the issue of evaluation, the sketch could draw extensively on the list of criteria presented in Section 3.22, stating as well to which extent the four-stage approach to criteria selection and application has been followed.

3.26 Component selection/design

Describe the dialogue manager and its origin.

Off-the-shelf dialogue managers hardly exist yet. Most dialogue managers are built in-house for particular applications and as part of an SLDS.

3.27 Maintenance

How easy is the dialogue manager to maintain, cost estimates, etc.

Maintenance costs for dialogue managers are usually low. When research projects are finished, time may still be spent on keeping the SLDS running for demonstration purposes, but that is all. Also for commercial dialogue managers, maintenance costs seem to be at a minimum. Rather, resources may be spent on modifications, additions and customisation, cf. below.

3.28 Portability

How easily can the dialogue manager be ported?

Basically, a dialogue manager only requires that the language in which it is being developed

can run on the platform to which it is being ported. However, the SLDSs of which a dialogue manager is typically a component, may impose much more severe restrictions on portability deriving from its other components.

3.29 Modifications

What is required if the dialogue manager is to be modified?

This question may be difficult to answer in general since what is required depends on the type of modification. Small modifications compatible with the lines along which the dialogue manager has been developed already, may be fairly easy to implement, such as slight modifications of some sub-task. Larger modification, on the other hand, can be very difficult to do, such as the addition of a new task. Modification difficulty depends on how generic the dialogue manager is relative to the modifications proposed.

3.30 Additions, customisation

Has customisation of the dialogue manager been attempted or carried out? Is there a strategy for resource updates? Is there a tool to enforce that the optimal sequence of update steps is followed?

Most dialogue managers are built as one-of-a-kind to fit into a specific SLDS. This means that the possibility of additions and customisation has not necessarily been considered. However, dialogue managers do exist which are intended for adaptation to new tasks. As a minimum, when being adapted to a new service, the dialogue manager will need a model of the new task(s), a model of the domain and a model of the new dialogue. If the dialogue manager is language independent, it should be relatively easy to switch to another language. Moreover, if the knowledge bases are kept separate it will also be fairly easy to adapt the dialogue manager to using the new ones.

3.31 Property rights

Describe the property rights situation for the system/component.

The property rights to the dialogue manager typically belong to the site(s) where the component was developed but, of course, other arrangements may exist.

3.32 Documentation of the design process

Please include documents from the design process which illustrate the process and facilitate and support its understanding.

Documentation of the design process includes any document which may throw light on the dialogue manager design process, including requirements specification and design specification (no matter how fragmented these specifications may be), meeting protocols and minutes, design sketches and diagrams, scenarios, transcriptions, task analyses, etc.

3.33 References to additional dialogue manager documentation

Please include references to any background material in which information on the dialogue manager can be found.

References may be to conference papers, technical reports or any other material which contains information on the dialogue manager. □

4. Conclusions and Future Work

The present paper is a working paper, even a draft working paper. Its descriptive apparatus has proved sufficient for capturing all of the available details on properties, development and evaluation of five rather diverse state-of-the-art dialogue managers, and all or most of those details have been illustrated in chapters 2 and 3 above. But there is much more work to do. We need to discuss the results in the DISC consortium in order to arrive at a stable version of the findings. We need to "mine" the results for generalisations that can be used to extend current theory of dialogue management. We need to expand the normative aspects of the present synthesis on dialogue management towards a first draft best practice model of dialogue management development and evaluation. We need to test the draft on novel, and different, exemplars from industry and research as well as on the developers who represent the target audience for the DISC results. And finally, we need to package the final DISC best practice methodology for dialogue management and evaluation for optimised usability by developers, for instance by turning the methodology into a decision support tool or a hypertext/hypermedia design support system.

5. Acknowledgements

The work presented was done in the Esprit Long-Term Concerted Action No. 24823, DISC (1997-1998): Spoken Language Dialogue Systems and Components: Best practice in development and evaluation. The support is gratefully acknowledged. We also want to warmly thank the DISC partners who have demonstrated their systems and provided detailed information for the grid and life cycle analyses.

6. References

Bernsen, N. O., Dybkjær, H. and Dybkjær, L.: *Designing Interactive Speech Systems. From First Ideas to User Testing.* Springer Verlag 1998.

Williams, D. (Ed.): *Working Paper on Human Factors Current Practice*. DISC Draft Deliverable D1.6. Vocalis Ltd., 1998.

7. Appendix. Exemplar Dialogue Manager Grids and Life-Cycles

7.1 Daimler-Benz Dialogue Manager Grid

Laila Dybkjær and Niels Ole Bernsen

The Maersk Institute, Odense University
5230 Odense M, Denmark
laila@mip.ou.dk, nob@mip.ou.dk

1. Introduction

This paper presents an analysis of the Daimler-Benz dialogue manager (D-B DM). The analysis is presented in the form of (a) a 'grid' which describes the system's properties with particular emphasis on dialogue management, (b) a life-cycle model which provides a structured description of the system's development and evaluation process, and (c) supporting material, such as component architecture, dialogues and references.

The presented information will be cross-checked with the developers of D-B DM as well as with the complementary descriptions of other aspects of the D-B system provided by the DISC partners. These other descriptions address language understanding and generation, done by IMS.

Demonstrator: Available in Ulm

Developer: Daimler-Benz

Contact: Paul Heisterkamp

System/component identifier	
The Daimler-Benz dialogue manager is a stand-alone module. Basically it origins from the Sundial project. The task domains of the Sundial systems were train timetable information and flight information. However, the dialogue manager has later been used in four other (research as well as commercial) applications with continuous speech input. These four systems include Access which is on automation of call centres for personal intensive applications in insurance; STORM which is an application that provides digital road map updates with dynamic information on closed streets, ongoing work, traffic jam, etc. used for regional traffic management in the Stuttgart area; two internal projects the contents of which are not publicly available. URL: Not available.	
System performance	
Cooperativity	Yes.
Initiative	Mixed.
Influencing users	Yes, if the uninfluenced stuff does not work.
Real-time	Yes.
Transaction success	80%+ (overall application).
General evaluation	No ISO standards or other general methods used.

Speech input	
Nature	Continuous, spontaneous speech.
Device(s)	Telephone, microphone.
Phone server	Yes.
Acoustic models	SCHMM.
Search	A*.
Vocabulary	5000; good performance for up to 10000 words but not used in any application yet.
Barge-in	No.
Word hypotheses	Yes, Verbmobil word hypothesis graph.
Grammar	Statistic language model or finite state model over categories.
Prosody	-
Speech output	
Device(s)	Loudspeaker, telephone.
Language(s)	German.
Input	Files (for pre-recorded speech), text strings for TTS.
Lexicon	No relation with parser or recogniser lexicon.
Sound generation technique	Coded or parametric, e.g. formant synthesiser; PSOLA from Verbmobil.
Prosody	Yes (for coded as well as for parametric).
Pronunciation description units	-
Flexibility	-
Miscellaneous	-
User utterances	
Lexicon	Up to 5000 full forms are used in applications.
Grammar	Lexical grammar approach based on the theory of Unification Categorial Grammar. The amount of grammar rules is restricted to a few basic rules of combination. Lexical entries are represented as complex feature structures which are combined by simple unification. Feature structures (or signs) consist of morphologic, syntactic and semantic attribute fields. There are two types of sign: basic signs without arguments and functor signs with a list of arguments. Semantic representation is constructed simultaneously with the syntactic derivation by stating co-references between syntactic and semantic attributes in the functor sign. Also other grammar types can be used: LAUG (left association unification grammar) and PSG (phrase structure grammar).

Parsing	Word graph parser. Word graphs are directed acyclic graphs. Each edge is labelled with a scored hypothesis and each node with a point in time. There are no gaps or overlaps in word graphs. A word graph has a single start node and a single end node. Each path from the start node to the end node forms a possible sentence hypothesis. Scores are positive numbers which assign a (pseudo-)probability measure to a word hypothesis. 0 is the best score. The A* algorithm is used for search and integrated in an agenda driven chart parser whose initial edges are built from word hypothesis. The agenda is initialised with seed entries ordered with respect to a heuristic score assignment. Seed definitions are provided by top-down predictions from the dialogue management component. The initial ordering on the agenda which combines heuristics with contextual knowledge implements an island parsing strategy. The parser terminates with the first complete hypothesis. If no complete hypothesis can be found, a robust parser delivers partial solutions. The resulting syntactic and semantic descriptions are passed to the dialogue level. Parsing of multiple results between the linguistic level and the dialogue level is allowed.
Style	-
Semantics	SIL (Semantic Interface Language) is used for representing the surface semantics of an utterance as one or more independent structures.
Discourse, context	Context is used in terms of predictions.
System utterances	
Generation	Separate module. Simple filling of sentence template.
Lexicon	-
Grammar	-
Semantics	-
Style	-
Processing	-
Discourse, context	-
Multimodal aspects	
Device(s)	-
None (unimodal system)	Yes.
Non-speech input	-
Non-speech output	-
Role(s)	-
Attentional state	
Focus, prior	Expectations (see below) as well as knowledge of the surface

	structure of the previous system utterance are taken into account to reduce the search space. They are used to focus search and decide where to start.
Sub-task identification	The contextual interpretation (belief module) anchors the utterance parts received from the linguistic analysis in their order in the surface semantic representation of the discourse universe. Anchoring means instantiation of possible discourse objects through the surface SIL expression. A part is anchored into the semantic context spanned by the preceding question or spanned by the part(s) already anchored. The belief module also interacts with the task module on sub-task identification. Linguistic markers, such as topic shift markers, are not used.
Expectations	Predictions are being used in relation with the recogniser. A set of dialogue goals is mapped onto a dialogue state descriptor. From this a language model is chosen. Semantic predictions are used in the interaction between contextual interpretation and linguistic analysis. A semantic prediction is a partially instantiated structure of surface semantic descriptions or a list thereof. It can specify the expected contents of an utterance to be one of a set of lexical entries (downward prediction: e.g. yes or no or quasi-synonyms of these). Or it can specify the expected contents to be of a certain class of lexical entries which have the same semantic description or a certain class of semantic structures which can fill the same roles in larger structures (upward prediction: e.g. city + preposition 'to'). The parser uses the predictions to start the search of the word graph at nodes fulfilling predictions. Moreover it can check partial results against the predictions to enhance the speed of parsing and to guide the parsing in that direction. Semantic predictions are presented in SIL and handled by the linguistic interface. Predictions for the recogniser are just language model numbers.
Intentional structure	
Task(s)	The dialogue manager is fairly task-independent. In the German Sundial system in which it was first used, the task was train timetable information. However, the dialogue manager is being used in several (research as well as commercial) applications with continuous speech input. These include information providing applications like train time table and flight enquiry systems, and information seeking applications like road map update for long term and short term modifications used for regional traffic management in the Stuttgart area, and telephone-based applications for direct insurance and call management.

Task complexity	The tasks for which the system has been used have been well-structured tasks requiring a well-defined set of pieces of information (i.e. slots to be filled) in order to provide the requested information. In Access which is the largest application so far incorporating the dialogue manager, about 25 pieces of basic information can be asked for/provided by the system. Access has an active vocabulary of about 5000 words. Because of the domain which requires that e.g. all car types are included in the system's vocabulary, it has been necessary to chunk the vocabulary since it far exceeds the 5000 available at that time. The task structure in the existing applications has a depth of 2-3 levels. In Access, e.g., there are two levels and in STORM there are three.
Communication	The domain communication is mixed initiative. The system will ask questions but the user may provide more information than asked for. In fact the user may provide all the parameters needed by the system in one utterance if s/he likes. The dialogue strategy is one of an ordered set of possibilities that serve to determine the optimal continuation of the dialogue. It is ordered in the sense that its setting corresponds to the minimal number of dialogue turns to be executed in order to solve the user's request, given that a user mentions all the relevant parameters for the request in the first utterance. The choice of the current strategy follows a degradation and recovery meta-strategy. The maximum strategy threshold can be set by the system developer, cf. Figure 4. The strategy is dynamic in that the system adjust its setting according to whether the system encounters a contradiction from the user or not. If the user contradicts the system it is assumed that something in its understanding went wrong. It thus tries to restrict the understanding task by asking more specifically, finally degrading to a leading initiative mode where the user is asked not only to provide an isolated task parameter but also to give it in some specific form that makes e.g. the recognition more reliable because a specifically restricted language model can be used. An example of the degradation strategy in use is shown in Figure 3. The system can do general contextual inferences as well as task dependent inferences. General inferences include e.g. time inferences and general contradiction detection. Task dependent inferences are tied to task objects. For example in the Access system the dialogue manager may ask the application about the power of a motor and get back a figure.

	This figure may mean horsepower or it may mean kilo watt. Which interpretation to use is decided by the belief module depending on the current context. Neither speech acts nor indirect speech acts are handled. System feedback is always used in order to make it as easy as possible for the user to detect misunderstandings. System-initiated meta-communication: The system is able to ask for repetition in various ways depending on what it thinks was the cause of lack of understanding. For instance it may ask the user to speak louder, simply to repeat, or it may tell that it did not understand the user. A possibility, not yet included in any application, is for the system to ask the user which of two alternatives s/he said, e.g. "Did you say Hamm or Hamburg?". In case of underspecified user information the system may initiate clarification. Generally this is done by presenting the most credible interpretation to the user and then leave it to the user to contradict if s/he disagrees. This is e.g. done in case of underspecified time input, such as four o'clock. If the user e.g. has indicated a time interval in which there are many train connections between the specified stations, the system may ask the user to be more specific. User-initiated meta-communication: The user may ask for repetions in several ways, e.g. by saying Pardon or Could you say that again. The system can reformulate itself in different ways. Other kinds of user-initiated meta-communication are not supported. For dialogue introduction and closure the system uses other forms of communication than domain and meta-communication.
Interaction level	The system's dialogue strategy has five levels, cf. Figure 4. Graceful degradation is used in case of misunderstandings.
Implementation of dialogue management	The approach taken regards dialogue as a joint activity. It is based on a layered set of units that, taken together, model the dialogue as a combination of belief and intention states of the system. The dialogue manager first tries to incorporate the surface semantic description of the user utterance into the contextual model. This may result in one of five types of change in the contextual model. Any sub-part of the utterance may either be new for the system, repeated by the user, inferred by the system, modified by the user or negated by the user. These changes lead to an appropriate change in the goal states of the pragmatic component (or dialogue module) as do changes coming from the application system interface (or task module) such as a request for a new parameter or the delivery of a solution to the caller's request. From these goal

	states the pragmatic component determines its next utterance trying to reach as many goals as possible without overloading the caller with information or different requests. No more than 3-4 items in an utterance are accepted. Dialogue goals may belong to one of three classes: initiative, reaction and evaluation. A request goal, e.g., is an initiative, a confirm goal is an evaluation, etc. The dialogue strategy determines how initiatives, reactions and evaluations are ranked and combined. In general, evaluations that are not initiatives are ranked higher than reactions, and reactions have precedence over initiatives. This ensures that answers to questions are realised earlier than system questions. The standard setting is that any number of reactions (except confirm goals for inferred items) and one initiative may be realised at the same time, making the dialogue fast. If something goes wrong a meta-strategy of degradation and recovery may move this setting, cf. Figure 4.
Linguistic structure	
Speech acts	The system does not identify speech (or dialogue) acts in the users' input.
Discourse particles	The system does not identify discourse particles in the users' input.
Co-reference	Intra-sentential co-references are resolved by the parser. Other kinds of co-reference resolution are not done but a mechanism for doing it is already incorporated. It is based on interaction between the belief module and the linguistic history maintained by the linguistic interface. For example the utterance "Does it stop in Ulm" has "vehicle" attached to "it" as surface semantics. Then a backward search in time is made until the first possibility which satisfies the conditions is met.
Ellipses	The system copes with ellipses via robust parsing allowing non-complete sentences.
Segmentation	The parser identifies sentences and phrases (parts) and represent an utterance as one or more independent structures (parts). This is the input to the dialogue management part.
Interaction history	
Linguistic	The linguistic interface module is responsible for maintaining a linguistic model of system and user utterances. Each SIL structure has an ID attached which is maintained by the linguistic history. Semantics are recorded and you may ask which surface realisations exist. Thus linguistic history is mainly surface structure representation. System utterances are used to trigger predictions. The record may be used for co-reference resolution, cf. above.

Topic	Semantic objects in the belief module have a kind of time stamp relative to each other. Semantic objects are represented in SIL. This information is not used for the moment but could be used in case a user mentions the same value for a slot more than once, e.g. Hamburg as the departure station, since it indicated that there may be some misunderstandings in the dialogue.
Task	The task module contains (represented in task SIL) all the most recent versions of data provided during the dialogue. This means that the dialogue can be finished at any time transferring all data to either the application system (if this was not done during the dialogue) or to the terminal masks of the human operator.
Performance	The system does not really maintain a record of the user's performance during interaction. However, if it detects a contradiction from the user, the system assumes that it has misunderstood something and tries to facilitate understanding by asking more specifically (degrading to a lower level). The dialogue manager has five levels of interaction, the lowest one being spelling mode. This has so far been sufficient. The strategy threshold can be set and you can never get above this threshold. The maximum number of allowed failures in a turn can be set. Usually three retries have been permitted. If the system still cannot understand the user it will then redirect to an operator if possible, or otherwise tell the user that it will stop the dialogue due to lack of understanding and ask for a re-try.
Domain model	
Data	The domain data used by the system during interaction depend on the precise application. In the German Sundial system, e.g. the data were train stations, connections, dates, departure and arrival times. The data have in all applications been fully realistic, e.g. a full train timetable. The representation of data varies, depending on the application. In the task module there will be a translation from Prolog to whatever the applications needs when data have to be retrieved.
Rules	The basic idea is that the available information should be exploited to the extent possible. For instance the train timetable database has a marker as to whether a train runs on a daily basis. If this is the case the system needs not negotiate a date with the user. Such inferences are carried out in interaction between the task module and the belief module. The task module also knows about system defaults and inferences. For inferences, see also communication above.
User model	

Goals	From the system's point of view, the user's goal during interaction is to carry out a task which the system knows about, such as to get train timetable information or to get an insurance offer.
Beliefs	Through feedback the system tries to make clear to the user what it has understood. Then it is up the user to contradict if his/her beliefs about the dialogue differ.
Preferences	In the current systems there are no user preferences handled. However, this facility could easily be added, but it needs caller identification.
User group	There are no distinctions among user groups, such as between domain novices and experts, novices and experts in using the system. However, it would be a possibility to add this using the level strategy.
Cognition	This may be considered part of graceful degradation.
Component architecture and function	
Generic architecture	The component has a domain-independent, generic architecture (easy to adapt to different task domains). It is language independent since it uses its own language (SIL) for input representation. The component has been implemented in Quintus Prolog and runs on a variety of different platforms (Unix, NT, Windows95). The syntactic and semantic descriptions resulting from the parsing process are passed to the dialogue level. Passing of multiple results between the linguistic level and the dialogue level is allowed. SIL (Semantic Interface Language) is used for interfacing between the linguistic analysis and the contextual interpretation via the linguistic interface module. SIL is also used for interfacing between the message planning module and the generation module. The dialogue manager also communicates predictions to the recognition module and to the linguistic analysis module.
Sub-components	The dialogue manager consists of five modules: The linguistic interface module interfaces the dialogue manager with the parser and is responsible for maintaining a linguistic model of system and user utterances. The dialogue module is responsible for maintaining a model of dialogue context, building an interpretation of user utterances and determine how the dialogue should continue. It receives as input from the belief module changes in the knowledge state. It then finds the best local continuation in terms of a system utterance, based on the actual dialogue situation. To determine the appropriate contextual interpretation of

	user utterances, the dialogue module interacts with the belief module which maintains a model of belief containing not only concepts created directly as a result of user utterances, but also inferential extensions. For example, if the system initiated an exchange to determine the departure date of a flight, this exchange can be closed if the belief model can interpret the user's utterance as referring to a date concept. The belief module requires context information from the dialogue module in order to guide the interpretation process. The belief module receives input from the linguistic analysis via the linguistic interface module. It anchors the received input into the semantic context. This process results in a change of the knowledge state and the instantiations of surface expressions pertaining to the application are mapped onto task objects by means of transfer rules. The belief module also co-operates with the message planning module in order to provide semantic descriptions of concepts referenced in the plan of system utterances. The task module is responsible for maintaining a model of the task structure of the dialogue, consulting domain-specific application databases and informing the dialogue module of the current state of the task. Typically, this involves deciding whether sufficient task information has been provided by the user and, if not, which additional parameters are required. In case of sufficient but incorrect information provided by the user, the system tries to relax the task parameters and propose alternatives. The task module receives from the belief module changes in the state of the task objects.
Flow	See the flow among the sub-components in Figure 1.
Function	The dialogue manager is the control module of an overall system.
System architecture	
Platform	-
Tools and methods	-
Generic	-
No. components	-
Flow	-
Processing times	-
System integration	
System resource utilisation	-
Shared information resources	-
Interactions	-
Data passing	-

2. System/component architecture

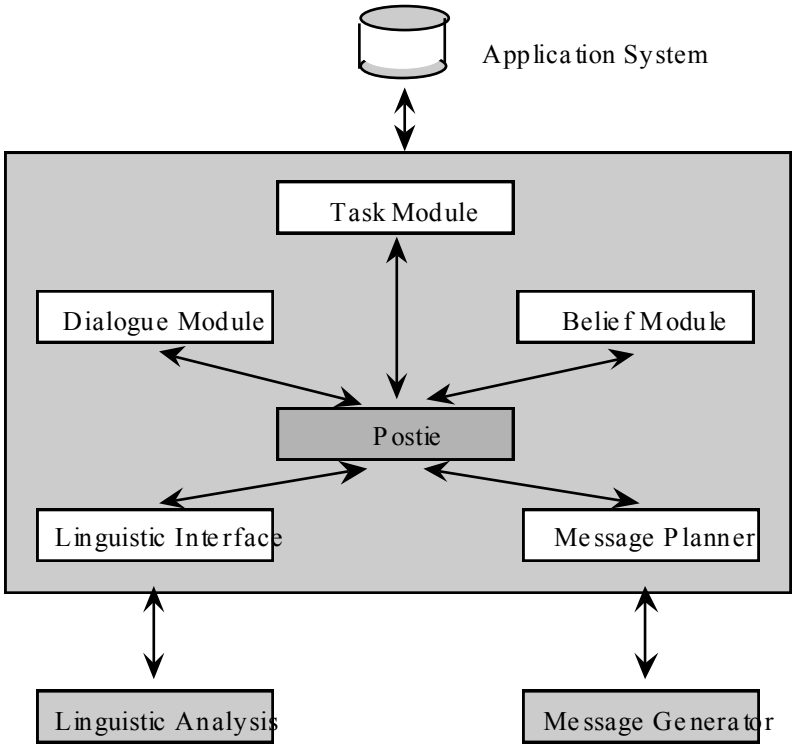


Figure 1. The architecture of the D-B dialogue manager.

3. System dialogue(s)

System1: This is the Automatic Intercity Inquiry System. Which information do you want?

Caller2: I need a connection from Essen to Hamburg tomorrow.

System3: You want to travel from Essen to Hamburg. You want to travel on Wednesday, September 22nd. At what time do you want to leave?

Caller4: At 16.00 hours.

System5: You want to leave around 16.00 hours?

Caller6: Yes.

System7: I have found two connections for you. The first train leaves Essen at 15.45, 15.45, arriving in Hamburg main station 19.32. The second train leaves Essen at 16.18, 16.18, arriving in Hamburg main station 19.43. Do you have another inquiry?

Caller8: No thanks [Hang-up].

Figure 2. Combined confirmation strategy in the German Sundial system [Heisterkamp 1993].

System1: This is the Automatic Intercity Inquiry System. Which information do you want?

Caller2: I want to go from Essen to Hamburg.

System3: You want to travel from Essen to Hamm? On what day do you want to travel?

Caller4: To Hamburg.

System5: You want to travel to Hanover?

Caller6: No, Hamburg.

System7: You want to travel to Hamm?

Caller8: No.

System9: Please spell the name of the arrival city.

Caller10: h-a-m-b-u-r-g.

System11: You want to travel to Hamburg?

Caller12: Yes.

System13: On what day do you want to travel?

Caller14: On Wednesday.

System15: You want to travel on Wednesday, September 22nd. At what time do you want to leave?

Caller16: Around four o'clock.

System17: You want to leave around 16.00 hours?

Caller18: Yes.

System19: I have found two connections for you. (...) Do you have another inquiry?

Caller20: No thanks [Hang-up].

Figure 3. Degradation and recovery in the German Sundial system [Heisterkamp 1993].

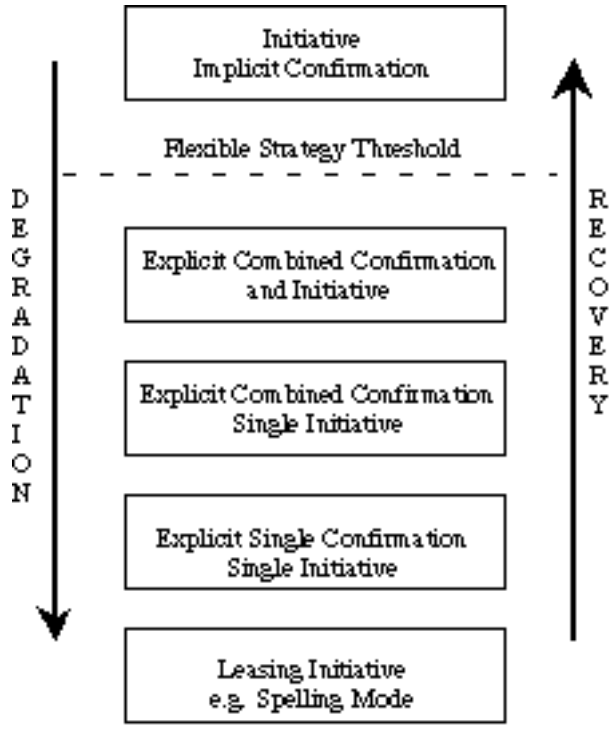


Figure 4. Dynamic Dialogue Strategy [Heisterkamp 1993].

7.2 Daimler-Benz Dialogue Manager Life Cycle

Laila Dybkjær and Niels Ole Bernsen

The Maersk Institute, Odense University
5230 Odense M, Denmark
laila@mip.ou.dk, nob@mip.ou.dk

1. Life cycle description

Overall design goal(s):

What is the general purpose(s) of the design process?

The original design goal was to build a dialogue manager for the Sundial research systems. This entailed the following features:

- Multilingual capability
- Different application areas
- Modularity (because of distributed development at several sites)

Later the design goal has turned into a goal of being able to use the dialogue manager not only in research prototypes but also in commercial products.

Hardware constraints:

Were there any a priori constraints on the hardware to be used in the design process?

No; Sun WS were eventually chosen because of availability at most sites in Sundial, but systems also ran on DEC Ultrix WS and HP. The first German parsers from Siemens ran on HP Explorer Lisp Machines.

Software constraints:

Were there any a priori constraints on the software to be used in the design process?

No; Quintus Prolog was chosen again on practical grounds.

Customer constraints:

Which constraints does the customer (if any) impose on the system/component? Note that customer constraints may overlap with some of the other constraints. In that case, they should only be inserted once, i.e. under one type of constraint.

The dialogue manager was originally developed for the Sundial research prototype systems. This means that there were no customers. In the Access project customers are insurance companies. Since such companies are not interested in revealing all their internal calculation rules on which they base their offers, the system developers could not have access to the entire task structure. This means that part of the task model and the inferences had to be made

external (in contrast to what has been the case earlier applications including the dialogue manager) to the dialogue manager. They are provided by the insurance database meaning that task goals may also be introduced from outside the task interpretation module.

In other applications, the constraints were mainly on the type of the task interface, in software, protocols, and hardware connections. Currently, there is an urge towards PC-based solutions.

Other constraints:

Were there any other constraints on the design process?

Cf. above.

Design ideas:

Did the designers have any particular design ideas which they would try to realise in the design process?

There were two basic ideas underlying the design of the dialogue manager. First, it had to be generic, i.e. had to be able to work in more than one language and across several task domains. Second, it had to be co-operative during interaction with the user.

Designer preferences:

Did the designers impose any constraints on the design which were not dictated from elsewhere?

Rapid prototyping was necessary in order to meet the need for intermediate demos.

Design process type:

What is the nature of the design process?

Originally the design process type was exploratory research. Later design stages were adaptation and optimisation.

Development process type:

How was the system/component developed?

In the Sundial project the dialogue model was developed through Wizard of Oz.

Requirements and design specification documentation:

Is one or both of these specifications documented?

There was a first-year deliverable in Sundial called 'Detailed design specification'.

Development process representation:

Has the development process itself been explicitly represented in some way? How?

No.

Realism criteria:

Will the system/component meet real user needs, will it meet them better, in some sense to be explained (cheaper, more efficiently, faster, other), than known alternatives, is the system/component "just" meant for exploring specific possibilities (explain), other (explain)?

The goal was to

- explore the possibilities of what today would be called 'conversational' dialogue;
- build a prototype (especially for Vocalis) to demonstrate technological leadership in speech dialogue.

Typically, the applications in which the dialogue manager has been used , have been meant to make the performance of a task faster and more efficient.

In the further development and the porting to different applications, central criteria were performance criteria and financial adequacy criteria.

Functionality criteria:

Which functionalities should the system/component have (this entry expands the overall design goals)?

The dialogue manager is a module the main functionalities of which are

- to assign an interpretation to the input it receives from the linguistic analysis module,
- to negotiate the effects of this interpretation with one or more given application systems,
- to decide, on both the interpretation and the application information, how the dialogue may best continue, and,
- if it is the system's turn to speak, plan an adequate system utterance.

Usability criteria:

What are the aims in terms of usability?

The component should be language independent, it should be possible to use the component in a variety of applications with a minimum of modifications, and the component should be able to run in different environments.

Organisational aspects:

Will the system/component have to fit into some organisation or other, how?

The dialogue manager must be usable for DASA and fit their programs.

Customer(s):

Who is the customer for the system/component (if any)?

The component started as part of a research system (Sundial) for which there was no real customer but for which a potential customer could have been a travel agency. In fact British Airways was involved. For the Access system companies with a call centre, e.g. insurance companies, are the (potential) customers. For the dialogue manager as such DASA is the customer.

Users:

Who are the intended users of the system/component?

The original dialogue manager was developed for four different languages: German, French, Italian and English. D-B's later use of the dialogue manager has been for German systems. All systems in which the dialogue manager has been used, are walk-up-and-use systems which should be self-explicatory and thus requiring no particular background from their users.

Developers:

How many people took significant part in the development? Did that cause any significant problems, such as time delays, loss of information, other (explain)? Characterise each person who took part in terms of novice/intermediate/expert wrt. developing the system/component in question and in terms of relevant background (e.g., novice phonetician, skilled human factors specialist, intermediate electrical engineer).

Between 5 and 12 people at different sites participated in the development of the Sundial dialogue manager. These were people with many different backgrounds, such as electrical engineer, philosophy, cognitive science, computer scientist and linguist. After Sundial finished three people at D-B have currently been involved in working on the dialogue manager, though not full time. Two of these people have a background in computer science and linguistics and the third one in German philology.

Development time:

When was the system developed? What was the actual development time for the system/component (estimated in person/months)? Was that more or less than planned? Why?

The first dialogue manager was developed between 1989 and 1993. Between 40 and 50 person years were spent on the dialogue manager. This was also what had been planned. Since then there has currently been ongoing work on improving the dialogue manager and using it in other applications. About 4-5 person years have been spent on the dialogue manager after 1993. It typically takes 4-6 person months spent on dialogue to adapt the dialogue manager to a new application (and 3 person months for the grammar).

Requirements and design specification evaluation:

Were the requirements and/or design specifications themselves subjected to evaluation in some way, prior to system/ component implementation? If so, how?

No. The closest to this would be the reviewers' comments on the plans in the Sundial project. However, these are not publicly available. In later projects there has been no requirements and design specification evaluation for the dialogue manager, but there has been for the application as such. Of course, the dialogue manager has to be able to manage the appropriate dialogue structures.

Evaluation criteria:

Which quantitative and qualitative performance measures should the system/component satisfy?

Such criteria did not exist when the Sundial project started. They were invented when there was a need which means when the system/component was going to be evaluated. In the Access project there are very few evaluation metrics and they are basically all related to costs. The Call Centre manager (in the ACCeSS-1 Insurance application) has a figure of Cost-Per-Contract, i.e. the expenses needed to get one customer to sign a contract. This figure can be measured for both the 'natural' and the automatic version of the dialogue. This figure captures all aspects of the dialogue, in that an insufficient system will only get considerably less contracts, i.e. customer acceptance. So, at the end of the day, you compare the Cost-Per-Contract, and this comparison also tells you exactly what the dialogue system may cost, i. e. its value on the market.

Evaluation:

At which stages during design and development was the system/component subjected to testing/evaluation? How? Describe the results.

WOZ simulation was used to begin with to extract a user model.

In Ulm data collection was done in logfiles. The dialogues were transcribed but not annotated. Thus what was available was the transcribed dialogues, the speech data and for the user utterances the recogniser results, the parser results and the predictions used during processing. There was no automatic information extraction from the data. The dialogues were looked through and transaction successes counted. In case a problem was detected, it was tried to identify the reason and find a way in which to repair the problem. Problem detection was not done systematically but rather on the fly by browsing the transcribed dialogues.

Test suites were used. A set of over 100 different types of test dialogues were used. They are available as annexes to the Sundial final report. These tests used typed input. The test files define a subset of typical dialogue situations and incorporate the knowledge of conversational rules, contextual interpretation of utterances, dialogue strategies and database results. Other tests were performed but cannot be replicated since spoken input was used.

The overall performance of the implemented German Sundial system was tested with semi-naive users who were familiar with computers but not knowing details about the dialogue system. Microphone input was used. The person who supervised the tests instructed the subjects. Each subject received four intercity train timetable scenarios. Two of them were pre-defined, the other scenarios depended on the subjects' personal choice. Before starting their inquiries they had to define them by stating the places of departure and arrival and the departure time in a protocol. Each inquiry was tried until the system provided the required connection or spotted that no answer existed. Having carried out the dialogues the users had to fill in a questionnaire describing their attitudes towards the system. The experimenter kept a hand-written protocol during the sessions. This protocol would e.g. tell what subjects actually said whereas the logfile only contained what was recognised. The hand-written notes were later inserted in the logfile.

Measure: Dialogue completion rate	**No.**	**%**
Failed dialogues	94	36.8
Successful dialogues	79	31.0
Spotting "difficulties"	82	32.2
Total successful	161	63.1
Measure: Contextual appropriateness of system utterance	**No.**	**%**
AP appropriate	635	98
IA inappropriate	11	2
AI not agreed AP/IA	0	0
TF total failure (ruled out)		
IC incomprehensible	0	0
Measure: Transaction success of the dialogue	**No.**	**%**
S successful	5	6

SC relaxed constraints	63	80
SN announcing no solution	11	14
F failure	0	0
Dialogue length	**User turns**	
Failed dialogues	3.6	
Successful dialogues	8.0	
Spotting "difficulties"	6.1	

Table 1. Evaluation results from the Sundial system [Eckert et al. 1993].

A total of 255 dialogues were recorded. Evaluation focused on the 79 successful dialogues, cf. Table 1.

Evaluation metric applied were contextual appropriateness, turn correction rate, transaction success, dialogue completion rate and average length of a dialogue [Eckert et al. 1993]. In Erlangen tagging of the user and system turns is performed by human experts who classify the appropriateness of utterances and the success of dialogues and also identify correction turns. Tags are counted automatically.

Nothing is really stated about comparability of the test (result)s with those of other components of similar scope.

Blackbox and glassbox evaluation of the Sundial system was done [Simpson and Fraser 1993].

	P2	P3	P3+	P4	P5	P6	P7	P8
collection date, started	9306	9311	9312	9401	9404	9408	9409	9501
corpus size (MB)	127	42	53	133	50	28	204	on-go-ing
number of dialogues	237	49	77	161	42	35	325	
number of utterances	1742	585	533	1365	199	303	2187	
number of words	6384	1841	1668	4238	1144	1154	6773	
number of different words	239	191	168	320	196	174	1056	
number of unknown words	68	25	21	111	77	-	-	
total duration in sec	3983	1318	1677	4152	1590	902	6324	
avg length of dialogue in sec	183	220	204	183	-	188	179	
avg length of utterance in sec	2.4	2.3	3.1	3.0	2.8	3.0	2.89	

avg words per utterance	3.67	3.16	3.13	3.11	5.75	3.80	3.10	
microphone/telephone	mic	PABX		PSTN				
supervised/ unsupervised	supervised		self-sup.	unsupervised				
user skills	semi-naive		ex-pert	naive				

Table 2. Evaluation data from [Eckert et al. 1995].

Data on a further development of the Sundial system have been collected through a series of iterations, cf. Table 2 [Eckert et al. 1995]. Phase 1 used read speech for recogniser evaluation and is not included in the table. In phase 2 a high quality microphone was used and a first corpus of in-house inquiries was collected. Phase 3 switched to telephone quality and a more sophisticated parser. In phase 3+ substantial improvements were made to the system. In phase 4 the first corpus of spontaneous speech was collected over public telephone line. In phase 5 a different dialogue model was used. In phase 6 the language models of the recogniser were enhanced. In phase 7 a new release of the parser and of the dialogue manager was incorporated.

Mastery of the development and evaluation process:

Of which parts of the process did the team have sufficient mastery in advance? Of which parts didn't it have such mastery?

In the Sundial project there was no mastery of the development and evaluation process in advance since everything was new. In later projects the team has had good mastery of the process.

Problems during development and evaluation:

Were there any major problems during development and evaluation? Describe these.

In Sundial there were competing implementations of some of the modules of the dialogue manager. For example there were two task modules and two dialogue modules. This caused problems as to which module to use. Also it was not unproblematic that so many sites were involved. There was a tendency to develop individual methodologies at the individual sites and not always conform to the same standards.

Development and evaluation process sketch:

Please summarise in a couple of pages key points of development and evaluation of the system/component. To be done by the developers.

Requirements were specified, an architecture was outlined, these were distributed among the sites in the Sundial project. The flow (messages) between the individual modules was specified, a message file was specified to enable each module to interact with dummies. Every three months there would be an integration week. This meant that representatives from the involved sites would meet for a couple of days to work together and i.e. address interface problems between the modules. After each integration meeting there would be one person responsible for distributing the new official versions of software. The evaluation was mainly done in terms

of comparing the current version to the requirements to see what worked and what did not work yet.

Component selection/design:

Describe the system components and their origins.

The first dialogue manager version was built in the Sundial project. The sites involved were Vocalis, Erlangen University, Daimler-Benz, Surrey University, CSELT, CNET, Cap Gemini and IRISA. The dialogue manager was later further developed in-house by D-B.

Robustness:

How robust is the system/component? How has this been measured? What has been done to ensure robustness?

Graceful degradation is used to ensure a robust and co-operative dialogue interaction. In the beginning this method caused the system to break down quite often. Now, however, it is very stable and if it turns out that the system cannot help the user, i.e. even the lowest level does not help the system to understand the user correctly, the user is told that this is the case and/or redirected to a human operator.

Maintenance:

How easy is the system to maintain, cost estimates, etc.

Maintenance is not needed very much. There are no guidelines on how to do it. For modifications, customisation and additions see below.

Portability:

How easily can the system/component be ported?

If you can run Quintus Prolog on the machine it is very easy to port the dialogue manager to this machine. It runs on Windows, NT, and Unix platforms. A version for Sixtus Prolog under Linux is being considered.

Modifications:

What is required if the system is to be modified?

The dialogue management model has been extended for multimodal applications where a speech interface is integrated into a direct manipulation environment [McGlashan 1996]. Interpretation of graphical input is based on the same semantic and pragmatic structures required for spoken language, although the structures are less complex to process due to the absence of underspecified input like anaphora and ellipsis. Moreover, the algorithm for realising system dialogue acts has been modified to allow generation of graphical output, and the semantic and interpretative functions have been enhanced to handle 'command and control' utterances. However, the basic principles of the dialogue management model used in speech-only applications remain intact.

Additions, customisation:

To adapt the system to a new information service, modelling the tasks, the application system, the discourse world and the language coverage is sufficient. Language and task can be changed in the dialogue manager simply by resetting switches which govern the static knowledge bases

consulted during dialogue management.

Property rights:

Describe the property rights situation for the system/component.

Daimler-Benz has the property rights of the dialogue manager.

2. Documentation of the design process

E.g. specification documents or parts thereof.

Nothing available.

3. References to additional project/system/component documentation

Bayer, Thomas; Heisterkamp, Paul; Mecklenburg, Klaus; Renz, Ingrid; Regel-Brietzmann, Peter; Kaltenmeier, Alfred; Ehrlich, Ute (1995): Natürliche Sprache - ein multimedialer Träger von Information. InfoPort - ein Projekt zur Überbrückung von Medienbrüchen. In: Proceedings of DAGM-95, Bielefeld.

Brietzmann, Astrid; Class, Fritz; Ehrlich, Ute; Heisterkamp, Paul; Kaltenmeier, Alfred; Mecklenburg, Klaus; Regel-Brietzmann, Peter; Hanrieder, Gerhard; Hiltl, Waltraud (1994): Robust speech understanding. In: Proceedings of ICSLP '94, Yokohama.

Eckert, W., Kuhn, T., Niemann, H. Rieck, S., Scheuer, A. and Schukat-Talamazzini, E.-G.: A spoken dialogue system for German intercity train timetable inquiries. Proceedings of Eurospeech'93, 1993, 1871-1874.

Eckert, W., Nöth, E., Niemann, H. and Schukat-Talamazzini, E.-G.: Real users behave weird - Experiences made collecting large human-machine-dialog corpora. Proceedings of ESCA Workshop on Spoken Dialogue Systems, Vigsø, Denmark, 1995.

Hanrieder, Gerhard; Heisterkamp, Paul (1994): Robust analysis and interpretation in speech dialogue. In: Niemann, Heinrich; de Mori, Renato; Hanrieder, Gerhard (Eds.): Progress and prospects of speech research and technology. Proceedings of the CRIM/Forwiss Workshop, 5.-7. September 1994, Munich, Germany. St. Augustin: Infix. (Proceedings in Artificial Intelligence. 1.).

Heisterkamp, Paul; McGlashan, Scott; Youd, Nick (1992): Dialogue semantics for an oral dialogue system. In: Proceedings of ICSLP-92, Banff, Alberta, Canada, 1992.

Heisterkamp, Paul (1993): Ambiguity and uncertainty in spoken dialogue. In: Proceedings of Eurospeech '93, Berlin.

Heisterkamp, Paul (1996): Natural language analysis and generation. Materials of the course held at the 4th European Summer School on Language and Speech Communication - Dialogue Systems. Budapest, Hungary.

Heisterkamp, Paul; McGlashan, Scott (1996): Units of dialogue management: an example. In: Proceedings of ICSLP-96, Philadelphia, Pa.

McGlashan, Scott; Fraser, Norman M.; Gilbert, G. Nigel; Bilange, Éric; Heisterkamp, Paul; Youd, Nick (1992): Dialogue management for telephone information systems. In: Proceedings of the 3rd Conference on Applied Natural Language Processing, Trento, Italy.

Mecklenburg, Klaus; Hanrieder, Gerhard; Heisterkamp, Paul (1995): A Robust parser for continuous spoken language using PROLOG. In: Proceedings of Natural Language

Understanding and Logic Programming 1995, Lisbon, Portugal.

Regel-Brietzmann, Peter et al.(forthcoming): ACCeSS - Automated Call CEnter through Speech understanding System. A description of an advanced application. Proceedings of Eurospeech '97, Rhodes.

Simpson, A. and Fraser, N.: Blacbox and glassbox evaluation of the Sundial project. Proceedings of Eurospeech'93, 1993, 1423-1426.

7.3 Danish Dialogue System Dialogue Manager Grid

Arne Jönsson, Patrik Elmberg and Nils Dahlbäck

Department of Computer and Information Science
Linköping University, S-581 83 Linköping, Sweden
arnjo@ida.liu.se, x97patel@ida.liu.se, nilda@ida.liu.se

1. Dialogue Component Tasks

What are the main tasks of the Dialogue Component?

Interpret semantic objects and decide on the next action to take: database request or user output.

Send predictions to speech recogniser and parser on which sub-grammar and which sub-lexicon to use.

Task management and turn-taking involving basic task operations and meta-communication.

Does the dialogue component only have a mediating function, as is usually the case with translation systems in which the system does not really act as a dialogue participant?

The DM is the central part of the system. Controls the dialogue and usually communicates with all other modules.

To which system requirements do the tasks of the dialogue component contribute (robustness, efficiency, etc.), and what does this contribution consist of?

Robustness as the DM handles meta-communication. Also efficiency as it decides on the sub-grammar to use which ought to speed up recognition and parsing. From the 500 word lexicon at most 100 words based on the current task is selected.

2. Dialogue Component Architecture

a. *Architecture specification*

What are the submodules of the dialogue management component of the system, what are their functions, and how do they interact (see also below)?

Provide, if possible, a figure of the architecture of the dialogue management component, thereby indicating the interactions between the submodules.

The central submodule is the item handler which receives input from a user input module, decides on the task to perform, communicates with a domain handler and eventually provides a response to the user output module. Other submodules are a dialogue history module and a user model. All modules communicate via the item handler.

The dialogue structure module describes the default order of tasks.

b.*Generic architecture*

Does the Dialogue Management component of the system have a generic architecture? Give an answer to this question insofar it is relevant to the grid analysis, thereby also indicating the level of genericity (language-independent architecture, domain-independent architecture, etc.).

The architecture is domain/task dependent, i.e. the task handler can only work with this specific domain/dialogue structure. Changing domain is not easy.

c. *Interaction with other system components*

Specify the input/output characteristics of the dialogue component.

Messages are sent to other system modules via the "Interpretation and Control Module".

3.1 The Implemented Approach

a. *Kind of descriptive approach*

Which of the following dialogue management approach(es) (some of which are partially overlapping) have been implemented in the dialogue system, this according to the descriptions given below?

The task structure determines the behaviour of the dialogue manager and the system. Hence the approach is topical. It is collaborative in the sense that the system takes into account the user's task. The dialogue is modelled in graphs used to describe the dialogue model when developing the system.

b. *Use of single versus multiple approach*

Does the system comprise a multiple approach?

If the system uses a multiple approach, what are the reasons for assuming this multiple approach, what are the functions of the single approaches and what is their prominence (equally prominent, one subservient to the other, etc.)?

Only the task structure and dialogue structure and they use the same approach.

c. *Implementation characteristics*

Which characteristic approach-dependent options have been realised in the implementation?

A graphical representation language has been developed and the task structure is described in this language. See above on grammar.

d. *Use of hybrid identification and prediction methods*

Does the system comprise a hybrid approach, also making use of additional statistical methods?

Give a specification of the set of dialogue functions for which statistical methods are used (see also below), making explicit whether the method applies to all structural levels in the dialogue or just to a specific kind?

The Dialogue manager does not use statistical methods.

3.2 Dialogue Structure

A. Intentional Structure

Given the task-oriented character of the dialogues under consideration, we take the intentional

structure of a dialogue as the (mostly hierarchical) structure of dialogue segment purposes to perform a (sub)task.

a. *Task characteristics*

(i) *Task specification*

Which specific tasks are performed by the system (making a reservation, showing a time table, fixing a meeting date, etc.)?

(ii) *Task structure*

Are all tasks well-structured, i.e. do all tasks have a stereotypical structure explicating both the amount and order of information updates needed to complete the task (see, in particular, Bernsen et al. 1998 on this point)?

Danish domestic flight ticket reservation; well-structured task. Task items model the tasks and each task in turn consist of sub-tasks.

A distinction is made between well-structured, i.e. stereotypical tasks, and ill-structured. Ill-structured tasks are considered difficult to predict and are not handled by the system.

b. *Communication types*

(i) *Domain communication*

Specify the relevant characteristics of the task-oriented domain communication, in particular whether and/or how the system accounts for the following phenomena (some of which in fact may give rise to meta-communication):

System directed, user directed or mixed initiative communication

Free or bound order of main tasks

Discontinuous user input ('input gaps')

Contextual inferences of different types

Indirect speech acts

Incomplete, underspecified user requests

Incomplete answers to system questions

Other

System-directed task communication.

Bound task order. Discontinuous input not handled. If a user provides more information than requested the recogniser/parser does not forward this to the dialogue manager and hence the dialogue manager need not be able to cope with such problems.

Mixed initiative meta-communication; users may initiate meta-communication through keywords.

System-directed other communication, such as the opening and closing of a dialogue. If the user wants to close the session he/she has to either hang up or wait until the system asks if more information is needed.

Some questions are yes/no or multiple choice, most are general and focused.

(ii) *Meta-communication*

Does the system allow for both system- and user-initiated meta-communication, and in what way?

How does the system behave with respect to different types of meta-

communication, such as:

Repairs (repair types, repair success, etc.)

Clarifications (location-restricted/-unrestricted, system-/user-initiated, etc.)

Repetitions

Reasons ('nucleus reasons' vs. 'satellite reasons')

Deliberations (deliberation types)

Feedback (feedback realisation types)

Other

The system does not distinguish between different levels of communication.

(iii) *Other communication types*

Dialogue introduction and dialogue closure phases (domain-independent communication)

Side structure information induced by digressions (related to, but out of the domain communication)

The system distinguishes different types of errors: silence, input which does not make sense, domain error input, the system is in error.

Two types of introductory messages, one for novices and one for experts.

Closure request from the system. The user can not close the session by taking the initiative unless s/he hangs up.

c. *Modelling of intentional structure*

How is the dialogue's intentional structure modelled in terms of the chosen dialogue management approach?

Although task-oriented dialogues have an underlying goal structure, these dialogues are not always modelled in terms of the goals to perform these tasks (see above).

(i) *Uniform versus partial application*

In case of a multiple approach, are all the single approaches uniformly applied to both the higher and lower intentional levels in the dialogue, or are they applied to just a specific part of this structure?

The intentional structure is modelled in the task structure and no explicit representation of a user's intentions is made.

(ii) *Intentional levels*

How many intentional levels are distinguished?

a. *Intentional continuum*

Is every underlying discourse segment purpose represented separately in terms of the chosen approach?

As for a multiple approach, how do the containing approaches differ with respect to the phenomenon of an intentional continuum?

Does the system define a hierarchical order for the fine-graded intentional levels it distinguishes?

Note that the phenomenon of an intentional continuum characteristic for human-human dialogues is usually not fully accounted for in dialogue systems.

b. *Representation of larger intentional units*

If only larger intentional segments are distinguished by the system, which are these and are they hierarchically structured?

Relevant questions wrt. the assumption of larger intentional units are the following:

How is the subdivision in intentional units motivated?

What motivation is given for the assumed basic dialogue units, specific classes of which constitute the higher-order levels?

How are basic dialogue units determined?

What is said about the relation between dialogue phases and the classes of basic dialogue units constituting these phases?

Which typology is defined for basic dialogue units? Indicate whether the set of basic dialogue units is an open set, whether the proposed typology contains domain-dependent basic units, etc.

The basic dialogue units are turns and sub-tasks.

d. *Dialogue parsing*

Give a description of the dialogue parsing process, accounting for at least the following points:

What type of dialogue parser has been chosen (a task dependent parser, a topic-dependent parser, etc.)?

What kind of parsing strategy has been followed (top-down or bottom-up strategy, complete or partial parsing strategies, etc.)?

Give a brief description of the parsing process, including a specification of parsing constraints (syntactic constraints, semantic constraints, prosodic constraints, contextual constraints, etc.).

How does the dialogue parser account for the isomorphism between intentional and linguistic structure? Does parsing automatically give rise to a representation of the linguistic structure?

How does the dialogue parser account for updates of the corresponding attentional states, either directly or indirectly?

No parser is used.

B. Attentional state

a. *Modelling of attentional state*

(i) *Local focus*

How does the system model local focus?

In what specific way does the modelling of local focus give rise to expectations for the next user input?

How do focus values selected by answers to local questions give rise to updates on the global level (see also above)?

How is local focus determined in case of other speech acts than questions, and how is it determined in case of extended answers to questions?

Of what types of linguistic data is made use to identify local focus (specific prosodic information, information about contextually determined word order, etc.)?

Current sub-task, for instance to determine the number of travellers, plus meta-communication tasks. The user can only modify the previous sub-task, recursively. Thus, if a user wants to modify a sub-task discussed before the previous task, he/she must modify all sub-tasks between the current one and the one to be corrected. This is because all sub-tasks are validated before they are accepted, and thus, a change of sub-task value can affect all other succeeding sub-tasks.

Predictions sent to recognizer and parser; task dependent parsing. Predictions are static, i.e. fixed at run-time. This works for a system-initiated dialogue but not for mixed-initiative where a user can change sub-task.

(ii) *Global focus*

Does the system model global focus? If so, how does it model global focus in relation to local focus?

If the system makes use of expectations induced by global focus, what is their nature and impact?

Does the system make use of linguistic markers of global focus phenomena, e.g. topic shift markers which are relatively easy to implement?

No global focus.

b. *Additional expectations*

What other (statistical or other) methods are implemented to compute, in particular, the immediate local expectations which arise from a current system/user question?

What is the function of a multiple approach in this respect?

Only one method.

c. *Co-operativity*

Task realization in dialogue is considered to be part of its dynamic, attentional structure. How does the system account for a co-operative performance of these tasks, e.g. with respect to the following phenomena?

Under-informativeness (e.g. the system's exhaustiveness strategies)

Over-informativeness (e.g. the system's refinement strategies in case that the requested task is too broad)

Truthfulness of provided information (the system's information checking mechanisms)

Brief and orderly presentation of requested information (the system's presentation mechanisms)

If possible, indicate whether and how co-operative action initiation as part of these strategies is goal determined.

Under-informativeness is handled through limited clarification sub-dialogues.

C. Linguistic Structure

The linguistic structure of a dialogue constitutes its segmentation structure. The dialogue's linguistic structure is in correspondence with both the intentional structure of the dialogue and

the structure of attentional states.

a. *Dialogue segments*

How many segmentation levels are distinguished?

Because of the correspondence between intentional structure and segmentation structure, dialogue segmentation within the system crucially depends on how many intentional levels are distinguished (see above).

The dialogue is segmented into sub-tasks and turns.

b. *Speech act types*

Which speech act types (e.g. questions, assertions, commands, etc.) can be handled by the system?

How are the different types identified?

How successful is the system with respect to the interpretation of indirect speech acts (see also above), and how does this depend on the dialogue management approach chosen?

Primitive distinction between commands (meta-communication) and statements (answers) in user input; use of commands (questions), and statements for providing feedback, error messages and other information in output. Two selected user keywords; "repeat", and "correct", and two system keywords; "not understood" and two versions of "time-out".

Speech acts are not explicitly identified. System acts are questions, statements and reactions to meta-communication. User acts are answers and the two meta-communication keywords.

No indirect speech acts are handled.

c. *(Co-)reference*

How does the system behave with respect to different linguistic realisations of the same referential object (pronouns, anaphoric descriptions, etc.)?

Does the system keep a history of co-referential expressions?

How does co-reference relate to the dialogue's intentional and attentional structure?

No anaphora resolution; Ellipsis is handled in a primitive way. No well formed sentences are provided from the parser. Pronouns are not treated.

3.3 Dialogue Context

A. Dialogue History

a. *Representation of Intentional Structure*

Does the system incrementally construct a representation of the intentional structure as the result of the preceding context?

What is the format of representation of the intentional structure (tree representation, matrix representation, etc.)?

Which submodule of the dialogue component is responsible for this representation (e.g. the dialogue planner) and where is the representation stored (e.g. in the dialogue memory)?

Which other system component does the stored intentional information contribute to and how?

A task record is maintained.

b. *Representation of Attentional States*

Does the system provide an incremental representation structure of the set of attentional states constituting the dynamic development of the dialogue, thereby providing a representation of the set of open alternatives ('focus set') at any dialogue information state?

What is the format of representation (e.g. an updated SIL object or a list of such objects in Sundial and the Daimler-Benz dialogue system)?

How does the incremental representation structure of attentional states account for the set of *referentially accessible objects* at any dialogue state?

Which other system component does the representation of attentional structure contribute to and how?

Semantic contents of the previous system and user utterances. The system semantics is used to interpret yes/no-responses to two-valued items. The user semantics is used in the synthesis of error messages.

Only previous turn is recorded.

The names of the previous tasks and the names of the previous task graphs. They are stored in a stack which is used for the meta-communication command "correct".

c. *Representation of Linguistic Structure*

Does the system represent the segmentation structure of the preceding dialogue, and what is the relation with the incremental representation of the corresponding intentional structure?

Does the system also represent referential structure (see above)?

Which other system component does the represented linguistic structure contribute to and how?

Only a representation of semantics, not of surface language.

B. User Model

a. *Modelling of performance-relevant aspects of the user*

To what extent does the system model performance-relevant aspects of the user, such as user beliefs (his own beliefs as well as beliefs about the system), user desires (preferences), user expertise, etc.? What is precisely their function within the system?

Does the system also allow for inferencing over initial beliefs?

What is the status of, in particular, the user beliefs, desires and intentions? And, what is most important, how do these interact in the process of dialogue control?

The only goal is assumed to be flight ticket reservation. However, the system can inform on departure time if the user underspecifies this, e.g. responding "afternoon".

No inferencing on user beliefs.

The sub-dialogues are hard-coded. Preferences are determined at run-time; the scope is the current reservation task.

System novice/expert distinction; the system's introduction and discount information is optional. Novices and experts are distinguished based on an initial request if the user is

familiar with the system.

b. *Contribution to co-operativity*

What contribution is provided by the user model with respect to co-operative system behaviour?

Give an indication of the remaining, most relevant problems in this respect.

It provides more initial information to a novice user and also provides information on discounts. This is not provided for experts.

C. Domain Model

a. *Data*

Give a specification of the data types determined by the task(s).

Provide the global characteristics of objects, properties and relations relevant to these tasks.

Timetable, fares, flights, customers, reservations. There is no clear cut between the domain model and the database in the sense that part of the inferential rules are in the database and another part in the dialogue module. This is pointed out as a problem when extending and maintaining the program. All data are in the database.

b. *Rules*

Give a global characterisation of world knowledge rules used by the system, as well as the characteristic properties of their application.

The dialogue handler must be able to set item values, retrieve item values, and delete item values in the domain representation.

D. Dialogue Settings

a. *Physical setting*

What are the characteristic physical circumstances in which the dialogues take place (face-to-face situation, communication through electronic devices, etc.), and how is their influence on co-operative communication modelled in the system?

b. *Social setting*

In what type of communicative situation do the dialogues take place (expert-novice, employer-employee, etc.), what roles do user and system have in these situations, and how is social knowledge of this type modelled in the system?

The system is intended for walk-up-and-use and was tested with external users via the public telephone line, although with a simulated recogniser.

7.4 Danish Dialogue System Dialogue Manager Life Cycle

Nils Dahlbäck, Patrik Elmberg and Arne Jönsson

Department of Computer and Information Science
Linköping University, S-581 83 Linköping, Sweden
nilda@ida.liu.se, x97patel@ida.liu.se, arnjo@ida.liu.se

1. Life Cycle Questions, General

Overall design goal(s):

What is the general purpose(s) of the design process?

To build realistic prototypes of advanced interactive speech systems, focusing on the integration of speech technology, natural language processing, knowledge representation and human-machine interaction. To explore speech technology. To create and consolidate know-how.

Other goals were; real-time performance, robustness and flexibility, speaker independent continuous input. Accept natural forms of language and dialogue in a walk up and use-system.

Hardware constraints:

Were there any a priori constraints on the hardware to be used in the design process?

PC based, see customer constraints below. A telephone line for the speech input.

Software constraints:

Were there any a priori constraints on the software to be used in the design process?

Use of the Dialogue Description Language.

Customer constraints:

Which constraints does the customer (if any) impose on the system/component?

There was no real customer. The choice of a PC-based system was based on the assumption that this was what prospective customers, (read Danish travel agencies), could afford. Unix workstations were much more expensive than a PC.

Other constraints:

Were there any other constraints on the design process?

As a telephone was used, the developers assumed a real-time performance necessary. This together with the weak performance of the speech recogniser put additional constraints on the vocabulary. A lexicon with no more than 500 words of which maximum 100 active in the memory. This was known to be insufficient for a commercial system, but was assumed not to prevent the project from arriving at interesting research results.

The limited capability of current speech and NLP technology is mentioned as one constraint, though since this is true of all projects, it is not clear that it should be included here.

Design ideas:

Did the designers have any particular design ideas which they would try to realize in the design process?

No specific theories of dialogue was to be tested.

Designer preferences:

Did the designers impose any constraints on the design which were not dictated from elsewhere?

Use of the Dialogue Description Language.

Design process type:

What is the nature of the design process?

Research prototype development.

Development process type:

How was the system/component developed?

General software engineering life-cycle model, adapted to advanced interactive speech systems. WOZ simulations to capture the user behaviour.

Another adaptation of the general model is that it is more loosely specified than would be required for a commercial project.

Requirements and design specification documentation:

Is one or both of these specifications documented?

A requirement specification was developed, including issues of strategic goals and system goals and constraints like overall performance, kind of speech input accepted, type of domain model.

Development process representation:

Has the development process itself been explicitly represented in some way?

Used the DSD/DR (Design Space Development and Design Rationale) at the end for the blackbox testing. This method was developed within another project simultaneously at the same site.

The experience from using this was positive, but some minor problems are mentioned, especially that training is required for inserting information in the right place; the frames tend to become very long and complex as the project progresses; it is time-consuming to use. But the benefits outweigh these aspects.

It is an open question though, to which extent the method can be used by others than its originators.

Realism criteria:

Will the system/component meet real user needs, will it meet them better, in some sense to be

explained, than known alternatives, is the system/component "just" meant for exploring possibilities, or what?

The system was a research project how to get a working prototype of a telephone based speech system. No user studies preceded the decision to build a system of this kind. However, contacts were established to a travel agency in the early design phase, i.a. with the intention to get input on realism.

The general objectives (casual users, walk up and use, without any pre-training, etc.), seem realistic, or are at least shared by the research community today. But no analysis of expected changes or development in hardware or network capabilities and how these may influence user and customer needs and preferences are presented.

Functionality criteria:

Which functionalities should the system/component have?

The system should enable the user to make reservations for Danish domestic flights. Language should be Danish, with a small vocabulary (500 words, only 100 at a time), short user utterances (3-4 on the average, not longer than 10 words), and allowing for natural forms of dialogue.

Usability criteria:

What are the aims in terms of usability?

A walk up and use system. The choice of vocabulary should as far as possible be chosen not to affect the naturalness of the users' language.

Organisational aspects:

Will the system/component have to fit into some organisation or other, and how? If so, how is the system situated in the workflow in which it is embedded?

No.

Customer(s):

Who is the customer for the system/component (if any)?

This is a research project with no real users. The system was however intended for Danish travel agencies.

Users:

What are the intended users of the system/component? What are the aims in terms of usability/user training?

Presumably all fluent speakers of Danish, i.e. with no foreign accent. Novices as well as experts.

Developers:

How many people took significant part in the development? Did that cause any significant problems, such as time delays, loss of information, other (explain)? Characterise each person who took part in terms of novice/intermediate/expert wrt. developing the system/component in question and in terms of relevant background (e.g., novice phonetician, skilled human factors specialist, intermediate electrical engineer.)

Two skilled computer scientists and one human factors specialist worked more or less full-time developing the dialogue component. Other system components were developed at other sites in Denmark.

Development time:

When was the system developed? What was the actual development time for the system/component (estimated in person/months)? Was that more or less than planned? Why?

The project was said to go on between 1991 and 1995 with a total of 30 person-years. 10 of these accounted for the dialogue component. However, the speech recognition slowed things down, and when additional resources came in, they were used for user testing.

Requirements and design specification evaluation:

Were the requirements and/or design specifications themselves subjected to evaluation in some way, prior to system/ component implementation? If so, how?

No formal evaluation was made prior to implementation. The requirements were based on what domains and tasks current (1991) state-of-the-art systems could handle. The reservation task was new and therefore chosen.

Evaluation criteria:

Which quantitative and qualitative performance measures should the system/component satisfy?

For the relevant criteria below, state their definitions, describe the performance targets and state whether these were achieved.

(DM) Robustness - wrt. topic identification:

No topic evaluation has been done.

(DM) Robustness - wrt. unexpected deviations from the dialogue plan:

Not measured.

(DM) Number of turns: Average number of turns per dialogue. A high figure could indicate a low user acceptance. For human-human reservation task it was 20. This was not exceeded in the WOZ test nor in the user test.

(DM) Average and max. utterance length (for user and for system):

3-4 words average, 10 words maximum.

(DM) Average number of "long" turns per dialogue turns (for user and for system):

Longest turn, 12 words maximum, only three had more than 10 words.

(DM) Average number of word types and word tokens per dialogue:

A 500 word lexicon was enough in each test run, but there was little overlap, and hence the 500 words limitation was insufficient for a working system

(DM) Cumulative type/token ratio:

User's vocabulary did not converge in WOZ, indicating the 500 word lexicon being to small.

(DM) Number of questions in relation to total number of turns (for user and for system):

A rough measure of who (user or system) had the initiative. The interaction became more and more system directed due to lexicon constraints.

(DM) Complexity of interaction model, e.g. in terms of number of nodes if a graph representation is used:

This was used to measure how complicated the wizard's task was. Measured as the total number of nodes, nodes containing system phrases (including questions), and system questions. The work load increased as the interaction model became more well-specified.

(DM) Number of ad hoc generated phrases in relation to total number of turns (WOZ only):

An estimate over how well the pre-defined phrases in the interaction model covered the task domain. Also in some respect how well the wizard was performing. Sank drastically to below 20% as a tool was introduced for the wizard.

(DM) Average number of ad hoc generated jumps per dialogue (WOZ only, system-directed dialogue):

Error jumps and jumps missing in the interaction model graphs. Decreased as the wizard improved and the interaction model became more well-specified.

(DM) Naturalness - mixed initiative dialogue:

User's answers to questionnaires from WOZ resulted in 23% for flexibility and 45% for ease of making corrections, where a figure below 50% is negative.

(DM) Naturalness - no interaction problems:

Analysis of the user test corpus revealed that users had problems with the "recursive" style of task repair dialogues. Also problems with the keyword "change". Users also wanted to be able to provide more than one information piece at a time.

(DM) Transaction success:

Transaction success rate was measured in a user test. Success was defined as reservations carried out according to the scenario (or to the subjects mistaken understanding of the scenario). The non-success category included all failures to obtain a reservation, even if this was due to user error. Based on this conservative metric, an 86% success rate was achieved. The majority of the users however still preferred a human travel agent for this task. This is probably caused by the low flexibility by the system (only 23% considered this satisfactory). The designers assume this to be based on the rigid and system directed dialogue structure.

(DM) How successful is the system wrt. the interpretation of indirect speech acts, and how does this depend on the dialogue management approach chosen:

No indirect speech acts.

(HF) User satisfaction and other subjective parameters (explain):

(HF) Average time for task completion:

(HF) Do the system phrases conform to the cooperativity guidelines (individually as well as in context):

(System) Re-usability:

Robustness - other (explain):

Naturalness - other (explain):

Multimodality: Describe the evaluation of the multimodal aspects of the system, if any. Method(s)? Results? Was the evaluation procedure appropriate?

Evaluation:

At which stages during design and development was the system/component subjected to

testing? How? Results?

Empirical tests were performed both during the development and in a final evaluation. (For some reason that is not clear to us all kinds of empirical tests are called 'evaluation' in the report. We will here only confine ourselves with the final evaluation.)

A list of evaluation criteria is presented, however, they are, according to the team a reconstruction, and were never stated explicitly during the project.

Both WOZ testing and user tests were performed. Some of these data are given above.

Performance evaluation using WOZ.

During WOZ the system was made more and more system directed, i.e. less and less user initiative allowed. This also leading to less flexibility. This was due to the real-time constraint and the poor performance of the recogniser.

It was possible to restrict the users' linguistic behavior, both concerning language and dialogue, to what the system could handle. The task domain coverage was considered functionally acceptable, but limitations in the dialogue model prevented the system from managing more complicated user tasks. It is claimed that this can, within the current dialogue architecture, only achieve this to the price of a more cumbersome dialogue for simpler tasks. Alternatively the system must be re-designed to allow for mixed initiative dialogue.

The users do accept the system. However, at certain points they did not find it satisfactory. The dialogue model could not be much different given the limitations from the recogniser.

No traditional IR-analysis in terms of recall and precision is reported.

Regarding the test material. The scenarios given to the users are presented. Differences between graphical and verbal scenarios are discussed. Also the graphical scenarios also included the text from the verbal version.

Mastery of the development and evaluation process:

Of which parts of the process did the team have sufficient mastery in advance? Of which parts didn't it have such mastery?

The project seems at the beginning to have had most competence on general system development, on human factors, and on linguistic competence, whereas the knowledge in general (non-computational) dialogue theory was limited. The project is basically a bottom-up project, where evaluation of the steps taken has been the prime mover in the shaping of the final design.

Problems during development and evaluation:

Any major problems?

The major problem during the entire development was an unsufficient speech recogniser.

No evaluation of the linguistic part was done.

Development and evaluation process sketch:

Please summarise in a couple of pages key points of development and evaluation of the system/component. To be done by the developers.

Component selection/design:

Describe the components and their origins.

All developed in-house.

Robustness:

How robust is the system/component? Has this been measured? What has been done to ensure robustness?

Robustness is primarily achieved through having a system directed dialogue. But this robustness is bought at the price of less flexibility and difficulty in scaling up the system.

Maintenance:

How easy is the system to maintain, cost estimates, etc.?

The system is not designed for continuous use, and no measures have been taken to ensure an ease of maintenance, apart from the general benefits of having a well structured and modularised project.

Portability:

How easily can the system/component be ported?

The system was made to run on a Linux PC. To port it to e.g. Windows would be costly.

Modifications:

What is required if the system is to be modified?

As the task model is hard-wired it would probably be quite hard to make anything but smaller modifications. To substantially improve system performance and user acceptance, the system needs to be changed from system initiative to mixed initiative dialogue. This seems to require a thorough re-design of the dialogue component. Other improvements, such as a larger vocabulary, seem easier to achieve, since there is no need for re-design here.

Additions, customisation:

Has a customisation of the system been done? (Change of domain, language, etc.) How hard is this?

The developers find it hard to see how to re-use parts of the task structure for other domains, which makes a customisation costly.

Property rights:

Describe the property rights situation for the system/component.

The property rights belong to the developers.

2. References

Niels Ole Bernsen, Hans Dybkjær, and Laila Dybkjær. *Designing Interactive Speech Systems*. Springer Verlag, 1998.

7.5 RailTel/ARISE Dialogue Manager Grid

Arne Jönsson, Patrik Elmberg and Nils Dahlbäck

Department of Computer and Information Science
Linköping University, S-581 83 Linköping, Sweden
arnjo@ida.liu.se, x97patel@ida.liu.se, nilda@ida.liu.se

[The following analysis has not been verified by the developers.]

1. Dialogue Component Tasks

What are the main tasks of the Dialogue Component?

Control the dialogue.

Does the dialogue component only have a mediating function, as is usually the case with translation systems in which the system does not really act as a dialogue participant?

The dialogue manager controls all the other modules.

To which system requirements do the tasks of the dialogue component contribute (robustness, efficiency, etc.), and what does this contribution consist of?

It contributes to robustness as it handles meta-communication. It also contributes to efficiency in that the user is free to interrupt the system at any time during the dialogue.

2. Dialogue Component Architecture

a. *Architecture specification*

> What are the submodules of the dialogue management component of the system, what are their functions, and how do they interact (see also below)?
>
> Provide, if possible, a figure of the architecture of the dialogue management component, thereby indicating the interactions between the submodules.

The dialogue handling module consists of a dialogue manager which accesses a task model description and a dialogue model description. The dialogue model contains information for controlling the interaction whereas the task model contains information on the task. The dialogue manager consults the dialogue and task models to control the interaction.

b. *Generic architecture*

> Does the Dialogue Management component of the system have a generic architecture? Give an answer to this question insofar it is relevant to the grid analysis, thereby also indicating the level of genericity (language-independent architecture, domain-independent architecture, etc.).

The architecture is generic. The dialogue model is domain independent for information retrieval dialogues. The task model is domain dependent and has been configured for one

train and one flight information domain. The dialogue manager has also been used in a multi-modal system.

c. *Interaction with other system components*

Specify the input/output characteristics of the dialogue component.

The dialogue manager provides a dialogue act description to the response generator. The response generator is responsible for determining the system's response. However, clarifications are handled by the dialogue manager e.g. if there are too many items to list the dialogue manager asks for further specification.

The interpretation module does not utilise information from the dialogue manager.

3.1 The Implemented Approach

a. *Kind of descriptive approach*

Which of the following dialogue management approach(es) (some of which are partially overlapping) have been implemented in the dialogue system, this according to the descriptions given below?

Structured topical approach for the task model. Not hierarchical or sequential for the task. A dialogue grammar for the dialogue model.

b. *Use of single versus multiple approach*

Does the system comprise a multiple approach?

If the system uses a multiple approach, what are the reasons for assuming this multiple approach, what are the functions of the single approaches and what is their prominence (equally prominent, one subservient to the other, etc.)?

Single approach.

c. *Implementation characteristics*

Which characteristic approach-dependent options have been realised in the implementation?

Application dependent task model implemented in a semantic frame.

Application independent dialogue model implemented in a Finite State Machine (FSM).

d. *Use of hybrid identification and prediction methods*

Does the system comprise a hybrid approach, also making use of additional statistical methods?

Give a specification of the set of dialogue functions for which statistical methods are used (see also below), making explicit whether the method applies to all structural levels in the dialogue or just to a specific kind?

The Dialogue manager does not use statistical methods.

3.2 Dialogue Structure

A. Intentional Structure

Given the task-oriented character of the dialogues under consideration, we take the intentional structure of a dialogue as the (mostly hierarchical) structure of dialogue segment purposes to perform a (sub)task.

a. *Task characteristics*

(i) *Task specification*

Which specific tasks are performed by the system (making a reservation, showing a time table, fixing a meeting date, etc.)?

(ii) *Task structure*

Are all tasks well-structured, i.e. do all tasks have a stereotypical structure explicating both the amount and order of information updates needed to complete the task (see, in particular, Bernsen et al. 1997 on this point)?

Information on the SNCF static timetable as well as additional information about services offered on the trains, fare-related restrictions and supplements. The task model is categorised into several concepts: train time, fare, change, type, service, reduction and reserve. Task information was determined by analyses of queries taken from the training corpora and from knowledge on the information in the database.

b. *Communication types*

(i) *Domain communication*

Specify the relevant characteristics of the task-oriented domain communication, in particular whether and/or how the system accounts for the following phenomena (some of which in fact may give rise to meta-communication):

Repairs (repair types, repair success, etc.)

Clarifications (location-restricted/-unrestricted, system-/user-initiated, etc.)

Repetitions

Reasons ('nucleus reasons' vs. 'satellite reasons')

Deliberations (deliberation types)

Feedback (feedback realization types)

Other

Uses a mixed-initiative strategy, where users are free to ask any question at any time. The system takes initiative and prompts the user for any missing information. Experienced users are able to provide all information for database access in a single utterance, whereas less experienced users are allowed to provide shorter responses being more guided by the system.

No explicit system information generated to the user. However, during the simulations such information was provided before the interaction.

Different types of responses can be generated during the dialogue depending upon the dialogue structure: system presentations, prompts, restarts, requests for information, responses, reformulations, confirmations and domain explanations. This manages incomplete or underspecified user requests. Indirect speech acts does not occur as the interpreter only provides keywords to the dialogue manager.

According to the developers, the system never provides a negative response to the user, unless the information is really not available.

(ii) *Meta-communication*

Does the system allow for both system- and user-initiated meta-communication, and in what way?

How does the system behave with respect to different types of meta-communication, such as:

Repairs (repair types, repair success, etc.)

Clarifications (location-restricted/-unrestricted, system-/user-initiated, etc.)

Repetitions

Reasons ('nucleus reasons' vs. 'satellite reasons')

Deliberations (deliberation types)

Feedback (feedback realisation types)

Other

The system allows for both system and user initiated meta-communication. Fixed meta-communication in the FSM. The system selects the most relevant information in this static representation.

Feedback by repeating the current semantic task frame.

(iii) *Other communication types*

Dialogue introduction and dialogue closure phases (domain-independent communication)

Side structure information induced by digressions (related to, but out of the domain communication)

Closing phrase, and uncomplicated introductory phrase.

c. *Modelling of intentional structure*

How is the dialogue's intentional structure modelled in terms of the chosen dialogue management approach?

Although task-oriented dialogues have an underlying goal structure, these dialogues are not always modelled in terms of the goals to perform these tasks (see above).

(i) *Uniform versus partial application*

In case of a multiple approach, are all the single approaches uniformly applied to both the higher and lower intentional levels in the dialogue, or are they applied to just a specific part of this structure?

Single approach.

(ii) *Intentional levels*

How many intentional levels are distinguished?

a. *Intentional continuum*

Is every underlying discourse segment purpose represented separately in terms of the chosen approach?

As for a multiple approach, how do the containing approaches differ with respect to the phenomenon of an intentional continuum?

Does the system define a hierarchical order for the fine-graded intentional levels it distinguishes?

Note that the phenomenon of an intentional continuum characteristic for human-human dialogues is usually not fully accounted for in dialogue systems.

No explicit representation of a hierarchical order of intentions. Sub-dialogues are represented in the FSM.

b. *Representation of larger intentional units*

If only larger intentional segments are distinguished by the system, which are these and are they hierarchically structured?

Relevant questions wrt. the assumption of larger intentional units are the following:

How is the subdivision in intentional units motivated?

What motivation is given for the assumed basic dialogue units, specific classes of which constitute the higher-order levels?

How are basic dialogue units determined?

What is said about the relation between dialogue phases and the classes of basic dialogue units constituting these phases?

Which typology is defined for basic dialogue units? Indicate whether the set of basic dialogue units is an open set, whether the proposed typology contains domain-dependent basic units, etc.

The dialogue is divided into three phases: opening formalities, main information exchange and closing formality exchange. Each dialogue is structured into sub-dialogues with a particular functional value. Sub-dialogues are further divided into sub-dialogues concerning the task, the dialogue or meta-dialogues. Sub-dialogues concerning the dialogue are application-dependent and involve for instance opening formalities.

Meta-dialogues corresponds to parts of the discourse that do not directly concern the information enquiries.

Basic dialogue units were determined from the corpus.

The task structure triggers the sub-dialogue types.

d. *Dialogue parsing*

Give a description of the dialogue parsing process, accounting for at least the following points:

What type of dialogue parser has been chosen (a task dependent parser, a topic-dependent parser, etc.)?

What kind of parsing strategy has been followed (top-down or bottom-up strategy, complete or partial parsing strategies, etc.)?

Give a brief description of the parsing process, including a specification of parsing constraints (syntactic constraints, semantic constraints, prosodic constraints, contextual constraints, etc.).

How does the dialogue parser account for the isomorphism between intentional and linguistic structure? Does parsing automatically give rise to a representation of the linguistic structure?

How does the dialogue parser account for updates of the corresponding attentional states, either directly or indirectly?

FSM controls the dialogue and is the implementation of the dialogue model. No parser for the task model.

B. Attentional state

a. *Modelling of attentional state*

(i) *Local focus*

How does the system model local focus?

In what specific way does the modelling of local focus give rise to expectations for the next user input?

How do focus values selected by answers to local questions give rise to updates on the global level (see also above)?

How is local focus determined in case of other speech acts than questions, and how is it determined in case of extended answers to questions?

Of what types of linguistic data is made use to identify local focus (specific prosodic information, information about contextually determined word order, etc.)?

The current case-frame based on properties of the domain model represents focus of attention. It includes domain specific information such as train-time, fare, change, type, reserve, service and reduction. The semantic frame contains a set of slots instantiated by the meaningful words of the utterance.

The focus of attention is used to decide what action to take, e.g. asking for more information.

A formal grammar controls the interaction where non-terminals are conditioned by the context. At each user dialogue act, the response generator builds a sentence where gaps are filled in from the content of the current case-frame, the dialogue history and the response from the background system.

No expectations used. If the user violates the expectations the dialogue manager tries to fit the information into the task structure. For instance if the user to a request for departure time responds "to Paris", the departure location state will be altered.

(ii) *Global focus*

Does the system model global focus? If so, how does it model global focus in relation to local focus?

If the system makes use of expectations induced by global focus, what is their nature and impact?

Does the system make use of linguistic markers of global focus phenomena, e.g. topic shift markers which are relatively easy to implement?

The semantic frames from the previous interaction, user utterances as well as system questions, forms the "generation history".

No structured global focus. The unstructured set of semantic frames comprise the global focus.

b. *Additional expectations*

What other (statistical or other) methods are implemented to compute, in particular, the immediate local expectations which arise from a current system/user question?

What is the function of a multiple approach in this respect?

No statistical model.

c. *Co-operativity*

Task realization in dialogue is considered to be part of its dynamic, attentional structure. How does the system account for a co-operative performance of these tasks, e.g. with respect to the

following phenomena?

Under-informativeness (e.g. the system's exhaustiveness strategies)

Over-informativeness (e.g. the system's refinement strategies in case that the requested task is too broad)

Truthfulness of provided information (the system's information checking mechanisms)

Brief and orderly presentation of requested information (the system's presentation mechanisms)

If possible, indicate whether and how co-operative action initiation as part of these strategies is goal determined.

Under-informativeness is handled through clarification sub-dialogues.

The system repeats the current content of the semantic frame.

It is possible to interrupt the system's response.

C. Linguistic Structure

The linguistic structure of a dialogue constitutes its segmentation structure. The dialogue's linguistic structure is in correspondence with both the intentional structure of the dialogue and the structure of attentional states.

a. *Dialogue segments*

How many segmentation levels are distinguished?

Because of the correspondence between intentional structure and segmentation structure, dialogue segmentation within the system crucially depends on how many intentional levels are distinguished (see above).

Only one level, the FSM.

b. *Speech act types*

Which speech act types (e.g. questions, assertions, commands, etc.) can be handled by the system?

How are the different types identified?

How successful is the system with respect to the interpretation of indirect speech acts (see also above), and how does this depend on the dialogue management approach chosen?

Thirteen dialogue acts, e.g. confirmation request, answer, information request.

Combines formal grammars and speech act theory. The grammar non-terminals corresponds to sub-dialogues and the terminals to dialogue acts.

c. *(Co-)reference*

How does the system behave with respect to different linguistic realisations of the same referential object (pronouns, anaphoric descriptions, etc.)?

Does the system keep a history of co-referential expressions?

How does co-reference relate to the dialogue's intentional and attentional structure?

The system is keyword based and the interpreter does not forward co-referential information.

Ellipsis is implicitly handled through the task model. No pronoun resolution.

3.3 Dialogue Context

A. Dialogue History

a. *Representation of Intentional Structure*

Does the system incrementally construct a representation of the intentional structure as the result of the preceding context?

What is the format of representation of the intentional structure (tree representation, matrix representation, etc.)?

Which submodule of the dialogue component is responsible for this representation (e.g. the dialogue planner) and where is the representation stored (e.g. in the dialogue memory)?

Which other system component does the stored intentional information contribute to and how?

The intentional structure is modelled in the FSM.

b. *Representation of Attentional States*

Does the system provide an incremental representation structure of the set of attentional states constituting the dynamic development of the dialogue, thereby providing a representation of the set of open alternatives ('focus set') at any dialogue information state?

What is the format of representation (e.g. an updated SIL object or a list of such objects in Sundial and the Daimler-Benz dialogue system)?

How does the incremental representation structure of attentional states account for the set of *referentially accessible objects* at any dialogue state?

Which other system component does the representation of attentional structure contribute to and how?

The semantic frame which contains domain specific information is completed with the history of all previous semantic frames

The system forgets the dialogue history when the user explicitly changes the request. Furthermore, if the user modifies a constraint, all dependent constraints are removed from history.

The task is recorded in the dialogue history, as the case-frame information contains also task information.

c. *Representation of Linguistic Structure*

Does the system represent the segmentation structure of the preceding dialogue, and what is the relation with the incremental representation of the corresponding intentional structure?

Does the system also represent referential structure (see above)?

Which other system component does the represented linguistic structure contribute to and how?

No history of the linguistic structure.

B. User Model

a. *Modelling of performance-relevant aspects of the user*

To what extent does the system model performance-relevant aspects of the user, such as user beliefs (his own beliefs as well as beliefs about the system), user desires (preferences), user expertise, etc.? What is precisely their function within the system?

Does the system also allow for inferencing over initial beliefs?

What is the status of, in particular, the user beliefs, desires and intentions? And, what is most important, how do these interact in the process of dialogue control?

No model of user goals except that users are assumed to fulfil the goal to achieve information on train times.

No specific analysis or models on different user's cognitive abilities.

b. *Contribution to co-operativity*

What contribution is provided by the user model with respect to co-operative system behaviour?

Give an indication of the remaining, most relevant problems in this respect.

The strategy allows experienced users to state their request in one query while less experienced might need a more system-directed dialogue, but this is not explicitly modelled.

Inexperienced users can be asked to spell out if the system can not interpret the utterance.

C. Domain Model

a. *Data*

Give a specification of the data types determined by the task(s).

Provide the global characteristics of objects, properties and relations relevant to these tasks.

The domain is modelled in the case-frame, see above on Focus.

b. *Rules*

Give a global characterisation of world knowledge rules used by the system, as well as the characteristic properties of their application.

Rules for default values and transformation of missing values. Supplies default values not specified by the user, e.g. if departure month hasn't been specified, the current month is assumed. Transforms imprecise values given by the user into appropriate ones used by the system, e.g. "this morning" is transformed to "today between 6 am and 12 noon".

D. Dialogue Settings

a. *Physical setting*

What are the characteristic physical circumstances in which the dialogues take place (face-to-face situation, communication through electronic devices, etc.), and how is their influence on co-operative communication modelled in the system?

b. *Social setting*

In what type of communicative situation do the dialogues take place (expert-novice, employer-employee, etc.), what roles do user and system have in these situations, and how is social knowledge of this type modelled in the system?

Telephone system. Walk-up-and-use. Speaker independent allowing any French speaking

(or with a heavy American accent) to interact with the system.

7.6 Railtel/Arise Dialogue Manager Life Cycle

Patrik Elmberg, Nils Dahlbäck and Arne Jönsson

Department of Computer and Information Science
Linköping University, S-581 83 Linköping, Sweden
x97patel@ida.liu.se, nilda@ida.liu.se, arnjo@ida.liu.se

The RailTel system was developed from the multi-modal MASK system. Arise is a further development of RailTel for other languages than French.

[The following analysis has not been verified by the system developers.]

1. Life Cycle Questions, General

Overall design goal(s):

What is the general purpose(s) of the design process?

To develop a prototype telephone service for access to train travel information. Mixed initiative dialogue, offering reservations, and giving information on timetables, fares, and seating.

Hardware constraints:

Were there any a priori constraints on the hardware to be used in the design process?

A fast enough computer for real-time demands.

Software constraints:

Were there any a priori constraints on the software to be used in the design process?

Using a UNIX-environment.

Customer constraints:

Which constraints does the customer (if any) impose on the system/component?

For the MASK system the customer wanted the 350 most frequent stations. In the phone-based RailTel/ARISE system the additional real-time demand appeared.

Other constraints:

Were there any other constraints on the design process?

No other constraints.

Design ideas:

Did the designers have any particular design ideas which they would try to realize in the

design process?

To develop a task independent dialogue management module.

Designer preferences:

Did the designers impose any constraints on the design which were not dictated from elsewhere?

No.

Design process type:

What is the nature of the design process?

Research prototype.

Development process type:

How was the system/component developed?

Incrementally from the French ATIS and MASK to RailTel/Arise.

Requirements and design specification documentation:

Is one or both of these specifications documented?

None.

Development process representation:

Has the development process itself been explicitly represented in some way?

No.

Realism criteria:

Will the system/component meet real user needs, will it meet them better, in some sense to be explained, than known alternatives, is the system/component "just" meant for exploring possibilities, or what?

Currently no explicit evaluation on how the system meets user needs. However, currently MASK is being evaluated, but for RailTel no evaluation from the perspective of user needs is planned.

Functionality criteria:

Which functionalities should the system/component have?

Allow user to make reservations, ask for information, include barge-in. Real-time response. For the MASK system allow mix between touch-screen interface and speech.

Usability criteria:

What are the aims in terms of usability?

A walk-up-and-use system for a well-defined task domain.

Organisational aspects:

Will the system/component have to fit into some organisation or other, and how? If so, how is

the system situated in the workflow in which it is embedded?

No.

Customer(s):

Who is the customer for the system/component (if any)?

It was a joint project with the French railway company, SNCF.

Users:

What are the intended users of the system/component? What are the aims in terms of usability/user training?

A walk-up and use system for French speaking users.

Developers:

How many people took significant part in the development? Did that cause any significant problems, such as time delays, loss of information, other (explain)? Characterise each person who took part in terms of novice/intermediate/expert wrt. developing the system/component in question and in terms of relevant background (e.g., novice phonetician, skilled human factors specialist, intermediate electrical engineer.)

The dialogue manager was more or less developed by one skilled computer scientist. A human factors specialist acted as an advisor for the design of the touch-screen interface of the MASK system.

Development time:

When was the system developed? What was the actual development time for the system/component (estimated in person/months)? Was that more or less than planned? Why?

The dialogue manager was developed during 1993-94 roughly taking 8 person-months for the kernel and additional 3 person-months for the rules. The dialogue control module was developed as a thesis project in 1991.

Requirements and design specification evaluation:

Were the requirements and/or design specifications themselves subjected to evaluation in some way, prior to system/ component implementation? If so, how?

No.

Evaluation criteria:

Which quantitative and qualitative performance measures should the system/component satisfy?

For the relevant criteria below, state their definitions, describe the performance targets and state whether these were achieved.

(DM) Robustness - wrt. topic identification:

Not measured.

(DM) Robustness - wrt. unexpected deviations from the dialogue plan:

Not measured.

(DM) Number of turns:

Considered, but no data available.

(DM) Average and max. utterance length (for user and for system):

Considered, but no data available.

(DM) Average number of "long" turns per dialogue turns (for user and for system):

Not measured.

(DM) Average number of word types and word tokens per dialogue:

Not measured.

(DM) Cumulative type/token ratio:

Not measured.

(DM) Number of questions in relation to total number of turns (for user and for system):

Not measured.

(DM) Complexity of interaction model, e.g. in terms of number of nodes if a graph representation is used:

Not measured.

(DM) Number of ad hoc generated phrases in relation to total number of turns (WOZ only):

Not measured.

(DM) Average number of ad hoc generated jumps per dialogue (WOZ only, system-directed dialogue): Not measured.

(DM) Naturalness - mixed initiative dialogue:

Not measured.

(DM) Naturalness - no interaction problems:

Not measured.

(DM) Transaction success:

The dialogue errors were defined as the ratio of erroneous system responses to the total number of system responses.

(DM) How successful is the system wrt. the interpretation of indirect speech acts, and how does this depend on the dialogue management approach chosen:

Indirect speech acts treated as direct requests.

Evaluation:

At which stages during design and development was the system/component subjected to testing? How? Results?

Two tests were run. One using the implemented system, and one using a WOZ-experiment.

In the test of the implemented system 100 subjects were tested. Most recruited through an advertisement. Two scenarios were used, but each subject only used one of these. A post trial questionnaire was distributed.

Overall success rate was 72%. 14% of the failures are attributed to the dialogue management, though no details of what is included here is provided in our documentation. The distribution of these errors between the two scenarios is very uneven (0.9% vs. 10.3%). The higher figure is in fact in part due to a database error, but to which extent has not been checked.

The over-all evaluation of the system is favourable, with no differences between the two scenarios. Still, however, a majority prefers service from a person. Females gave somewhat higher scores on evaluation, but were less motivated to use such a system in the future. In the WOZ evaluation 20 native speakers interacted with the system (the instructions, scenarios and background of the subjects are not given in the paper). Only data on the absolute frequencies of the different dialogue acts are given. Only a few explanation acts are used by the subjects. The system's distribution of dialogue acts is, not surprisingly, a complement set of the user set, showing that the dialogue is very Q/A-oriented. The most important conclusions that can be drawn from this is that there are few needs to clarify questions etc., indicating a successfully working system. But since it is not known how strictly the Wizard followed the dialogue model to be implemented, the result is difficult to interpret.

Mastery of the development and evaluation process:

Of which parts of the process did the team have sufficient mastery in advance? Of which parts didn't it have such mastery?

The team had sufficient mastery in development in advance, but no explicit development process was used.

Problems during development and evaluation:

Any major problems?

The biggest problem going from MASK to RailTel was how to generate good system responses when only utilising a phone.

Development and evaluation process sketch:

Please summarise in a couple of pages key points of development and evaluation of the system/component. To be done by the developers.

Component selection/design:

Describe the components and their origins.

In house developed dialogue manager and knowledge bases.

Robustness:

How robust is the system/component? Has this been measured? What has been done to ensure robustness?

There are mainly two things to ensure robustness. Firstly, the possibility for the user to barge-in. This has not been measured, but a log file of users behaviour exists. Secondly, the system uses a robust parser. Clarification questions possible.

Maintenance:

How easy is the system to maintain, cost estimates, etc.?

Claimed to be fairly easy to maintain, but no cost estimates.

Portability:

How easily can the system/component be ported?

Fairly easy because of the use of common GNU compilers. A port to Windows 95/NT is

currently being made,

Modifications:

What is required if the system is to be modified?

Basically changing the task-structure, grammar and lexicon.

Additions, customisation:

Has a customisation of the system been done? (Change of domain, language, etc.) How hard is this?

Yes, going from MASK to RailTel (new domain) to Arise (new language).

Property rights:

Describe the property rights situation for the system/component.

7.7 VERBMOBIL

DISC
DIALOGUE COMPONENT GRID ANALYSIS

JAN VAN KUPPEVELT AND ULRICH HEID

University of Stuttgart
Institut für maschinelle Sprachverarbeitung
– Computerlinguistik –
Azenbergstrasse 12
D-70174 Stuttgart
(Jan.v.Kuppevelt@ims.uni-stuttgart.de; Uli@ims.uni-stuttgart.de)

Prefinal version. Analysis is based on the questionnaire
"DISC Dialogue Component Grid Questions - A Proposal"
(3rd version, April 1998).

Not verified by system developers.

May 1998

CONTENTS

b. Additional expectations
c. Co-operativity

C. Linguistic Structure

a Dialogue segments
b. Speech act types
c. (Co-)reference

3.3 DIALOGUE CONTEXT

A. Dialogue History

a. Representation of Intentional Structure
b. Representation of Attentional States
c. Representation of Linguistic Structure

B. User Model

a. Modelling of performance-relevant aspects of the user
b. Contribution to co-operativity

C. Domain Model

a. Data
b. Rules

D. Dialogue Settings

a. Physical setting
b. Social setting

1. Dialogue Component Tasks

- What are the main tasks of the Dialogue Component?

  1) Constraining the decisions to be made by other system components:

  - *Speech Recognition component*

    Contextual information in the form of top-down dialogue act predictions is used to constrain the search space of the Speech Recognition component, this by constraining the set of words which are likely to occur in the next utterance.

  - *Syntactic Analysis component*

    Contextual information (top-down dialogue act predictions) is also used to constrain the search space of the Syntactic Analysis component by narrowing down the set of applicable grammar rules to a specific subgrammar.

  - *Keyword Spotting component*

    The same type of contextual information (dialogue act predictions) is used for constraining the search space of the Keyword Spotting component, namely by delimiting its search space to the most probable keywords.

  - *Semantic Evaluation component*

    Contextual information is also used for constraining the search space of the Semantic Evaluation component, this in the sense that, e.g., during semantic evaluation contextual information is used to delimit the set of possible antecedents in the resolution of sentence external anaphora.

  - *Transfer component*

    Contextual information (discourse history information, in particular information on the current dialogue phase, the Introductory Phase, the Negotiation Phase or the Closing Phase) is used to disambiguate translation equivalents during transfer (Geht es bei Ihnen? which can be translated as Does it suit you? or How about your place?).

- *Generation component*

  Contextual information (referential structure information representing the various linguistic realisations of the same referential object) is used to control lexical variation in the generation of target language expressions.

2) Dialogue control for meta-communication, e.g. conduct meta-dialogues for clarification and (plan-based) repair.

3) Following the discourse in case that no translation is asked for, namely by shallow processing (see below).

- Does the dialogue component only have a mediating function, as is usually the case with translation systems in which the system does not really act as a dialogue participant?

In contrast to most other spoken language dialogue systems, the Verbmobil translation system only fulfils a mediating function in the (human-human) spoken dialogues involved, except in cases of, e.g., recursively embedded clarification dialogues which, in principle, can occur anywhere in the dialogue and in which the system functions as a real dialogue participant, implying the fulfilment of a dialogue control function.

- To which main system requirements (robustness, efficiency, etc.) of the overall system do the tasks of the dialogue component contribute?

The dialogue component of Verbmobil contributes to the following three system requirements:

1) Efficiency

The requirement of real-time processing calls for efficient dialogue processing. The dialogue component contributes to this in the sense that the contextual information it represents constrains the decisions of the other system components (see first point above).

2) Robustness

Spoken language processing requires robust processing methods which can deal with unreliable and (grammatically) incomplete system input. Robust

processing is enabled by the contextual information represented in the dialogue component, as well as by taking not sentences but dialogue acts as the basic information units.

3) Translations on demand

The requirement of translations on demand has given rise to the following two processing methods:[1]

- *Deep processing*

  Deep processing only occurs in case that one of the discourse participants wants a translation. In this case the system input goes through all relevant components, including the dialogue component.

- *Shallow processing*

  Shallow processing occurs when both dialogue participants speak the same language. In this case the system input first goes through the keyword spotter and then through the dialogue component (dialogue act identification is now made on the basis of key words only), this in order to follow the dialogue and to be able to provide predictions in case that translations are needed.

2. Dialogue Component Architecture

a. *Architecture specification*

- What are the submodules of the dialogue management component of the system?

  The dialogue management component consists of the following three submodules:

  1) A Dialogue Planner

  2) A Finite State Machine

  3) A Statistic submodule

[1] Maximally 50% of the dialogue is processed in depth, which is the case if the owner only speaks German (VM Report 49, January 1995).

- What are their functions?

  Function of the Dialogue Planner

  The Dialogue Planner contains a knowledge-intensive implementation of the dialogue model. The Dialogue Planner is responsible for the incremental representation of the underlying goal, thematic and referential structure. Apart from this, its main function consists in handling inconsistencies of an incoming dialogue act with what is predicted on the basis of the dialogue model.

  Function of Finite State Machine

  The Finite State Machine contains another implementation of the dialogue model. Its main function is to signal inconsistencies of a new dialogue act within the standard dialogue model.

  Function of the Statistic submodule

  The function of the statistic submodule is the prediction of follow-up dialogue acts on the basis of the history of previous dialogue acts, thereby using knowledge about dialogue act frequencies in the training corpus.

- How do they interact?

  Provide an architecture diagram and a reference to it, and indicate the internal information flow within the dialogue manager.

  The Finite State Machine signals the Dialogue Planner when an inconsistency has occurred, so that it can activate (plan-based or statistical) repair techniques. The dialogue act predictions provided by the statistic submodule are used by the Dialogue Planner for constructing an incremental representation of the dialogue structure.

  An architecture diagram can be found in several Verbmobil reports, e.g. Verbmobil Report 50 (December 1994).

b. *Generic architecture*

- Does the dialogue management component of the system have a generic architecture? Give an answer to this question insofar it is relevant to the grid analysis, thereby also indicating the *level* of genericity (language-independent

architecture, domain-independent architecture, etc.).

While the architecture of the dialogue component of the system may be generic, the basic transfer units which are central to it are not. In the Verbmobil translation system the basic transfer units are dialogue acts. A distinction is made between domain-dependent and domain-independent dialogue acts, the latter of which are not primarily defined in terms of the illocutionary property (a request, a suggestion, etc.) that is central to the domain-independent dialogue acts but rather in terms of refinements of these dialogue acts in terms of their functional role or propositional content (e.g. ACCEPT_DATE, ACCEPT_LOCATION, etc.). The domain-dependent dialogue acts makes straightforward portability to other kinds of negotiation domains impossible. However, in Verbmobil dialogue acts may be domain-dependent in another way, namely wrt. the dialogue phase in which they may occur, e.g. the dialogue acts SUGGEST, ACCEPT and REJECT which are typical to occur in the negotiation phase of the dialogues in question. In this respect straightforward portability to domains other than those determined by some kind of negotiation is excluded at all.

c. *Interaction with other system components*

- Specify the input/output characteristics of the dialogue component.

1) Deep processing input/output characteristics

In case of deep processing the dialogue component of Verbmobil may receive input from the Semantic Evaluation component of the system and may send its output to the Speech Recognition, the Semantic Evaluation, the Generation (German) and/or the Speech Synthesis (German) system components.

2) Shallow processing input/output characteristics

In case of shallow processing the dialogue component of Verbmobil only maintains a bi-directional information flow with the Keyword Spotting system component.

System architecture diagram

See, e.g., VM Report 65 (April 1995) for a diagram on the architecture of the Verbmobil system, also indicating the uni- and bi-directional information flow between system components.

3. Dialogue Management

3.1 THE DIALOGUE MANAGEMENT APPROACH

a. *Kind of approach*

- *The underlying theoretical model*

  Which theoretical approach or approaches to modelling dialogue (structure) underlie the implementation of the dialogue management component, this according to the descriptions given below as far as the developers see themselves bound to a given approach?

  The Verbmobil translation system underlies three different theoretical approaches to dialogue management, namely a plan-based approach, a dialogue grammar approach and a statistic approach.

  The approaches are not equally prominent: both the dialogue grammar approach and the statistic approach are subservient to the plan-based approach the implementation of which is responsible for, among other things, the incremental representation of contextual information (see section 3.2 below) and the treatment of different repair techniques (see, in particular, 3.2 A point b.-(ii)).

  The motivation for the multiple approach is the following. According to the developers (VM-Report 65, April 1995), "a planning-only approach is inappropriate when the dialogue is processed only partially", and "if we use structural knowledge sources like plans or dialogue grammars, top-down predictions are difficult to make, because usually one can infer many possible follow-up speech acts from such knowledge sources that are not scored".

- *Computational tools used*

  Which computational tools are used for implementation, possibly also making use of tools developed within the framework of other theoretical approaches? Give a specification of the tools used in relation to the adopted approach, e.g. dialogue grammar techniques, plan-based techniques (non-dynamic or incremental plan recognition/formation techniques, multi-level planning/plan recognition techniques accounting for both domain and discourse plans), word spotting techniques for identifying topic units, dialogue acts, etc.

  The three different approaches to dialogue management are computationally accounted for in terms of three different computational techniques carried out by the three submodules of the dialogue component:

1) The Dialogue Planner

The Dialogue Planner makes use of the basic plan-based technique in which (linguistic and non-linguistic) dialogue actions, i.e. those giving rise to new (updated) dialogue states, are modelled with plan operators.

The plan operators used are characterised by the following:

- *Constraint-based application*

  The application of plan operators is constrained by contextual and pragmatic factors.

- *Hierarchy of plan operators*

  Planning proceeds in a top-down fashion and implies a hierarchy of plan operators. The top-level plan operator comprises the top-level main dialogue goal of scheduling a meeting. The top-level plan operator is directly composed of the sub-plan operators comprising the three sub-goals which are responsible for the treatment of the dialogue segments 'Introductory phase', 'Negotiation phase' and 'Closing phase'. These subgoals have to be fulfilled in this order. At the lowest level, the most specific sub-plan operators correspond to individual dialogue (speech) acts.

- *Actions slot*

  Plan operators contain a slot for specific actions which are induced after a successful fulfilment of the subgoals, e.g. the action of updating the thematic structure of the dialogue. For instance, in the Offer-operator which is responsible for a dialogue act SUGGESTION the action Retrieve-theme filters the relevant information and updates the thematic structure represented in the dialogue memory.

  In general, the action slots associated with plan operators are responsible for the incremental construction of both the thematic and referential structure corresponding with the goal structure of the dialogue (see sections 3.2 point B. and 3.2 point C.).

2) The Finite State Machine

The Finite State Machine variant of the dialogue grammar provides a description of the set of admissible speech acts in the type of dialogues involved (appointment scheduling dialogues). Its functions are the following:

- *it parses the set of speech acts provided by the preceding context*

- *it checks whether a new speech act is in agreement with that which can be expected on the basis of the standard dialogue model*

- *it signals the Dialogue Planner in case of an inconsistent dialogue state*

- *it also allows for speech acts which might occur at each dialogue state such as (recursively embedded) clarifications, deliberations and reasons*

See, e.g., VM Report 50 (December 1994) for a diagram of the Finite State Machine implementation of the dialogue model for appointment scheduling dialogues.

3) The statistic submodule

As statistical techniques are used the well-known language model techniques from the field of speech recognition. In speech recognition these models are commonly used to reduce the search space when determining a word that can match a part of the input. However, the units to be processed here are not words but dialogue acts.

The main task of the statistic submodule is the prediction of a new dialogue act on the basis of the history of previous dialogue acts.

The most probable dialogue act d_j satisfies

$$P(d_j \mid d_1, \ldots, d_{j-2}, d_{j-1}) = \max P(d_1, \ldots, d_{j-2}, d_{j-1})$$

where $d_1, \ldots, d_{j-2}, d_{j-1}$ represents the history of d_j.

As for an approximation of determining the probabilities of

(arbitrary) long sequences of dialogue acts, the deleted interpolation method using n-grams[2] is used. The n-gram probability $P(d_j \mid d_{j-n+1}, ..., d_{j-1})$ approximates the required probability $P(d_j \mid d_1, ..., d_{j-2}, d_{j-1})$. Since even short histories very often are not in a training set, the probability is interpolated by combining histories of different lengths n, thereby multiplying each probability by a fixed weight q_i and the n-gram relative frequency f_n of dialogue acts.[3] Considering 1-, 2- and 3-gram frequencies of speech acts we only get

$$P(d_j \mid d_{j-2}, d_{j-1}) = q_1 f(d_j) + q_2 f(d_j \mid d_{j-1}) + q_3 f(d_j \mid d_{j-2}, d_{j-1})$$

where f are the relative frequencies and the sum of weights q_i is 1 ($\Sigma q_i = 1$). The relative frequencies f_n are derived from the training corpus annotated with the dialogue acts of the utterances. Different methods have been introduced to determine the model weights q_i (see, e.g., VM Report 151, August 1996).

b. *Use of statistical identification and prediction tools*

- Does the dialogue management component comprise a special submodule containing a statistical implementation of the dialogue model besides a knowledge-intensive submodule containing, e.g., a plan-based implementation? (Note that the latter one itself may also make use of statistical methods.)

  The Verbmobil system makes use of hybrid 3-layered approach to dialogue management. Besides the Dialogue Planner and the Finite State Machine, both of which make use of different knowledge sources, the statistical submodule completely relies on numerical information.

- Give a specification of all dialogue functions for which statistical methods are used, making explicit whether the method applies to all structural levels in the dialogue or just to a specific kind.

  The Dialogue Planner intensively makes use of linguistic information; the Statistic Submodule only makes use of numerical information. The Finite State Machine "can be located in between these two components" (VM Report 50, December 1994).

---

[2] In this context an *n*-gram is a sequence of *n* subsequently uttered dialogue acts which serves as a *shortened history* for predicting the most probable next dialogue act.

[3] The relative frequency f_n is the number of occurrences of the respective *n*-gram (say the sequence of dialogue acts $d_1, ..., d_n$) in the training corpus divided by the number of occurrences of the *(n-1)*-gram ($d_1, ..., d_{n-1}$).

3.2 DIALOGUE STRUCTURE

A. Intentional Structure

Given the task-oriented character of the dialogue systems under consideration, we take the intentional structure of a dialogue as the mostly hierarchical structure of common goals ('discourse segment purposes') to perform a (sub)task.

a. *Task characteristics*

(i) *Multiple tasks*

Does the system perform more than one overall task?

No. The Verbmobil system itself does not perform a dialogue task at all, at least not in the standard sense in which a dialogue system acts as a real participant in the dialogues it processes. The Verbmobil system is a translation system providing translations on demand for the human-human task-oriented dialogues it processes. However, it also processes these dialogues in case that no translation is asked for (shallow processing), involving, among other things, a representation of the single overall task that is communicated in these dialogues.

(ii) *Task specification*

Which (overall) tasks are performed by the system (e.g. making a flight ticket reservation in the Danish Dialogue System, showing a time-table in Waxholm and fixing a meeting date in Verbmobil)?

The Verbmobil system provides translations for dialogues the main task of which is scheduling a meeting.

(iii) *Task structure*

Are all tasks well-structured, i.e. do all tasks have a stereotypical structure explicating both the amount and order of information updates needed to complete the task (see, in particular, Bernsen, Dybkjaer & Dybkjaer 1998 for more information on this point)?

The task communicated in Verbmobil's human-human dialogues is well-structured. The structure of this task, i.e. fixing a meeting date, is expressed

by the dialogue model that underlies both the Dialogue Planner and the Finite State Machine. The latter, e.g., in fact defines the set of possible follow-up dialogue states at any point in the development of the dialogue, though it also accounts for deviations (digressions, clarifications, deliberations, etc.) which, by definition, may occur everywhere in the dialogue.

b. *Communication types*

(i) *Domain communication*

Specify the relevant characteristics of the task-oriented domain communication, in particular whether and/or how the system accounts for the following phenomena (some of which in fact may give rise to meta-communication):

- System directed, user directed or mixed initiative communication

  Because of its mainly mediating function (providing adequate translations on demand instead of being a dialogue participant), the Verbmobil system does not control the exchange of task-relevant information between the users.

- Free or bound order of main tasks

  The overall task of fixing a meeting date is hierarchically composed of a set of subtasks the order of which is not free but bound, namely as defined by the underlying dialogue model (see 3.2 A. point c. and 3.3 A. point a. below for the system's modelling and representation of the dialogue's intentional structure).

- Discontinuous user input ('input gaps')

  Discontinuous user input to the system involving large gaps in the expected sequence of dialogue acts is problematic for Verbmobil. In case of small input gaps dialogue act assignment by the Semantic Evaluation component may be blocked, as a consequence of which dialogue act prediction by the Keyword Spotting component will fill in missing dialogue acts. However, the technique used by the Keyword Spotting component is insufficient in case that the gaps are larger than two dialogue acts.

- Contextual inferences of different types

The Verbmobil system does destructively allow for inferencing over user input. Inference rules are introduced accounting for some specific inferential phenomena, namely that (i) a counterproposal (for a date, etc.) implies the rejection of a previous proposal, (ii) a new proposal (for a date) implies the acceptance of a (not incompatible, less specific) previous proposal and (iii) a change of dialogue phase implies the acceptance of a previous proposal (for a date).

- Indirect speech acts

A problem for most spoken language dialogue systems is the determination of indirect speech (dialogue) acts. In the case of indirect speech acts the intention underlying the utterance is not (fully) expressed in the surface form. Using the keyword spotting technique only is thus useless in these cases. As a general solution it has been proposed within the Verbmobil project to account for indirect speech acts in terms of semantic and pragmatic information. Until now, however, a solution has only been found for a small subset of indirect speech acts, in particular those associated with the inferential phenomena referred to above (implicit acceptance in the case of a new, but more specific proposal for a meeting date, implicit rejection in the case of a counterproposal for a meeting date and implicit acceptance in the case of phase change).

- Incomplete, underspecified user requests

Not applicable (on the level of domain communication the Verbmobil translation system does not act as a dialogue participant).

- Incomplete answers to system questions

Idem.

- Overcomplete answers to system questions

Idem

- Ambiguous user utterances

During translation the system aims at preserving possible ambiguities from the source to the target language. Disambiguation techniques only apply if parts of the system have given rise to translation equivalents for a (non-ambiguous) utterance in the source language.

(ii) *Meta-communication*

System-initiated meta-communication

Does the system initiate meta-communication, and in what way?

Yes. System initiated repairs and system requested repetitions may involve (human-machine) clarification dialogues which are initiated by modules that detect an error or inconsistency (see relevant points below).

User-initiated meta-communication

Does the system allow for user-initiated meta-communication? Make sufficiently explicit in what way.

Yes. In Verbmobil human-human meta-communication which is characterised as 'deviations' consists of reasons, deliberations, digressions and (human-human) clarifications.

Types of meta-communication

How does the system behave with respect to different types of meta-communication, such as:

- Repairs (repair types, repair success, etc.)

Repairs are induced as the result of unexpected system input, namely if an incoming dialogue act is incompatible with the dialogue model. Repairs are treated by the Dialogue Planner and detected by The Finite State Machine. The Finite State Machine signals the Dialogue Planner when an inconsistency has occurred, so that it can activate repair techniques.

The Verbmobil system makes use of two different repair techniques (repair success unknown).

(i) Statistical repair

The statistical repair technique makes use of statistical information provided by the Statistic submodule of the dialogue component. It consists in computing the most probable connection between the current dialogue state (dialogue act) and previous dialogue states (dialogue acts).

Example

Statistical repair occurs, e.g., if in the current dialogue state the dialogue act REJECT does not (as predicted by the Finite State Machine) follow a dialogue act SUGGEST but a dialogue act INIT which introduces the topic to be negotiated.[4] *In this case the Dialogue Planner tries to repair this state using statistical information, finding a dialogue act which most probably connects INIT and REJECT. Statistical repairs like these involve multiple dialogue acts implying the simultaneous occurrence of more than one dialogue act with one utterance (in the example given the assignment of both the dialogue acts INIT and SUGGEST resolve the incompatibility in question).*

(ii) Plan-based repair

If no statistical solution is possible plan-based repair is used, giving rise to the activation of a plan operator. Like in the case of statistical repairs, the plan-based repair technique is induced if the Finite State Machine has detected an inconsistency of the current dialogue act with what is predicted on the basis of the dialogue model. Depending on the type of dialogue act specific plan (repair) operators are used.

Example

The plan-based repair technique is used if the current dialogue act occurs within the 'wrong' dialogue phase (e.g., in case that a dialogue act of the Introduction phase occurs during the Closing phase, it is assumed that the latter has been terminated irregularly and that the current phase is a new introductory one).

- Clarifications (location-restricted/-unrestricted, system-/user-initiated, etc.)

a. Human-machine clarification dialogues

As said, system initiated repairs and requests for repetitions may involve clarification dialogues between user and machine. These clarification dialogues are induced at any place in which the system detects an error. Characteristics of the human-

[4] The dialogue act INIT can possibly be intervened by a dialogue act DELIBERATE which can occur at any point in the dialogue. In the repair process the latter one can be omitted because of its non-task oriented function.

machine clarification dialogues are the following:

- *no long clarifications are performed*
- *only simple yes/no answers to system questions are allowed*
- *if the problem cannot immediately be solved, the user is asked to repeat his/her utterance to be translated*

b. Human-human clarification dialogues

As said, human-human clarification dialogues are accounted for as deviations which may occur everywhere in the dialogue. For reasons of efficient processing (avoiding to test for every utterance whether it is one implying a deviation) deviations are treated as unforeseen dialogue acts inducing a (plan-based) repair operator (see above).

- Explanations

  Explanations are also treated as deviations.

- Repetitions

  Repetitions asked for by one dialogue participant to the other are treated as deviations too.

- Reasons ('nucleus reasons' vs. 'satellite reasons')[5]

  Also reasons are treated as deviations, though those providing a task-relevant value are not accounted for at all.

- Deliberations (deliberation types)

  Deliberations are also treated as deviations.

- Corrections (correction types)

[5] As opposed to satellite reasons, reasons occurring in a nucleus dialogue segment must be considered to belong to domain communication because of the property of providing an intentional value.

Until now the system does not account for corrections.

(iii) *Other communication types*

How does the system behave wrt. other communication types, such as:

- Dialogue introduction phases (domain-independent communication), e.g. greetings and introduction of dialogue participants

  In Verbmobil the Introduction phase consists of the following dialogue acts: GREETING, INTRODUCTION (of dialogue participants), POSITION (information about professional status) and INITIALISATION (introduction of the topic of the conversation, fixing a meeting date).

- Dialogue closure phases (domain-independent communication), e.g. final confirmation, thanks and saying goodbye

  In Verbmobil dialogue acts belonging to the Closure phase are: CONFIRMATION (final agreement), THANKS and BYE (dialogue closure) or INITIATION (of a new topic).

- Side structure information induced by digressions (related to, but out of the domain communication)

  Digressions are treated as deviations the location of which is unrestricted and the determination of which is realised by means of a repair operator.

c. *Modelling of intentional structure*

How is the dialogue's intentional structure modelled in terms of the chosen dialogue management approach, e.g. a dialogue grammar, topical or plan-based approach (see section 3.1 above)? Note that although task-oriented dialogues have an underlying goal structure, these dialogues are not always modelled in terms of the goals to perform these tasks.

(i) *Uniform versus partial application*

Do the chosen theoretical approaches of modelling dialogue underlie all intentional levels of the dialogue, or just a specific part of this structure?

In Verbmobil both the plan-based approach and the Finite State Machine capture all intentional levels.

(ii) *Intentional levels*

How many intentional levels are distinguished in the dialogue component of the system?

a. *Intentional continuum*

- Is every intentional unit represented separately in terms of the chosen approach or is the intentional continuum split up in just some main intentional levels?

  The Verbmobil system in which the plan-based approach is central does not represent each intentional unit separately but makes a distinction between four fundamental levels of representation.

- Does the system define a hierarchical order for the intentional levels it distinguishes?

  Yes.

b. *Representation of larger intentional units*

If only larger intentional units are distinguished by the system, which are these and are they hierarchically structured?

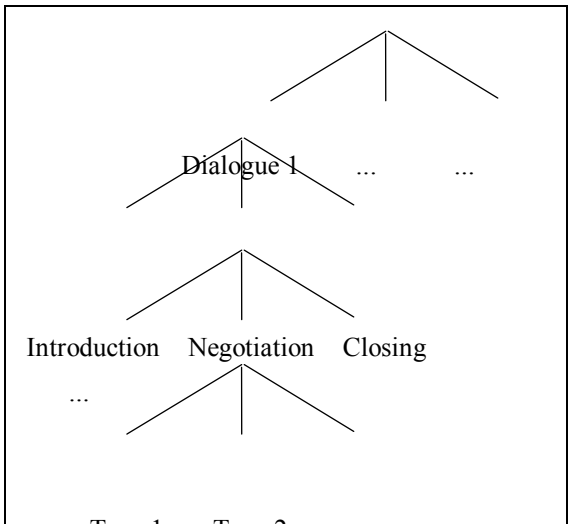
The Verbmobil system distinguishes the following hierarchy of intentional units, implying the assignment of a corresponding structure to the dialogue act sequences constituting a dialogue:

1) the highest intentional level of individual dialogues for fixing a date

2) the lower level of dialogue phases (the main phases are: Introductory Phase, Negotiation Phase and Closing Phase)

3) the level of dialogue turns

4) the lowest level of dialogue acts



Questions wrt. the assumption of larger intentional units are the following:

- How is the subdivision in intentional units motivated?

  No explicit motivation is given in this respect.

- What motivation is given for the assumed basic dialogue units, specific classes of which constitute the higher-order levels?

  Dialogue acts have been chosen as the basic transfer units.[6] *In contrast to written language, spoken language deals with the phenomenon of incomplete sentences (concatenation of sentence fragments) that are grammatically incomplete (e.g. elliptical sentences) and/or even grammatically incorrect (e.g. restarts). Sentences can thus not be chosen as the basic transfer units.*

- How are basic dialogue units determined?

  Dialogue acts are determined both during the system's shallow and deep processing:

  - *Shallow processing*

    In the case of shallow processing dialogue act

[6] Dialogue turns are thus not considered to be the basic transfer units; a dialogue turn consists of one or more successive dialogue acts.

determination is realised by the Keyword Spotter which identifies dialogue acts on the basis of key words only.

- *Deep processing*

  In the case of deep processing dialogue acts are determined on the basis of two additional methods.

The two dialogue act recognition methods used

(i) Statistic dialogue act recognition

As said, the Statistic submodule provides predictions on follow-up dialogue acts on the basis of previous dialogue acts, thereby using knowledge about dialogue act frequencies in the training corpus.

(ii) Knowledge-based dialogue act recognition

Knowledge-based dialogue act recognition which is based on the so-called FLEX system (Description Logic system; Berlin project group) makes use of weighted default rules, taking into account different types of information, e.g.:

- *syntactic information (sentence type, verb modus, etc.)*
- *keywords (discourse particles like 'leider')*
- *semantic information (conceptual information)*
- *micro-structural information (conventional information provided by the local context determining which syntactic and semantic structures can be used in order to express a certain dialogue act)*
- *macro-structural information (complementary conventional information provided by the global context expressing preferences for the next dialogue act)*

Note that dialogue act recognition is not a 1-1 activity; more

dialogue acts can be associated with the same unit.

- What is said about the relation between dialogue phases and the classes of basic dialogue units constituting these phases?

  See next point.

- Which typology is defined for basic dialogue units? Indicate whether the set of basic dialogue units is an open set, whether the proposed typology contains domain-dependent basic units, etc.

  - *The set of dialogue acts is a limited but open set. 54 dialogue act types (of which about 18 are domain-independent) were derived from a corpus of 200 dialogues annotated with these types (state of the art according to VM Report 65, April 1995).*[7, 8, 9]

  - *A distinction is made between domain-independent and domain-dependent dialogue acts, implying a distinction in ontology (see, e.g., VM Report 65 for the hierarchy of dialogue acts that is assumed):*

    a. *Domain-independent dialogue acts*

       Domain-independent dialogue acts are mostly of a illocutionary nature. Examples are: REQUEST, SUGGEST, ACCEPT and REJECT.

    b. *Domain-dependent dialogue acts*

       Domain-dependent dialogue acts are modelled as refinements of domain-independent dialogue acts, this according to their functional role or propositional content (ACCEPT_DATE; ACCEPT_LOCATION).

---

[7] The dialogue model covers approximately 90% of the 200 transcribed dialogues (VM Report 81, May 1995).

[8] The dialogue annotation scheme used was evaluated both according to *reproducibility* (coders agreement on dialogue act assignment) and *stability* (replicability in terms of one coder carrying out the same tasks twice). See for further information on this point VM Report 193 (April 1997).

[9] During the development the 200 annotated dialogues of the Verbmobil corpus functioned as both training and test data.

- *Another distinction is made between:*

  a. *Dialogue phase specific dialogue acts*

     Dialogue phase specific dialogue acts are the following:

     - *Introductory phase*

       The introductory phase consists of the following dialogue acts: GREETING, INTRODUCTION (of dialogue participants), POSITION (information about professional status), INITIALISATION (introduction of the topic of the conversation, e.g. to find a date for a business meeting).

     - *Negotiation phase*

       Dialogue acts which belong to this phase are, e.g., SUGGESTION (of time frames and refinement of time frames), REJECTION and REQUEST_FOR_ SUGGESTION.

     - *Closing phase*

       Dialogue acts belonging to this phase are: CONFIRMATION (final agreement), THANKS and BYE (dialogue closure) or INITIATION (of a new topic).

  b. *Deviations*

     Deviations are dialogue acts that in principle can occur anywhere in the dialogue. Examples are: GIVE_REASON, DELIBERATE, FEED-BACK, CLARIFY and DIGRESS.

- *As for contextual inferencing, a distinction is made between:*

a. Direct dialogue acts

In the case of direct dialogue acts the purpose of the utterance is expressed by its surface form.

b. Indirect dialogue acts

In case of indirect dialogue acts, on the other hand, the purpose of the utterance is not expressed by surface cues and must be inferred by the hearer.

d. *Dialogue parsing*

Give a description of the dialogue parsing process, accounting for the following points:

- If not already answered (section 3.1), what type of dialogue parser has been chosen (a task-dependent parser, a topic-dependent parser, etc.)?

  See section 3.1.

- What kind of parsing strategy has been followed (top-down or bottom-up strategy, complete or partial parsing strategies, etc.)?

  Idem.

- Give a brief description of the parsing process, including a specification of parsing constraints (syntactic constraints, semantic constraints, prosodic constraints, contextual constraints, etc.).

  Dialogue turns, e.g., which consist of one or more successive dialogue acts are parsed into dialogue act units ('utterances') on the basis of polygrams and multi-layer perceptrons, thereby also making use of prosodic information (VM-Report 130, 1996).[10] *See also section 3.1.*

- How does the dialogue parser account for the isomorphism between intentional and linguistic structure, in particular does parsing automatically give rise to a representation of the linguistic structure?

[10] In Kasper & Krieger (1996) (VM Report 141) turns are alternatively segmented into *sentential* or *phrasal* segments, using both grammatical and prosodic information.

Yes, the lowest intentional level consists of dialogue acts actually occurring in the dialogue at hand.

- How does the dialogue parser account for updates of the corresponding attentional states, either directly or indirectly?

  Plan operators contain slots for 'actions' consisting in updating the thematic structure of the dialogue.

B. Attentional state

The term attentional state is used to refer to the *current dialogue information state* and comprises the current subtasks which are not yet closed off, i.e. both the most recent (sub)task currently dealt with and the set of superimposing (sub)tasks to which this subtask contributes. The attentional state is dynamic, representing the (remaining) set of objects, properties or relations in *focus of attention* at each point in the development of the discourse. Following Sidner (1979) and Grosz & Sidner (1986) a distinction can be made between *local focus* ('focus prior' or 'immediate focus') and *global focus*, referring to the distinction between what is brought into focus of attention by the current subtask addressed by, e.g., the latest question and that which, at the same time, is still in focus of attention as the result of previous superimposing subtasks.

a. *Modelling of attentional state*

(i) *Local focus*

- How does the system model local focus?

  Like global focus, local focus is modelled in terms of the thematic structure of the dialogue. The thematic structure represents the hierarchical structure of temporal objects, namely months, weeks, days, etc. These temporal objects constitute the themes in the appointment scheduling dialogues. Local focus is the current theme, i.e. the suggested date or time frames under discussion.

- In what specific way does the modelling of local focus give rise to expectations for the next user input?

  Expectations are determined according to what can be expected to occur within the current theme (the date suggestion under discussion) on the basis of the underlying dialogue model. The suggestion of a new date, e.g., gives rise to the expectation that the next dialogue act is a confirmation or rejection rather than a new suggestion.

- How do focus values selected by answers to local questions give rise to updates on the global level?

  The evaluation of suggested temporal objects by the dialogue participants (time frame possible – POS; time frame impossible – NEG) gives rise to updates on the global level in terms of the set of remaining alternative dates for an appointment. The rejection of a suggested date, e.g., gives rise to an update on the global level in terms of the remaining alternative dates that are still open for further negotiation. Every further stage in the process of negotiation implies a further reduction of the set of alternative dates, thereby bringing the dialogue participants closer to the actual realization of fixing a date.

- How is local focus determined in case of other speech acts than questions, and how is it determined in case of extended answers to questions?

  Local focus is determined by identifying the dialogue act suggestion which has brought into focus the temporal object under discussion.

- Of what types of linguistic data is made use to identify local focus (specific prosodic information, information about contextually determined word order, etc.)?

  No other data than those used for identifying dialogue acts which suggest a new date or time frame for an appointment.

(ii) *Global focus*

- Does the system model global focus? If so, how does it model global focus in relation to local focus?

  Local focus is the theme (the alternative date or time frame) under discussion; global focus is the remaining set of alternative themes which are still open for discussion.

- If the system makes use of expectations induced by global focus, what is their nature and impact?

  Dialogue act prediction using global focus information is part of the set of macro-structural conventions (see 3.2 A point c. above) giving rise to dialogue act preferences induced by the global context.

- Does the system make use of linguistic markers of global focus phenomena, e.g. topic shift markers which are relatively easy to implement?

  Yes, but it does not make use of the full range of data in this respect.

b. *Additional expectations*

- What other (statistical or other) methods are implemented to compute, in particular, the immediate local expectations, e.g. those which arise from a current system/user question?

  See point on dialogue act determination in section 3.2 A above.

c. *Co-operativity*

Task realization in dialogue is considered to be part of its dynamic, attentional structure. How does the system computationally account for a co-operative performance of these tasks, e.g. with respect to the following phenomena?

- Under-informativeness (e.g. the system's exhaustiveness strategies)

  Verbmobil is a translation system which does not interact with the user during normal operation. The co-operative behaviour of a translation system fully consists in the capacity of providing a translation for almost every utterance.

- Over-informativeness (e.g. the system's refinement strategies in case that the requested task is too broad)

  Idem.

- Truthfulness of provided information (the system's information checking mechanisms)

  Idem.

- Brief and orderly presentation of requested information (the system's presentation mechanisms) If possible, indicate whether and how co-operative action initiation as part of these strategies is goal determined.

Idem.

C. Linguistic Structure

The linguistic structure of a dialogue is its segmentation structure. By definition (Grosz & Sidner 1986), the dialogue's linguistic structure is in correspondence with both the intentional structure of the dialogue and the structure of attentional states.

a *Dialogue segments*

- How many segmentation levels are distinguished?

  Because of the correspondence between intentional structure and segmentation structure, dialogue segmentation within the system crucially depends on how many intentional levels are distinguished (see 3.2 A. point d. on dialogue parsing).

  Because of this correspondence, four segmentation levels are distinguished (the levels of individual dialogues, dialogue phases, dialogue turns and dialogue acts).

b. *Speech act types*

- Which speech act types (e.g. questions, assertions, commands, etc.) can be handled by the system?[11]

  The system provides translations for all speech acts conventionally occurring in the domain of scheduling a meeting.

- How are the different types identified?

  Speech acts in the Verbmobil system function as higher-order dialogue acts, not refined in terms of functionality and propositional content. See further the point on dialogue act determination in section 3.2 A above.

- How successful is the system with respect to the interpretation of indirect speech acts, and how does this depend on the dialogue management approach chosen?

[11] What is meant here is what is captured by the theoretical notion of speech acts (e.g., Searle 1969) rather than the dialogue management notion of dialogue acts which is very often also defined in terms of propositional content.

See point on indirect dialogue acts in section 3.2 A above.

c. *(Co-)reference*

- How does the system behave with respect to different linguistic realisations of the same referential object (pronouns, anaphoric descriptions, etc.)?

  The system does account for co-referential linguistic expressions, thereby distinguishing between source and target language expressions (i.e. expressions in German and English).

- Does the system keep a history of co-referential expressions?

  Yes, in terms of constructing an incremental representation of the dialogue's referential structure.

3.3 DIALOGUE CONTEXT

A. Dialogue History

a. *Representation of Intentional Structure*

- Does the system incrementally construct a representation of the intentional structure as the result of the preceding context, in particular a representation of which part of a task has been completed?

  Yes, it incrementally constructs a representation of the intentional structure of the dialogue.

- What is the format of representation of the intentional structure (tree representation, matrix representation, etc.)?

  A tree-like structure is built up the root of which represents the top-level goal and the leaves correspond to single dialogue acts.

- Which submodule of the dialogue component is responsible for this representation (e.g. the plan recognition submodule) and where is the representation stored (e.g. in the dialogue memory)?

The Dialogue Planner gives rise to the tree-like representation of the intentional structure. This representation is stored in the Dialogue Memory (see section 2 for the diagram on the architecture of the dialogue management component of the system).

- Which other system component does the stored intentional information contribute to and how?

  The intentional information contributes to the following system components:

  - *Transfer system component*

    Knowledge about the dialogue phase to which an utterance belongs is used for determining the correct translation in case of the existence of translational alternatives (see section 1 for an example).

  - *Generation system component*

    Information about illocutionary force is needed to determine the functionally correct translation of an utterance. For instance, the utterance 'Schoen dann machen wir es so' which, without information about illocutionary force, gives rise to the following two functionally different translation possibilities: 'Okay, let's do it like that' and 'Should we do it like that?'. In this case knowledge about the current dialogue phase contributes to the determination of the correct illocutionary force.

b. *Representation of Attentional States*

- Does the system provide an incremental representation structure of the set of attentional states constituting the dynamic development of the dialogue?

  Yes.

- What is the format of representation (e.g. an updated SIL object or a list of such objects in Sundial and the Daimler-Benz dialogue system)?

  The system provides a hierarchical representation of the temporal objects in question.

- How does the incremental representation structure of attentional states

account for the set of *referentially accessible objects* at any dialogue state?

One advantage of the hierarchical representation of temporal objects is that it enables a representation of referentially accessible temporal objects.

Two other advantages of such a hierarchical representation are the following:

(i) it represents the set of open alternatives, i.e. the set of temporal objects that are still open for negotiation

(ii) it enables inheritance of temporal information:

- *Upwards inheritance*

  Upwards inheritance is induced as the result of temporal information of the type POSS. If a day is marked as possible, this also holds for the superordinating time frames (week, month, etc.) of which it is part.

- *Downwards inheritance*

  Downwards inheritance is induced as the result of temporal information of the type NEG, this in a similar way (e.g., if a week is marked as impossible, this also holds for any day of that week).

- Which other system component does the representation of attentional structure contribute to, and in what way?

Thematic structure information is used to support inferences required by both the system's Transfer and Generation components, e.g. those involving the determination of implicit acceptance or rejection of chained time frame proposals (see point on contextual inferences in section 3.2 A.). However, thematic structure is also used by the Semantic Evaluation and Transfer components to resolve anaphoric expressions, e.g. the anaphoric expression 'nächste' which can be translated as 'next' only if the date referred to is immediately after the speaking time.

c. *Representation of Linguistic Structure (Segmentation Structure)*

- Does the system represent the segmentation structure of the preceding

dialogue, and what is the relation with the incremental representation of the corresponding intentional structure?

Because of the fact that the lowest level of intentional structure consists of speech acts actually occurring in the dialogue, the hierarchical representation of the intentional structure directly gives rise to the corresponding representation of the segmentation structure of the dialogue.

- Does the system also represent referential structure (see 3.2 C., point c.)?

  Yes, see section 3.2 point C.

- Which other system component does the represented linguistic structure contribute to and how?

  For reasons which may be obvious, referential structure information is of direct importance to the Generation component.

B. User Model

a. *Modelling of performance-relevant aspects of the user*

- To what extent does the system model performance-relevant aspects of the user, such as user beliefs (his own beliefs as well as beliefs about the system), user desires (preferences), user expertise, etc.?

  Not applicable. As said, Verbmobil is a translation system which in the normal case (domain-relevant information exchange) does not act as a dialogue participant. For instance, no modelling of user preferences has been realised. However, much has been done to ensure that the system always provides translations and, in addition, that these translations are adequate.

- What is precisely the function of these aspects within the system?

  See first point.

- Does the system also allow for inferencing over initial beliefs?

  Idem.

- How do user beliefs, desires, intentions, etc. interact in the process of dialogue control?

  Idem.

b. *Contribution to co-operativity*

- What contribution is provided by the user model with respect to co-operative system behaviour?

  See first point under a. as well as section 3.2 B. point c.

- Give an indication of the remaining, most relevant problems in this respect.

  See previous point.

C. Domain Model

a. *Data*

- Give a specification of the data types determined by the task(s).

  The system has a world model, consisting of the domain, i.e. calendar knowledge. The temporal data are 'pretty realistic'.

- Provide the global characteristics of objects, properties and relations relevant to these tasks.

  Temporal objects (years, months, weeks, days, etc.) the properties and relations of which are defined by world knowledge.

b. *Rules*

- Give a global characterisation of world knowledge rules used by the system, as well as the characteristic properties of their application.

  No rules are operating on the domain data. The data are subject to all actions usually done with calendars.

D. Dialogue Settings[12]

a. *Physical setting*

- What are the characteristic physical circumstances in which the dialogues take place (face-to-face situation, communication through electronic devices, etc.), and how is their influence on co-operative communication modelled in the system?

  See human factor description of this point.

b. *Social setting*

- In what type of communicative situation do the dialogues take place (expert-novice, employer-employee, etc.), what roles do user and system have in these situations, and how is social knowledge of this type modelled in the system?

  See human factor description of this point.

[12] Although dialogue settings are directly relevant to dialogue management, in fact they are part of the human factor description.

7.8 Dialogue Management in Verbmobil VRP1

Niels Ole Bernsen and Laila Dybkjær

The Maersk Institute, Odense University
5230 Odense M, Denmark
laila@mip.ou.dk, nob@mip.ou.dk

Not verified by system developers.

1. Introduction

This paper present an analysis of dialogue management in the Verbmobil First Phase Research Prototype 1.0 demonstrator (VRP1). The analysis is presented in the form of (a) a 'grid' which describes the system's properties with particular emphasis on dialogue management and evaluation results, (b) a life-cycle model which provides a structured description of the system's development and evaluation process, and (c) supporting material, such as system architecture, example screen shots, dialogues and scenarios.

The presented information will be cross-checked with the developers of Verbmobil as well as with the complementary descriptions of other aspects of Verbmobil provided by the DISC partners. These other descriptions address speech recognition, done by LIMSI, speech generation, done by KTH, language understanding and generation, done by IMS, dialogue management, done by IMS, human factors, done by Vocalis, system integration, done by Vocalis.

Demonstrator: Available in November 1997. Phone demo planned for 10-97.

Developer: Verbmobil consortium (4 companies, 16 universities, 3 subcontractors outside Germany).

Contact: Reinhard Karger M. A., DFKI GmbH, Stuhlsatzenhausweg 3, D 66 123 Saarbrücken, Germany. Email: karger@dfki.de. URL: http://www.dfki.de/verbmobil.

Development phases:

Phase 1 (1993-1996):

1. Verbmobil Demonstrator, CeBIT 1995. 1292 words. Translates German appointment scheduling input into spoken English output.

2. Verbmobil Research Prototype 1.0 (VRP1), CeBIT 1997. 2461 words. Translates German and Japanese appointment scheduling input into spoken English output. Performs German-German clarification dialogues with users.

Phase 2 (1997-2000): Verbmobil-2 (V2). Not discussed here.

Verbmobil VRP1 Grid

The interaction model of Verbmobil, First Phase

The Verbmobil First Phase Research Prototype 1.0 demonstrator (VRP1) is a research prototype of a stand-alone appointment scheduling spoken dialogue translation support system which performs uni-directional German-to-English and Japanese-to-English translation. The Japanese-to-English part of the system will be disregarded in what follows.

URL: http://www.dfki.de/verbmobil

System performance	
Co-operativity	The issue of co-operative design of system utterances only applies to the meta-communication in German between VRP1 and the user. During phase I of Verbmobil, there were working packets that dealt with the questions of user acceptance of VERBMOBIL. An overview of the work is given in [Krause 1997]. Since Verbmobil does not interact with the user during normal operation, the main point to ensure was that it (almost) always translates an utterance. That is the co-operative behaviour users expect from a translation system: users simply don't want interactions with Verbmobil as a third dialogue partner. In the case of clarification dialogues - which is the only mode of operation where Verbmobil interrupts the dialogue of the human dialogue partners - the dialogues were based on the results of the simulations: no long clarification dialogues; allow only simple yes/no answers to system questions; if you can't immediately resolve, ask the user to repeat his/her utterance to be translated.
Initiative	Domain communication is fully natural (conversational) mixed initiative human-human communication. Meta-communication is done in German between VRP1 and user. Only the system can initiate meta-communication.
Influencing users	The system is intended for use by novice users who are advised to remain within the domain of the system. The system provides an introduction to its users. The introduction cannot be de-selected.
Real-time	The system responds in six times real-time. This is a strong limitation on the present version of the system as close-to-real-time is desirable.
Transaction success	Defined as speaker-intended contents/dialogue acts which are transferred into understandably equivalent contents in the target language. VRP1 produced +70% approximately correct translations in the domain of appointment scheduling. Evaluation was done by interpreters. +70% is insufficient for a realistic application. The notion of transaction success proper is hard to apply to VRP1 when interpreted as, e.g. the proportion of successful, fully translated real-life dialogues in the domain. Transaction success as defined for VRP1 was measured on isolated corpus sentences rather than on sentences occurring in real-life human-human dialogues. This makes the reported results non-transferable to real-life dialogues.
General evaluation	We have no information on the standards conformance of Verbmobil.

Speech input	
Nature	Continuous; spontaneous; speaker-adaptive with speaker-independent core; German, English.
Device	Close microphone. Telephone and mobile phone present additional challenges.
Phone server	N/A
Acoustic models	HMM and Neural Network based.
Search	Viterbi search and A*.
Vocabulary	2461 words. This is sufficient for the limited domain. In phase II the domain has been extended and the vocabulary size of the data collected did not exceed approx. 6000 words (except for city and street names). Word recognition accuracy 73.3% on random samples taken from previously unencountered input.
Barge-in	The system currently does not listen when it speaks.
Word hypotheses	Word hypothesis graph with probabilities.
Grammar	No grammar in the speech recogniser. Statistical language models are used.
Prosody	The prosody module recognises breaks, intonation, duration and energy of the input signal. Use of prosodic information for long-utterance segmentation, grammatical processing (speed-up of syntactical analysis, reduction of candidate interpretations), sense disambiguation, translation and dialogue management. In phase 1 the main information was boundary info, prosodic mood, and accent information. Without boundary info, the system doesn't work. The other information leads to different transfer results.
Speech output	
Device	Loudspeaker.
Language(s)	English; German paraphrases are not generated in Phase 1. What you can hear is the best chain in the word lattice from the acoustic scores.
Coded/parametric	Parametric speech. A concatenative approach is used. Hesitations etc., in input are just left out in output.
Prosody	No prosody is included in the German synthesis.
Voice character	VRP1 reproduces the voice character of the present speaker (e.g. male or female).

User utterances	
Lexicon	2461 words for German-English translation. The 2461 are for German, the English recognition list in phase 1 is approx. 900 words. The linguistic coverage in English is approx. 4000 full forms. For the coverage see above.
Grammar	German basic grammar for spontaneous speech. Grammatical coverage is sufficient. See [Bubetal97] for the evaluation.
Parsing	Applies syntactic and semantic constraints simultaneously. Combines deep and shallow semantic analysis: deep analysis through linguistically-based compositional syntactic-semantic analysis and semantics-based transfer of VITs (classes of underspecified DRSs); shallow, approximate analysis through schematic translation, dialogue act-based translation and statistical translation. Information extraction techniques are used to select the best result for semantic transfer German-to-English. The TRUG parser searches through the lattice and finds syntactically correct paths. It is pretty robust.
Style	Free in domain communication. Extremely long sentences possible. No limitations on their length. Meta-communication is avoided if possible.
System utterances	
Lexicon	Approx. 4000 words are in the generator lexicon.
Grammar	Reversible HPSG grammar for English. The grammatical coverage is sufficient.
Semantics	Domain communication: expression of the results of semantic transfer.
Style	Translation of domain utterances is reduced to expression of core messages. In case of clarification dialogues, utterances are short, the questions can be answered with yes/no.
Multimodal aspects	
None	The system is unimodal (speech-only).
Attentional state	
Focus, prior	The system maintains local focus on the current sub-task. The system maintains global focus on the sub-tasks that remain to be solved.
Sub-task id.	The system does sub-task identification by means of dialogue acts and dialogue phases. See [AlexanderssonReithinger97, ReithingerKlesen97].
Expectations	Dialogue act predictions are based on the sub-task in local focus and on the sub-tasks in global focus.
Intentional structure	
Task(s)	Appointment scheduling. Two human dialogue partners have to agree on a date to meet at a certain location. The system translates selective portions of their exchanges from German to English when requested by the users through pressing the Verbmobil button. The system only handles utterances that are strictly related to the appointment scheduling task and not, for instance, the reasons people are used to giving for their (non-) availability. This unnatural restriction constitutes a difficulty for

	application to real-life dialogues.
Task complexity	Partially structured task. Being negotiation-based, VRP1 cannot be characterised as having to fill a specific number of slots.
Communication	Domain communication: unconstrained within the limitation mentioned under 'task(s) above: VRP1 accepts very long sentences, any order of topics, any number of topics per sentence. *System-initiated meta-communication:* The system performs German-German repair and clarification dialogues with users in case of speech recognition or understanding problems. Initiation is done by modules that detect an error. Repair and clarification is being initiated if two words are phonetically similar in the recognition or in case of inconsistent time expressions like 18 o'clock in the morning. *User-initiated meta-communication:* The user has to wait for the translation, but can optionally get the best chain. If the speaker doesn't like the translation, he can repeat using a different wording.
Interaction level	Only applicable for meta-communication. None.
Dialogue structure	How represented? How implemented? Based on linguistic observations, the dialogue is structured at four levels: an entire dialogue, a dialogue phase, a dialogue turn, and individual dialogue acts. Verbmobil only performs dialogue control wrt. meta-communication. Domain communication is uncontrolled and left to the human interlocutors. Verbmobil's task is to translate the human-human dialogue contributions.
Linguistic structure	
Speech acts	The system identifies speech (or dialogue) acts in the users' input, such as 'suggest_date', which model intended utterance interpretations in abstraction from the performance phenomena of spontaneous speech**.** Dialogue acts help identify the best translation where several possibilities exist, e.g. of discourse particles such as 'ja', 'bitte' or 'vielleicht'. In shallow processing dialogue acts help select the templates used to generate target language expressions. Dialogue acts are identified through a combination of statistical and knowledge-based methods.
Discourse particles	Are partially identified and used for translation**.** Discourse particle functions distinguished include: structuring, including uptake, check, repair marking; speaker-attitude signalling; smoothing; coherence marking.
Co-reference	The system does partial co-reference resolution if needed for translation.
Ellipsis	The system does partial processing of ellipses if needed for translation.
Segmentation	The system does user turn segmentation using prosody and syntax.
Interaction history	
Linguistic	The language of the user, and indirectly the words uttered, are being recorded.
Topic	The system maintains a record of the order in which topics have been addressed during interaction. It is being used for focus determination,

	which is used for, i.a. co-reference resolution.
Task	The system maintains a record of the task-relevant information which has been exchanged. It is being used for, i.a., focus determination, which is used for reference resolution.
Performance	The system does not maintain a record of the user's performance during interaction. This is currently not needed.
Domain model	
Data	The system has a world model, consisting of the domain, i.e. calendar knowledge. The data are pretty realistic.
Rules	There are no rules, but all actions you usually do with calendars.
User model	
Goals	The user goal is assumed to be appointment scheduling, i.e. fixing time and location for meetings, without going into the user's reasons for their proposals and decisions made. This is an artificial curtailment of the user's goals in the domain.
Beliefs	Not applicable for Verbmobil. What is being said should be translated.
Preferences	Not applicable for Verbmobil. What is being said should be translated.
User group	Not applicable for Verbmobil. What is being said should be translated.
Cognition	Not applicable for Verbmobil. What is being said should be translated.
Architecture	
Platform	Standard hardware: Processor: SPARC. Memory: 500 MB. OS: UNIX (Solaris), other UNIX versions on demand; Linux version planned. Disk System: 500 MB, Swap: 2-3 Gb. The platform is adequate by today's standards.
Tools and methods	
Generic	Multi-agent, object-oriented. The generic software architecture is adequate according to today's standards. See [BubSchwinn96].
No. components	43.
Flow	See BubSchwinn96 or Bubetal97.
Processing times	On average: 38% speech recognition, 17% prosody, 25% syntax and semantics, 14% semantic evaluation and dialogue, 3% transfer, 3% generation.

Figure 1. High-level description of the Verbmobil First Phase Research Prototype 1.0.

Verbmobil dialogue(s)

See [Alexanderssonetal97].

Verbmobil screen shot(s)

See Bubetal97, BubSchwinn96 or Alexanderssonetal97.

7.9 Verbmobil VRP1
Dialogue Engineering Life-Cycle

Niels Ole Bernsen and Laila Dybkjær
The Maersk Institute, Odense University
5230 Odense M, Denmark
laila@mip.ou.dk, nob@mip.ou.dk

Not verified by system developers.

Overall design goal(s):

What is the general purpose(s) of the design process?

To give Germany research and industry world leadership in speech and language technology through the collaborative building of a system which can translate spoken appointment negotiation dialogues. Comparable only to the CMU Janus system, this is a path-breaking project.

Hardware constraints:

Were there any a priori constraints on the hardware to be used in the design process?

The project takes a pragmatic stance on hardware: to include the hardware necessary for meeting the software objectives.

Software constraints:

Were there any a priori constraints on the software to be used in the design process?

The first plan was to use keyword spotting to allow Verbmobil to follow a dialogue. However, the keyword spotter used did not work acceptably so the idea was dropped. In the November 1996 demonstrator one has to press a button before the utterance is spoken to which Verbmobil has to listen. After the utterance is spoken the button is released. In addition, Verbmobil "shadows" the dialogue and translates everything in "shallow" processing mode.

Each partner site in the project was supposed to contribute the best version(s) of the relevant software it might have at project start. The lack of common software constraints at the beginning of the project is unusual but appears justifiable from a pragmatic point of view.

Customer constraints:

Which constraints does the customer (if any) impose on the system/component? Note that customer constraints may overlap with some of the other constraints. In that case, they should only be inserted once, i.e. under one type of constraint.

No customers are involved. No hypothetical customer constraints were assumed. The lack of

even hypothetical customer constraints marks Verbmobil as a long-term research endeavour.

Other constraints:

Were there any other constraints on the design process?

The Verbmobil project is divided into two phases: Phase 1 from 1993 to 1996 and Phase 2 from 1997 to 2000. Phase 1 should, and did, deliver a convincing 1st demonstrator system. Funding comes from the German government plus approx. 60% contribution from the involved industries. Total 1st Phase funding was close to 100 mio. DM. No knowledge has been obtained on standards conformation.

Design ideas:

Did the designers have any particular design ideas which they would try to realise in the design process?

There are some things new where patents are pending.

Designer preferences:

Did the designers impose any constraints on the design which were not dictated from elsewhere?

Astonishingly few things where dictated. Basically the task was: build a working system. One thing we decided upon is the system's architecture, where we agreed on the whiteboard idea, which proved to be pretty good (flexible system, interchangeable modules,...).

Design process type:

What is the nature of the design process?

Exploratory research. A peculiarity of the design process is the large number of partners, academic as well as industrial, and the very heterogeneous nature of the initial software contributions.

Development process type:

How was the system/component developed?

The Hamburg site has developed a small WOZ corpus which was used for corpus analysis and evaluation (see Krause97).

Requirements and design specification documentation:

Is one or both of these specifications documented?

Not publicly available.

Development process representation:

Has the development process itself been explicitly represented in some way? How?

There is a project plan, technical docs about the interfaces, protocols etc.

Realism criteria:

Will the system/component meet real user needs, will it meet them better, in some sense to be explained, than known alternatives, is the system/component "just" meant for exploring

specific possibilities (explain), other (explain)?

It is not clear that the system can meet real user needs given that is covers an artificially limited domain: it does not allow users to state the reasons for their positions during appointment scheduling negotiations.

Functionality criteria:

Which functionalities should the system/component have (this entry expands the overall design goals)?

Provide for a proper translation, be robust, meet defined real-time requirements.

Customer(s):

Who is the customer for the system/component (if any)?

Verbmobil has no customer. However, industries involved in Verbmobil have produced spin-off products which do have customers.

Users:

Who are the intended users of the system/component?

Users are people who are going to meet for one reason or another and who have to agree on a date and a time for the meeting. Users are supposed to negotiate in English but to speak German or Japanese to Verbmobil. The system is for novice users.

Usability criteria:

What are the aims in terms of usability?

Adequate translation for novice users, of all possible German dialogue contributions in the domain. Also the English contributions are being processed.

Organisational aspects:

Will the system/component have to fit into some organisation or other, how?

N/A.

Evaluation criteria:

Which quantitative and qualitative performance measures should the system/component satisfy?

Provide for a proper translation, be robust, meet defined real-time requirements....

Evaluation:

At which stages during design and development was the system/component subjected to testing/evaluation? How?

Testing is done all the time (test data is exchanged between module developers in an early stage, later on you integrate other's modules and test in the Verbmobil testbed), evaluation was done on a large scale at the end of phase 1.

Coding reliability (between two coders or for the same coder at different times) is measured through (a) confusion matrices which show the dialogue acts that have been confused most frequently, and (b) kappa values. The VRP1 dialogue act coding scheme yielded a kappa value

of 0.83 for 10 pre-segmented dialogues labelled by two coders with equal experience. VRP1 had a dialogue act recognition rate between 65% and 75%. There are significant differences in how easily different dialogue act are identified. These differences are due to varying clarity of dialogue act definitions and to how easily the dialogue acts can be distinguished through their surface language expressions. It is argued that all task-relevant dialogue acts can be recognised with satisfactory accuracy. All other dialogue acts could be mapped onto a single 'garbage' dialogue act. It is not clear how this will be done or which dialogue acts will be involved. Verbmobil should ultimately use statistical training-cum-a-priori-success-rates + rule-based heuristics (weighted default rules) to identify dialogue acts.

All dialogue acts are described in terms of name, occurrence in which dialogue phases, related propositional content, definition, examples of occurrence in context in German, English and Japanese. Some of the definitions reveal the conceptual difficulties involved (see, e.g., the definition of FEEDBACK_POSITIVE).

Transaction success:

End-to-end evaluation: 20.000 turns, approximately 30.000 sentences; test sets were randomly selected and included no unknown words. The test corpus contains human-human negotiation dialogues in German. The dialogues are scenario-based. Justifications are left out on purpose since Verbmobil cannot handle justifications. This evaluation of transaction (translation) success must be characterised as early: it is corpus-based rather than based on real-time user interaction; unknown words were excluded; justifications were excluded.

Requirements and design specification evaluation:

Were the requirements and/or design specifications themselves subjected to evaluation in some way, prior to system/component implementation? If so, how?

Not known.

Development time:

When was the system developed? What was the actual development time for the system/component (estimated in person/months)? Was that more or less than planned? Why?

Let's assume 120 Persons for 4 years gives 480 person years.

Developers:

How many people took significant part in the development? Did that cause any significant problems (time delays, loss of information, other (explain))? Characterise each person who took part in terms of novice/intermediate/expert wrt. developing the system/component in question and in terms of relevant background (e.g., novice phonetician, skilled human factors specialist, intermediate electrical engineer).

I simply can't tell you details about such a large work force. We had all levels of expertise.

Mastery of the development and evaluation process:

Of which parts of the process did the team have sufficient mastery in advance? Of which parts didn't it have such mastery?

Well, we had mastery of all aspects, I think.

Problems during development and evaluation:

Were there any major problems during development and evaluation? Describe these.

Well, we did a good job in a grassroots way and came up with a good self organisation. There were of some edges, but we worked it out.

Maintenance:

How easy is the system to maintain, cost estimates, etc.

No estimates.

Portability:

How easily can the system/component be ported?

We are in the process to port to LINUX, and first modules work. As long as a reasonable OS (POSIX compliant, gcc, lisp, prolog running) is used, there should be no major problems.

Modifications:

What is required if the system is to be modified?

Not known.

Robustness:

How robust is the system/component? How has this been measured? What has been done to ensure robustness?

Robustness is enhanced by parallel tracks. See Bubetal97.

Platform and architecture:

On which OS(s) will the system/component run? Machine requirements? Was a particular development platform used? Description of the architecture of the system/component.

Sun Ultra 2; Solaris; 1 GB memory; 8 GB internal and external disk space; analogue-to-digital converter (October 1996 demonstrator). See Bubetal97, BubSchwinn96.

Component selection/design:

Describe the components and their origins.

The Verbmobil system is a hybrid system which means that it takes from different modules what they produce and concatenates the results. The Phase II Verbmobil system is expected to use a more intelligent selection.

Programming languages: Several programming languages were used, including FORTRAN, C and C++.

Speech recognition: Two sites have developed recognisers: D-B and Karlsruhe. Karlsruhe both have a German and a Japanese recogniser. The D-B recogniser was used in the November 1996 demonstrator.

Speech recognition: No prosody is included in the German synthesis. True-talk is being used for English synthesis.

Syntax and semantics: Several sites have developed syntax-semantics modules: IBM Heidelberg, Siemens AG Munich, Univ. des Saarlandes, Saarbrücken.

Dialogue management: Dialogue management was developed at DFKI Saarbrücken.

Inter-process communication: For inter-process communication ICE was used. ICE was developed by Verbmobil as an add-on to PVM which in itself was not powerful enough. These are the basic communication mechanisms. PVM is the well-known parallel virtual machine, ICE is the Intarc Communication Environment, which provides an easy to use interface-layer on top of PVM. See BubSchwinn96, Bubetal97, Alexanderssonetal97.

Development and evaluation process sketch:

Please summarise in a couple of pages key points of development and evaluation of the system/component. To be done by the developers.

Management of Verbmobil is centralised (DFKI Saarbrücken). System integration is done at DFKI Kaiserslautern. See BubSchwinn96, Bubetal97. For the dialogue module see [Alexanderssonetal97].

Property rights:

Seen from outside software belongs to the group which developed it. However, it is part of the contract that the involved industries have the right to use university partners' software and to obtain the source code. Universities may get source code from other universities but not from industry. For example, DFKI get binaries from D-B for system integration.

References

[Alexanderssonetal97] Jan Alexandersson, Norbert Reithinger and Elisabeth Maier: Insights into the Dialogue Processing of Verbmobil. Proceedings of the Fifth Conference on Applied Natural Language Processing, ANLP '97, Washington, DC 1997, 33-40.

[AlexanderssonReithinger97] Jan Alexandersson and Norbert Reithinger: Learning Dialogue Structures from a Corpus. Proceedings of EuroSpeech-97, Rhodes 1997, 2231-2235.

[BubSchwinn96] Thomas Bub, Johannes Schwinn. Verbmobil: The Evolution of a Complex Large Speech-to-Speech Translation System. DFKI GmbH Kaiserslautern. Proceedings of ICSLP'96, Philadelphia 1997, 2371--2374.

[Bubetal97] Thomas Bub, Wolfgang Wahlster and Alex Waibel: VERBMOBIL: The Combination of Deep and Shallow Processing for Spontaneous Speech Translation. Proceedings of ICASSP-97. Munich 1997.

[Krause 1997] Detlev Krause: Using an Interpretation System - Some Observations in Hidden Operator Simulations of VERBMOBIL. In Elisabeth Maier, Marion Mast and Susan LuperFoy (Eds.): Dialogue Processing in Spoken Language Systems. Lecture Notes in Artificial Intelligence, Springer Verlag, 1997.

[ReithingerKlesen97] Norbert Reithinger and Martin Klesen: Dialogue Act Classification Using Language Models. Proceedings of EuroSpeech-97, Rhodes 1997, 2235--2238.

7.10 THE WAXHOLM SYSTEM

DISC
DIALOGUE COMPONENT GRID ANALYSIS

JAN VAN KUPPEVELT AND ULRICH HEID

University of Stuttgart
Institut für maschinelle Sprachverarbeitung
– Computerlinguistik –
Azenbergstrasse 12
D-70174 Stuttgart
(Jan.v.Kuppevelt@ims.uni-stuttgart.de; Uli@ims.uni-stuttgart.de)

Second version. Analysis is based on the questionnaire "DISC Dialogue Component Grid Questions - A Proposal" (3rd version, April 1998).

Not verified by system developers.

April 1998

CONTENTS

a. Modelling of attentional state
   (i) Local focus
   (ii) Global focus
b. Additional expectations
c. Co-operativity

C. Linguistic Structure

a Dialogue segments
b. Speech act types
c. (Co-)reference

3.3 DIALOGUE CONTEXT

A. Dialogue History

a. Representation of Intentional Structure
b. Representation of Attentional States
c. Representation of Linguistic Structure (Segmentation Structure)

B. User Model

a. Modelling of performance-relevant aspects of the user
b. Contribution to co-operativity

C. Domain Model

a. Data
b. Rules

D. Dialogue Settings

a. Physical setting
b. Social setting

1. Dialogue Component Tasks

- What are the main tasks of the Dialogue Component?

  1) Dialogue control for task-oriented domain communication

     The main task of the dialogue component is dialogue control for task-oriented domain communication. Controlling the course of the dialogue with the user is realised by complementing the user's requests by system questions to achieve the user's goals.

  2) Dialogue control for meta-communication

     The Waxholm system conducts meta-dialogues for repairs (system-initiated repairs in case of no understanding and out of domain understanding), corrections (user-initiated corrections on system utterances) and repetitions (system and user-initiated requests for repetitions).

  3) Constraining the search space of other system components

     In agreement with the output characteristics of the dialogue component (a bi-directional information flow is only defined for the interaction between the Dialogue Manager and the Database Search component), no contextual information is used to constrain the search space of other system components such as the (Speech) Recognition component and the Grammar & Semantics component (in the Waxholm system syntactic and semantic analysis of user utterances are combined into one system component; no syntactic and semantic component exists for outgoing system utterances).

  4) Other contributions to system components

     Apart from its main task of dialogue control, the dialogue component of the Waxholm system also has a generation task. In Waxholm the generation of system utterances is part of the dialogue component.

- Does the dialogue component only have a mediating function, as is usually the case with translation systems in which the system does not really act as a dialogue participant?

  In the dialogues on boat traffic information the Waxholm system always functions as a real dialogue participant. No mediating functions are carried out.

- To which main system requirements (robustness, efficiency, etc.) of the overall system do the tasks of the dialogue component contribute?

  Among the system requirements are efficient (fast and accurate) and robust system performance. However, both efficiency and robustness are not realised with the help of contextual information in any substantial and systematic way. Efficient system performance is not realised with the help of contextual information constraining, e.g., the decisions of other system components (see first point above). As for robustness, the occurrence of 'unknown' topics, e.g., is not accounted for contextually (e.g. by identifying the new topic as inconsistent with what can be expected on the basis of the given input) but rather lexically by spotting on lexical semantic features of the topic-bearing utterance for an out of domain topic. An exception, however, form those cases in which an incomplete user sentence is given as an answer to a system question.

2. Dialogue Component Architecture

a. *Architecture specification*

- What are the submodules of the dialogue management component of the system?

  Although no explicit modularity has been defined for the dialogue management component, it contains a dialogue grammar and makes use of a probabilistic tool for topic identification (a statistical topic guesser).

- What are their functions?

  The topic identification tool is used for identifying main topics comprising main system tasks (e.g. the task of getting presented an adequate time-table); the dialogue grammar is used for parsing and controlling the course of the dialogue within a topic unit.

- How do they interact?

  As said, no explicit modularity has been defined for the Waxholm system.

b. *Generic architecture*

- Does the dialogue management component of the system have a generic architecture? Give an answer to this question insofar it is relevant to the grid analysis, thereby also indicating the *level* of genericity (language-independent architecture, domain-independent architecture, etc.).

Both the dialogue grammar and topic identification part of the dialogue component are domain-dependent. The dialogue grammar is a domain-dependent subgrammar only covering the corresponding data in the domain, while the topic-identification part is based on domain-dependent lexical semantic features (BOAT, PORT, etc.) assigned to the nodes in the grammatical analysis of the topic-introducing sentence.

c. *Interaction with other system components*

- Specify the input/output characteristics of the dialogue component.

Input characteristics

The Dialogue Management component only receives a direct input from the Grammar & Semantics component. Central to this input is a semantic frame resulting from semantic analysis. The latter is a straightforward process implying, among other things, a reduction of the syntactic tree into a semantic one by deleting all nodes and branches that contain no semantic information. The slots in the frame correspond to attribute-value information taken from the tree.

Output characteristics

The output of the Dialogue Management component is input for the Graphics component, the Speech & Face Synthesis component and the Sound component. Depending on positive constraints evaluation the dialogue management component gives rise to selected actions to take place, e.g. the synthesis of a (graphicalized) time-table message.

3. Dialogue Management

3.1 THE DIALOGUE MANAGEMENT APPROACH

a. *Kind of approach*

- *The underlying theoretical model*

  Which theoretical approach or approaches to modelling dialogue (structure) underlie the implementation of the dialogue management component, this according to the descriptions given below as far as the developers see themselves bound to a given approach?

Two approaches to modelling dialogue underlie the implementation of the dialogue component of the Waxholm system, namely a topical approach and a dialogue grammar approach. Although a goal is associated with every topic unit, dialogue structure is not described in terms of a plan-based or collaborative approach which provide a description primarily in terms of speakers' intentions or joint intentions respectively. The goals associated with the topic units have no function in the process of dialogue control. However, in the Waxholm system the two approaches apply to different structural levels implying, among other things, that they cannot be complementary in performing the task of dialogue control on each of these levels.

- *Computational tools used*

  Which computational tools are used for implementation, possibly also making use of tools developed within the framework of other theoretical approaches? Give a specification of the tools used in relation to the adopted approach, e.g. dialogue grammar techniques, plan-based techniques (non-dynamic or incremental plan recognition/formation techniques, multi-level planning/plan recognition techniques accounting for both domain and discourse plans), word spotting techniques for identifying topic units, dialogue acts, etc.

  The two underlying approaches which are not uniformly applied to all structural levels have given rise to a corresponding asymmetry in processing methods.

  - *The topical approach underlying the Waxholm system is directly based on a topic guesser (see 3.1 point b.).*

  - *The dialogue grammar which is a domain-dependent subgrammar is implemented in STINA. STINA is based on the MIT parser TINA (in particular, Seneff 1992). STINA is Swedish TINA. STINA is used both as a sentence and dialogue parser. It runs with two different time scales, one corresponding to the words in each utterance and one to the turns in the dialogue. In Waxholm the dialogue grammar which in this system is not used as a top-down prediction mechanism is presented into a graphical form.*

b. *Use of statistical identification and prediction tools*

- Does the dialogue management component comprise a special submodule containing a statistical implementation of the dialogue model besides a knowledge-intensive submodule containing, e.g., a plan-based implementation? (Note that the latter one itself may also make use of statistical methods.)

No.

- Give a specification of all dialogue functions for which statistical methods are used, making explicit whether the method applies to all structural levels in the dialogue or just to a specific kind.

  - *Topic identification by a statistical topic guesser*

    In the Waxholm system the topic guesser is a stochastic one making use of statistical knowledge for identifying topic units which are considered to belong to the highest structural level in the dialogue. Topic identification is based on several factors such as dialogue history and utterance content, the latter of which is only expressed by means of lexical semantic features assigned to its syntactic structure. The probability for each topic is expressed as p(topic|F), where F is a feature vector including all semantic features used in the utterance (according to this feature vector the BOAT feature, e.g., is a strong indication for the TIME_TABLE topic). Probabilities in the feature vector are obtained from labelled utterances in the Waxholm database. Only utterances indicating a topic have been included.

  - *Topic-internal dialogue control by a probabilistic ATN compilation of a dialogue grammar*

    The dialogue grammar which controls the topic-internal dialogue flow is compiled into an ATN (Augmented Transition Network). Like the syntactic nodes on the sentence level, the dialogue nodes representing a dialogue state are processed according to ATNs with probabilities on each arc.

3.2 DIALOGUE STRUCTURE

A. Intentional Structure

Given the task-oriented character of the dialogue systems under consideration, we take the intentional structure of a dialogue as the mostly hierarchical structure of common goals ('discourse segment purposes') to perform a (sub)task.

a. *Task characteristics*

(i) *Multiple tasks*

Does the system perform more than one overall task?

Yes. Each of these tasks consists in providing specific information about boat traffic in the Stockholm archipelago.

(ii) *Task specification*

Which (overall) tasks are performed by the system (e.g. making a flight ticket reservation in the Danish Dialogue System, showing a time-table in Waxholm and fixing a meeting date in Verbmobil)?

The main tasks performed by the system are providing information on time-tables, port locations, hotels, camping places and restaurants.

(iii) *Task structure*

Are all tasks well-structured, i.e. do all tasks have a stereotypical structure explicating both the amount and order of information updates needed to complete the task (see, in particular, Bernsen, Dybkjaer & Dybkjaer 1998 for more information on this point)?

The tasks are not well-structured in the sense that the semantic frame resulting from the semantic analysis of a topic-introducing utterance leaves undecided whether all slots have to be filled, as well as the order in which slot filling can be realised.

b. *Communication types*

(i) *Domain communication*

Specify the relevant characteristics of the task-oriented domain communication, in particular whether and/or how the system accounts for the following phenomena (some of which in fact may give rise to meta-communication):

- System directed, user directed or mixed initiative communication

  Domain communication is realised by mixed initiative. The system allows user initiatives without any instructions to the user.

- Free or bound order of main tasks

  The underlying semantic frame does not account for the order in which the main tasks can be realised.

- Discontinuous user input ('input gaps')[13]

  Discontinuous user input can in principle be handled by the system, because, as with topic identification, the system only considers the semantic features of the current input rather than characteristics of the immediate preceding system input. Many other systems identify new topics on the basis of the immediate preceding context.

- Contextual inferences of different types[14]

  In Waxholm contextual inferences are not dealt with, except a limited type of temporal inference (see 3.3 C., point b.).

- Indirect speech acts[15]

  Indirect speech acts are also not dealt with.

- Incomplete, underspecified user requests

  The user's requests may be complemented by system questions to achieve his/her goals (e.g. system questions induced to get visualised just a relevant part of a time-table). However, user utterances of the form 'I want to go to an island north of x (name island) on the day after Friday' give rise to misunderstanding in Waxholm.

- Incomplete answers to system questions

  [Unknown]

- Overcomplete answers to system questions

  [Unknown]

---

[13] Discontinuous user input involving large gaps between given and new system input may be very problematic in case that system performance is dependent on predictions derived on the basis of just recent system input.

[14] Different types of contextual inferences may require a varity of inference rules, e.g. those used in Verbmobil to account for the fact that a counterproposal (for a date, etc.) implies the rejection of a previous proposal, that a new proposal (for a date) implies the acceptance of a (not incompatible, less specific) previous proposal and that a change of dialogue phase implies the acceptance of a previous proposal (for a date).

[15] A problem for most spoken language dialogue systems is the determination of indirect speech (dialogue) acts. Because indirect speech acts cannot be detected on the basis of surface cues, keyword spotting is useless in these cases.

- Ambiguous user utterances

  No disambiguity operations are performed by the system.

- Other

  Examples of other domain communication not handled by the Waxholm system are complex sentences, multiple questions and conjunctions of questions.

  Central to Waxholm is the fact that the domain communication on a topic is mainly presented in a graphical form.

(ii) *Meta-communication*

System-initiated meta-communication

Does the system initiate meta-communication, and in what way?

Yes, see below.

User-initiated meta-communication

Does the system allow for user-initiated meta-communication? Make sufficiently explicit in what way.

Yes, see below.

Types of meta-communication

How does the system behave with respect to different types of meta-communication, such as:

- Repairs (repair types, repair success, etc.)

  System-initiated repairs in case of no understanding and out of domain understanding of user input can be handled by the system.

- Clarifications (location-restricted/-unrestricted, system-/user-initiated, etc.)

  Clarifications are not dealt with.

- Explanations

  Explanations are not dealt with too.

- Repetitions

  Both system- and user-initiated requests for repetitions are handled by the system.

- Reasons ('nucleus reasons' vs. 'satellite reasons')[16]

  Reasons are not dealt with.

- Deliberations (deliberation types)
  Deliberations are also not dealt with.

- Corrections (correction types)

  User-initiated corrections on system utterances can be dealt with.

- Other

  [Unknown]

(iii) *Other communication types*

How does the system behave wrt. other communication types, such as:

- Dialogue introduction phases (domain-independent communication), e.g. greetings and introduction of dialogue participants

  Like other systems, Waxholm makes use of a dialogue introduction phase.

- Dialogue closure phases (domain-independent communication), e.g. final confirmation, thanks and saying goodbye

[16] As opposed to satellite reasons, reasons occurring in a nucleus dialogue segment must be considered to belong to domain communication because of the property of providing an intentional value.

Like other systems, Waxholm also makes use of a closure phase.

- Side structure information induced by digressions (related to, but out of the domain communication)

System vocabulary is extended with entries the lexical semantic features of which are used for identifying out of domain topics.

c. *Modelling of intentional structure*

How is the dialogue's intentional structure modelled in terms of the chosen dialogue management approach, e.g. a dialogue grammar, topical or plan-based approach (see section 3.1 above)? Note that although task-oriented dialogues have an underlying goal structure, these dialogues are not always modelled in terms of the goals to perform these tasks.

(i) *Uniform versus partial application*

Do the chosen theoretical approaches of modelling dialogue underlie all intentional levels of the dialogue, or just a specific part of this structure?

The Waxholm system does not apply its two approaches to all levels. The topical approach is only applied to higher intentional structure (using statistical information, the higher level task communication is parsed into a sequence of topic units). The dialogue grammar, on the other hand, (which, as said, is compiled into an ATN) accounts for the topic-internal structure. Each predicted dialogue topic is explored according to the dialogue grammar rules. These rules which are presented in the Dialogue Node Specifications are responsible for the specific set of topic-internal dialogue actions. Dialogue grammar rules define which conditions have to be fulfilled to perform a specific action depending on the dialogue history. As said, the dialogue grammar is presented by a graphical interface.

(ii) *Intentional levels*

How many intentional levels are distinguished in the dialogue component of the system?

a. *Intentional continuum*

- Is every intentional unit represented separately in terms of the chosen approach or is the intentional continuum split up in just some main intentional levels?

Neither higher-order dialogue structure accounted for in terms

of topics, nor topic-internal dialogue structure defined in terms of dialogue actions show an intentional continuum implying the adoption of a separate level for every sub(meta-)task that is communicated.

- Does the system define a hierarchical order for the intentional levels it distinguishes?

  Not applicable.

b. *Representation of larger intentional units*

If only larger intentional units are distinguished by the system, which are these and are they hierarchically structured?

No explicit distinction between intentional levels is made in Waxholm, except the one between the level consisting of topic units and that comprising topic-internal structure. Although units belonging to the former level dominate those of the latter, no internal hierarchy is defined for the set of topics defining higher-order dialogue structure. As for topic-internal dialogue structure, on the other hand, dialogue rules just account for the sequential order of the topic-internal dialogue actions.

Questions wrt. the assumption of larger intentional units are the following:

- How is the subdivision in intentional units motivated?

  Not motivated in the Waxholm documentation.

- What motivation is given for the assumed basic dialogue units, specific classes of which constitute the higher-order levels?

  [Unknown]

- How are basic dialogue units determined?

  The Waxholm system does not identify the dialogue actions of the user.

- What is said about the relation between dialogue phases and the classes of basic dialogue units constituting these phases?

No explicit information is available on this point.

- Which typology is defined for basic dialogue units? Indicate whether the set of basic dialogue units is an open set, whether the proposed typology contains domain-dependent basic units, etc.

  No typology is defined for dialogue actions.

d. *Dialogue parsing*

Give a description of the dialogue parsing process, accounting for the following points:

- If not already answered (section 3.1), what type of dialogue parser has been chosen (a task-dependent parser, a topic-dependent parser, etc.)?

  The Waxholm system which also makes use of the STINA parser for dialogue structure contains a context-free grammar.

- What kind of parsing strategy has been followed (top-down or bottom-up strategy, complete or partial parsing strategies, etc.)?

  The parsing strategy is bottom-up and complete.

- Give a brief description of the parsing process, including a specification of parsing constraints (syntactic constraints, semantic constraints, prosodic constraints, contextual constraints, etc.).

  In contrast to the sentence parsing process which is described in detail, the Waxholm documentation does not account for the dialogue parsing process.

- How does the dialogue parser account for the isomorphism between intentional and linguistic structure, in particular does parsing automatically give rise to a representation of the linguistic structure?

  [Unknown]

- How does the dialogue parser account for updates of the corresponding attentional states, either directly or indirectly?

[Unknown]

B. Attentional state

The term attentional state is used to refer to the *current dialogue information state* and comprises the current subtasks which are not yet closed off, i.e. both the most recent (sub)task currently dealt with and the set of superimposing (sub)tasks to which this subtask contributes. The attentional state is dynamic, representing the (remaining) set of objects, properties or relations in *focus of attention* at each point in the development of the discourse. Following Sidner (1979) and Grosz & Sidner (1986) a distinction can be made between *local focus* ('focus prior' or 'immediate focus') and *global focus*, referring to the distinction between what is brought into focus of attention by the current subtask addressed by, e.g., the latest question and that which, at the same time, is still in focus of attention as the result of previous superimposing subtasks.

a. *Modelling of attentional state*

(i) *Local focus*

- How does the system model local focus?

  In the Waxholm system local focus is that which is in focus of attention at the current stage arrived at in the topic-internal graph structure. Although this is not made explicit by the developers, local focus is the contextually provided set of alternative realisations of the lower-order task associated with the current point in the graph structure. From this set of options one is chosen as the result of which a further stage in the graph is reached. Every further stage in the graph structure can be seen as bringing the dialogue participants closer to the Realization of the higher-order task associated with the current topic.

- In what specific way does the modelling of local focus give rise to expectations for the next user input?

  The model does not give rise to expectations derivable from local focus.

- How do focus values selected by answers to local questions give rise to updates on the global level?

  Each step in the development of the dialogue not only results in reaching a further stage in the graph but also gives rise to an updated

semantic frame on the global level (in fact, each step gives rise to a new semantic frame with reference pointers to the dialogue history).

- How is local focus determined in case of other speech acts than questions, and how is it determined in case of extended answers to questions?

  [Unknown]

- Of what types of linguistic data is made use to identify local focus (specific prosodic information, information about contextually determined word order, etc.)?

  No linguistic data are used to identify local focus.

(ii) *Global focus*

- Does the system model global focus? If so, how does it model global focus in relation to local focus?

  Without stating it, global focus is modelled in terms of the semantic frame associated with the current topic (e.g. a time-table topic, a show-map topic, an out of domain topic). This frame is updated as the result of updates on the local level, the latter bringing the dialogue to a further stage in the graph structure related to this topic.

- If the system makes use of expectations induced by global focus, what is their nature and impact?

  The system does not make use of expectations induced by global focus in an explicit way. However, dialogue grammar rules presented in the dialogue node specifications define which constraints apply to the performance of a new dialogue action.

- Does the system make use of linguistic markers of global focus phenomena, e.g. topic shift markers which are relatively easy to implement?

  No linguistic data are used to identify global focus, including no topic shift markers.

b. *Additional expectations*

- What other (statistical or other) methods are implemented to compute, in particular, the immediate local expectations, e.g. those which arise from a current system/user question?

  No other methods are implemented to compute local expectations.

c. *Co-operativity*[17]

Task Realization in dialogue is considered to be part of its dynamic, attentional structure. How does the system computationally account for a co-operative performance of these tasks, e.g. with respect to the following phenomena?

- Underinformativeness (e.g. the system's exhaustiveness strategies)

  In case of underinformativeness the semantic frame needs to be expanded with additional information. In order to avoid underinformativeness, a system question is synthesised to the user.

- Overinformativeness (e.g. the system's refinement strategies in case that the requested task is too broad)

  Also in case of expected overinformativeness, a system question is generated. For instance, in case of a time-table topic a system question may be generated in order to get presented only a relevant part of the time-table.

- Truthfulness of provided information (the system's information checking mechanisms)

  No information checking mechanisms are known.

- Brief and orderly presentation of requested information (the system's presentation mechanisms)

  The graphical interface enables an efficient presentation of the requested information, including listings of what has been secured thus far.

If possible, indicate whether and how co-operative action initiation as part of these strategies is goal determined.

[17] Although co-operative dialogue performance can be handled within *human factors*, it is inherently related to the dynamics of dialogue expressed by its structure of attentional states.

No special attention has been paid to the co-operative Realization of tasks. No relation exists between the system's behaviour in this respect and the goals associated with topics. In case of non-co-operative performance repair and other mechanisms of the type discussed before are applied.

C. Linguistic Structure

The linguistic structure of a dialogue is its segmentation structure. By definition (Grosz & Sidner 1986), the dialogue's linguistic structure is in correspondence with both the intentional structure of the dialogue and the structure of attentional states.

a *Dialogue segments*

- How many segmentation levels are distinguished?

  Because of the correspondence between intentional structure and segmentation structure, dialogue segmentation within the system crucially depends on how many intentional levels are distinguished (see 3.2 A. point d. on dialogue parsing).

  The segmentation of the dialogue is in agreement with the two intentional levels distinguished, namely the higher-order level which is segmented into a sequence of topic units and the lower topic-internal level which is segmented into (turns and) dialogue actions.

b. *Speech act types*

- Which speech act types (e.g. questions, assertions, commands, etc.) can be handled by the system?[18]

  The system processes both questions and assertions, the latter functioning as (partial) answers. However, unspecified is which other speech acts can be handled by the system.
- How are the different types identified?

  [Unknown]

- How successful is the system with respect to the interpretation of indirect speech acts (see also above), and how does this depend on the dialogue management approach chosen?

[18] What is meant here is what is captured by the theoretical notion of speech acts (e.g., Searle 1969) rather than the dialogue management notion of dialogue acts which is very often also defined in terms of propositional content.

As said before, the system does not handle indirect speech acts.

c. *(Co-)reference*

- How does the system behave with respect to different linguistic realisations of the same referential object (pronouns, anaphoric descriptions, etc.)?

  Co-reference resolution is not accounted for by the system.

- Does the system keep a history of co-referential expressions?

  No.

3.3 DIALOGUE CONTEXT

A. Dialogue History

a. *Representation of Intentional Structure*

- Does the system incrementally construct a representation of the intentional structure as the result of the preceding context, in particular a representation of which part of a task has been completed?

  Only an incremental representation of the (sub)tasks currently performed is presented by the system, namely in the form of a graphical overview of topic information the user has received so far (e.g. listings of hotels).

- What is the format of representation of the intentional structure (tree representation, matrix representation, etc.)?

  Intentional (topic) information is presented into a graphical form.

- Which submodule of the dialogue component is responsible for this representation (e.g. the plan recognition submodule) and where is the representation stored (e.g. in the dialogue memory)?

  Dialogue node specifications provide specifications according to node activities, specifying which selected actions can take place, e.g. the graphical display of a time-table.

- Which other system component does the stored intentional information contribute to and how?

  Intentional (topic) information presented in a graphical form does not contribute to any of the other system components.

b. *Representation of Attentional States*

- Does the system provide an incremental representation structure of the set of attentional states constituting the dynamic development of the dialogue?

  The structure of attentional states on the local level is expressed by the path through the graph structure that is associated with the current topic, while the attentional structure on the global level (global focus) is represented in terms of updates of the semantic frame derived from the combined syntactic-semantic analysis of the topic-introducing utterance.

- What is the format of representation (e.g. an updated SIL object or a list of such objects in Sundial and the Daimler-Benz dialogue system)?

  Local attentional structure is represented syntactically in terms of a graphical form, while that on the global level is represented semantically in terms of updates of a semantic frame.

- How does the incremental representation structure of attentional states account for the set of *referentially accessible objects* at any dialogue state?

  Co-reference resolution is not accounted for.

- Which other system component does the representation of attentional structure contribute to, and in what way?

  Attentional structure information does not contribute to any of the other system components.

c. *Representation of Linguistic Structure (Segmentation Structure)*

- Does the system represent the segmentation structure of the preceding dialogue, and what is the relation with the incremental representation of the

corresponding intentional structure?

The system does not maintain a representation of the segmentation structure of the dialogue, except than in terms of log files.

- Does the system also represent referential structure (see 3.2 C., point c.)?

  No.

- Which other system component does the represented linguistic structure contribute to and how?

  No segmentation structure (other than in terms of a log file) is represented by the system.

B. User Model

a. *Modelling of performance-relevant aspects of the user*

- To what extent does the system model performance-relevant aspects of the user, such as user beliefs (his own beliefs as well as beliefs about the system), user desires (preferences), user expertise, etc.?

  The system does not model performance-relevant aspects of the user.

- What is precisely the function of these aspects within the system?

  See first point.

- Does the system also allow for inferencing over initial beliefs?

  See first point.

- How do user beliefs, desires, intentions, etc. interact in the process of dialogue control?

  Although the system does not make use of user beliefs and user desires, it accounts for user's goals (e.g. to get a time-table presented, to get a map presented, to get a display of the available lodging and dining possibilities, etc.). User's goals are associated with topics introduced by user's

utterances. However, their functional status is nihil considering the fact that they have no function in dialogue control.

b. *Contribution to co-operativity*

- What contribution is provided by the user model with respect to co-operative system behaviour?

  No contribution in this respect.

- Give an indication of the remaining, most relevant problems in this respect.

  See first point.

C. Domain Model

a. *Data*

- Give a specification of the data types determined by the task(s).

  The system's database contains information on boat traffic in the Stockholm archipelago. Time-tables form the main part of the system's database. The database also contains information on port locations, hotels, camping places and restaurants. All information is accessed by the Standardised Query Language (SQL) of the Database Search component.

- Provide the global characteristics of objects, properties and relations relevant to these tasks.

  Not applicable.

b. *Rules*

- Give a global characterisation of world knowledge rules used by the system, as well as the characteristic properties of their application.

  Only limited inferencing over temporal objects is dealt with by the system, e.g. the inferencing required for interpreting phrases as 'next boat', 'next Friday', 'with Christmas', etc. The system cannot cope with phrases of the form 'a port north of island x', 'a port near island x', etc.

D. Dialogue Settings[19]

a. *Physical setting*

- What are the characteristic physical circumstances in which the dialogues take place (face-to-face situation, communication through electronic devices, etc.), and how is their influence on co-operative communication modelled in the system?

  See human factor description of this point.

b. *Social setting*

- In what type of communicative situation do the dialogues take place (expert-novice, employer-employee, etc.), what roles do user and system have in these situations, and how is social knowledge of this type modelled in the system?

  See human factor description of this point.

[19] Although dialogue settings are directly relevant to dialogue management, in fact they are part of the human factor description.

7.11 Waxholm Dialogue Manager Grid

Laila Dybkjær and Niels Ole Bernsen

The Maersk Institute, Odense University
5230 Odense M, Denmark
laila@mip.ou.dk, nob@mip.ou.dk

1. Introduction

This paper present an analysis of dialogue management in the Waxholm system. The analysis is presented in the form of (a) a `grid' which describes the system's properties with particular emphasis on dialogue management and evaluation results, (b) a life-cycle model which provides a structured description of the system's development and evaluation process, and (c) supporting material, such as system architecture, example screen shots, dialogues and scenarios.

The presented information will be cross-checked with the developers of Waxholm as well as with the complementary descriptions of other aspects of Waxholm provided by the DISC partners. These other descriptions address language understanding, done by LIMSI, speech generation, done by LIMSI, dialogue management, done by IMS, human factors, done by Vocalis, system integration, done by Vocalis.

Demonstrator: Available in Stockholm

Developer: KTH

Contact: Rolf Carlson, Inger Karlsson

System/component identifier	
The Waxholm system is a research prototype of a multimodal interactive speech system for information on boat traffic in the Stockholm archipelago. URL: http://www.speech.kth.se/waxholm/waxholm.html	
System performance	
Cooperativity	The system is intended to be co-operative. When problems were observed in user-system interaction an effort was made to repair the problems. However, cooperativity has not been a focal point and the design of system utterances has not been analysed and evaluated in any detail.
Initiative	Domain communication: User initiative complemented by system questions when necessary. User initiative works as long as user utterances address the domain covered by the system and as long as the user does not

	use a complex sentence structure. Conditional sentences, for instance, are not understood. The overall initiative distribution seems appropriate for the task(s). However, no measurements or evaluations of user and system initiative have been done except for counting number of utterances with user and system initiative, respectively. Meta-communication: Both the system and the user may initiate meta-communication. The system may inform the user that it did not understand what was said or it may inform the user that the topic the user is talking about is outside the domain handled by the system. Users may initiate repair, e.g. by saying `no, I said Waxholm, not Stockholm'. Users may ask for repetition. Clarification, for instance of ambiguities, is not being handled by the system. The meta-communication functionality has not been evaluated in any systematic way.
Influencing users	Walk-up-and-use system. Users receive a text introduction to the system on the screen, which is also presented in synthetic speech (slow). It is possible to by-pass the spoken introduction (and thereby also move away from the written introduction) by pressing a button.
Real-time	According to the developers the system should be able to respond in real time. During the demonstration it was slow, apparently due to an overloaded network. However, database look-ups seem to take quite some time. The recogniser runs in real-time with a delay of perhaps one second from time to time. Graphics takes time and so does synthesis. To make the user aware that something is happening the talking face will say "hmmm" and then "I'm looking for ...". No exact measurements on the individual system components have been made.
Transaction success	Transaction success was defined as whether the user did get what s/he asked for. Some scenarios were designed so that they could not be solved. In these cases dialogues would count as transaction successes if the user found an optimal solution taking into account that the task in itself was not solvable. An example could be the task of obtaining information on how to get to an island that has no ferry connection. In that case an optimal solution would be to get information on how to go to the nearest island. Users were encouraged to continue interaction with the system after they had carried out their scenarios. These parts of the dialogues were not counted as transaction successes/failures. In a

	number of cases it was too difficult to tell if the user got what s/he wanted. No analysis was made to decide whether this was due to a flawed definition of `transaction success' or to a conceptual problem with the concept of `transaction success' itself.
General evaluation	No ISO standards other general methods used.
Speech input	
Nature	Continuous; spontaneous; speaker-independent; Swedish. The nature of the speech input is appropriate to the task(s).
Device(s)	Microphone. A high-quality microphone was used during the WOZ experiments. For the demonstration a medium-quality microphone was used. Deployment appropriateness cannot be evaluated.
Phone server	No
Acoustic models	Based on neural networks (one output for each of the 40 Swedish phonemes used in the lexicon).
Search	A* N-best search. Viterbi search starts when the user starts talking.
Vocabulary	Approx. 1000 words. All words are active at any one time.
Barge-in	The system does not listen when it speaks. No measurement was made of the potential problems caused by that. However, barge-in by pressing a button to stop the synthetic speech is allowed at any time during interaction.
Word hypotheses	The recogniser typically produces an N-best list of 10 hypotheses. Recogniser score values are not used in the parsing.
Grammar	Simple bigram language model.
Prosody	The system does not process input prosody.
Speech output	
Device(s)	Loudspeaker.
Language(s)	Swedish.
Input	Text strings.
Lexicon	-
Sound generation technique	Parametric speech; formant-based. Quality has not been measured in the Waxholm project. Different voice characters (e.g. male/female) are not being used.
Prosody	Prosodic modelling. Grapheme to phoneme rules used in synthesis.
Pronunciation description units	-

Flexibility	-
Miscellaneous	-
User utterances	
Lexicon	Approx. 1000 words in the top-level lexicon. The parser uses a sub-lexicon. Based on one word in the top lexicon there may be, e.g., six different meanings of this word in the sub-lexicon. Lexical coverage has not been measured (see also on vocabulary above). Each lexical entry can have domain specific semantic features associated with it in addition to the basic syntactic features. Semantic features are of two types: basic features (e.g. BOAT, PORT) and function features which are typically associated with an action (e.g. TO_PLACE).
Grammar	Context-free grammar compiled into an ATN with probabilities assigned to each arc.
Parsing	The STINA parser is based on MIT's TINA parser. The syntactic tree is reduced to a semantic tree through deletion of all nodes and branches that contain no semantic information. Then a special process creates a semantic frame with slots corresponding to attribute-value information taken from the tree. The semantic frame has a feature specification describing which features are used in the frame and which information might have been added to the frame from the dialogue history. The parser runs with two different time scales corresponding to the words in each utterance and to the turns in the dialogue. Characteristics are a stack-decoding search strategy, a feature-passing mechanism to implement unification, and a robust parsing component. Parsing is done in three steps. The first step makes use of broad categories such as nouns, while the following step expands these into more detailed solutions. The last step involves re-calculation of hypothesis probabilities according to a multi-level N-gram model. The N-best list of hypotheses is re-sorted by the parser. Robust (skip nodes) and complete solutions compete all the time. In case of lack of understanding the user may be asked to provide a full sentence as input which will then be given a complete parse. The dialogue module controls the parser to accept incomplete sentences according to system dialogue questions. All words and grammar used by the system can be

	parsed.
Style	Different user styles were observed in the experiments.
Semantics	See parsing above.
Discourse, context	-
System utterances	
Generation	Generation of system utterances is done as part of dialogue management. A hierarchical structure where slots are filled in is used for generation. Everything is hardwired. Only slot values vary.
Lexicon	Lexical information is shared with the recognition module**.**
Grammar	There is no grammar.
Semantics	There is no semantics.
Style	Terse**.**
Processing	-
Discourse, context	-
Multimodal aspects	
Device(s)	The input device is a button. The output device is a screen.
None (unimodal system)	No.
Non-speech input	In addition to speech, the system accepts input from a button which can be pressed (simple haptic notation). You have to press the button while you speak. The button is also used to by-pass the spoken introduction (and thereby also move away from the written introduction).
Non-speech output	Static graphics output: pictures, maps, charts, time tables displayed on screen; synthetic face; speech recognition feedback (text). The graphics output cannot be scrolled by the user.
Role(s)	Output speech/facial: Lip movements synchronised with output speech, generated by the same algorithm. The head turns in order to "point" to new information displayed on the screen. The head will ask questions and react to user input. When information is displayed on the screen the face will briefly tell what you see.
Attentional state	
Focus, prior	The interaction model is a kind of graph structure and the focus is the point in the graph currently being addressed. However, this knowledge is only used for generating appropriate system output. There is no use of prior system focus (i.e. what the system has in active

	memory before the user's next utterance).
Sub-task identification	Sub-task identification is based on probabilities. A matrix of topics and features is being used. This matrix expresses, e.g., the probability of talking about restaurants if the word `food' is found in the input. Each semantic feature found in the syntactic and semantic analysis of user input is considered in the form of a conditional probability to decide on the topic. The probability for each topic is expressed as p(topic|F), where F is a feature vector including all semantic features used in the utterance. The system can only handle one topic at a time. Please note that this does not exclude response packages. Test on limited user data during WOZ yielded 12.9% topic recognition failure.
Expectations	Not used.
Intentional structure	
Task(s)	Information on boat timetables, port locations, hotels, camping grounds and restaurants in the Stockholm archipelago. The user may achieve departure and arrival times for boats, get a map displayed with the place of interest, get an overview of lodging and dining possibilities, get a presentation of possibilities for where to go. Adjacent tasks, such as price information and more specialised knowledge on, e.g., types of restaurants are not included. Also no booking tasks are included. In the WOZ experiments it was observed that people (when not carrying out the scenarios) often asked booking questions or asked for information at a more detailed level than what the system can provide, e.g. they might ask for menus or for a certain type of restaurant, e.g. fish restaurants. This problem is evidence that the task(s) handled by the system were not naturally circumscribed or that the users were not suitably informed of the system's task coverage.
Task complexity	The task is ill-structured in the sense that there are several different pieces of information which the user may or may not ask for and there is no particular order which s/he has to follow. The number of slots to be filled vary from sub-task to sub-task. For instance, the user may want to see a map. In that case there is only one slot to fill. On the other hand, if the user wants departure times the system must know departure and arrival port, date and in some cases a time interval. If a table contains more than seven lines of information, the table is not displayed. Instead the system asks a question in order to get more precise information and thus being

	able to throw away irrelevant information from the table.
Communication	Domain communication: Mixed initiative. The topic order is free but only one topic at a time can be handled by the system. Moreover, complex sentence structure cannot be handled. This basically means that the user has to express himself in main clauses. Natural response packages are allowed. Note that a crucial part of the system's domain communication is done graphically. System-initiated meta-communication: The system may ask for repetition and initiate repair meta-communication in case of missing understanding of user input (e.g. by asking for reformulation) or user input out of the system's domain. The system's vocabulary includes words from adjacent domains. This enables the system to tell the user, e.g., that is does not handle reservations rather than saying that it does not understand the user. User-initiated meta-communication: Users may ask for repetition of the system's speech output. Users may initiate repair, e.g. by saying `no, I said Waxholm, not Stockholm'. The user cannot initiate clarification dialogues with the system.
Interaction level	Most system questions are general and focused because they serve to elicit specific missing information items from the user. In case of understanding problems of a user answer to a system question the system requests a complete sentence from the user. The system thus has two interaction levels.
Implementation of dialogue management	The dialogue component is controlled by hand-crafted context free rules which control all steps in the dialogue including database search, speech and face synthesis, and other graphical feedback. Each step in the dialogue progress is associated to a new frame, with reference pointers to the history. The information that should be pushed forward is defined by semantic feature specifications in the dialogue rules. Each predicted dialogue topic is explored according to a set of rules. These rules define which constraints have to be fulfilled and what action should be taken depending on the dialogue history. A node can be a terminal node or the entry point to a network. In the case of a terminal, several node specific characteristics have to be defined in the rule system. These can be divided into three basic groups: constraint evaluation, system questions and system actions. The constraint evaluation is described in terms of features and in terms of the content in the

	semantic frame. If the frame needs to be expanded with additional information, a system question is synthesised. A positive response from the constraint evaluation opens the way for the selected action to take place, such as a database search or a graphic presentation of a map or a table. Most of the terminal nodes have no direct input from the user. Rather they deal with small details such as making the synthesis face look at a graphic object on the screen or to simply synthesise a message to the user. Thus, the dialogue rules do not only model turns in an oral dialogue, they cover all actions in the dialogue system. The expanded grammar notation makes it possible to separate the dialogue model from the system implementation.
Linguistic structure	
Speech acts	The system does not identify speech acts in the users' input.
Discourse particles	The system does not identify discourse particles in the users' input.
Co-reference	No co-reference resolution is performed. No problems have been observed as a result of this. No particular evaluation has been performed.
Ellipses	The system does not do any particular processing of ellipses. Incomplete sentences are allowed and ellipses are handled as such sentences. No problems have been observed. No particular evaluation has been performed.
Segmentation	No segmentation of user utterances is performed. No problems have been observed as a result of this. No particular evaluation has been performed.
Interaction history	
Linguistic	The system maintains a record of the surface language of the users' utterances. The dialogue is internally represented as a dialogue tree structure that allows reference specifications. However, the record is not being used for anything so far. No particular evaluation has been performed.
Topic	Each utterance is transformed into a semantic frame with attribute-value pairs. The time sequence of these frames is the history. A feature specification in the dialogue model tells what attributes to carry further and to keep in current use. Inside the time-table topic, for example, attributes corresponding to arrival port and so on are of interest, while other attributes are of interest to other topics. No particular evaluation has been performed.

Task	Graphical overview of the knowledge the user has received so far, in the form of listings of hotels and so on.
Performance	The system does not maintain a record of the user's performance during interaction. No particular evaluation has been performed.
Domain model	
Data	Boat timetables, port locations, hotels, camping grounds and restaurants in the Stockholm archipelago. The information is accessed through SQL queries to an ORACLE database.
Rules	The system will understand utterances that require temporal inference, such as "next Sunday before noon". Expressions such as 'first' or 'closest' or 'cheapest' are not being handled. The rules included are ad hoc rather than based on any systematic approach. No detailed evaluation of the rules has been carried out.
User model	
Goals	The user goal is assumed to be to obtain information on boat timetables, port locations, hotels, camping grounds and/or restaurants in the Stockholm archipelago.
Beliefs	The system does not specifically handle user beliefs during interaction.
Preferences	The system does not specifically handle user preferences during interaction.
User group	Any user; no distinction between user groups. No problems were observed as a consequence of this.
Cognition	Nothing has been done to explicitly take into account the specific cognitive characteristics of users, such as task load, limited memory, or limited attention span. However, it has been a goal the interaction with the system should be smooth. Hence, e.g., natural "response packages" are allowed and timetable information is listed on the screen rather than only read aloud.
Component architecture and function	
Generic architecture	-
Sub-components	The dialogue is described as a network. Different nodes have different functions, such as to synthesise an utterance. In this case an utterance frame with slots is described as part of the node description. The slots are filled with the help attribute names. An attribute can also be described with a frame, thus making the whole process recursive.
Flow	See Figure 2. The dialogue manager receives input from

	the linguistic part (grammar and semantics), interacts with the database and sends output to the output modules (graphics, synthesis and sound). Syntactic nodes and dialogue states are processed according to transition networks with probabilities on each arc.
Function	The dialogue manager is the control part of the system.
Architecture	
Platform	The system runs on HP UNIX OS 10. The platform is still adequate according to today's standards. The system runs on HP UNIX OS 10. No particular platform was used. The architecture is shown in Figure 2. The architecture is event-based.
Tools and methods	-
Generic	The system architecture is generic and domain independent. The generic software architecture seems to be event-based. The generic software architecture is still adequate according to today's standards.
No. components	See the architecture diagram in Figure 2.
Flow	See the architecture diagram in Figure 2.
Processing times	No information is available about the average percentages of processing time spent on the different processing tasks the system has.
System integration	
System resource utilisation	-
Shared information resources	-
Interactions	-
Data passing	-

It's a beautiful summer day and you are in Stockholm. You decide that you'd like to go to Waxholm. Your task is to find out when the boats leave for Waxholm this evening.

Figure 1. Scenario 1 used for all WOZ subjects.

2. System/component architecture

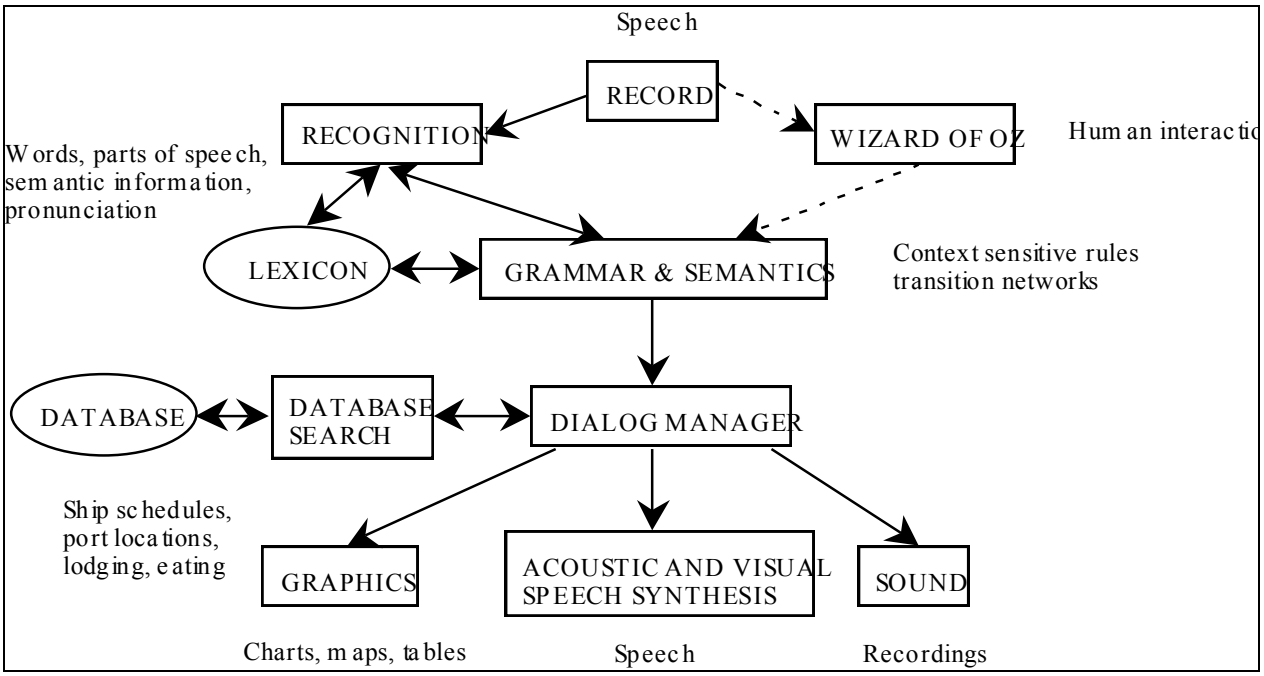


Figure 2. The modules in the Waxholm system.

3. System dialogue(s)

6 February 1998

U: jag vill åka till Vaxholm

S: Var ifrån vill du åka

U: Stockholm.

S: vilken veckodag vill du åka

U: jag vill åka på lördagen

S: det finns många båtar som går från stockholm till vaxholm på en lördag . när vill du åka.

U: jag vill åka på kvällen

S: detta är en tabell över de båtar som går från stockholm till vaxholm på en lördag efter klockan 17 och före klockan 24

U: var ligger som

S: Jag förstår inte

U: var är det Vaxholm

S: vaxholm finns på denna karta

U: finns det en Vaxholm

S: vaxholm finns på denna karta

U: finns det restaurang i Vaxholm

S: information om vaxholm restauranger visas i tabellen

Du talar för starkt. Kan du repetera.

U: vilken båt är samma morgon

S: Vad letar du efter.

U: finns

S: Jag förstod inte. Jag kom så här långt. finns . Kan du formulera om dig. Jag förstår inte. Vad letar du efter.

S: Jag förstår inte

U: jag letar efter sista stan

S: stockholm sista finns på denna karta

U: nu sista sedan

S: Jag förstår inte

U: jag letar sista

S: Jag förstår inte

U: ka söndag hem

S: Jag förstår inte

U: var kostar båten

S: Vad letar du efter

U: jag vill åka till Sandhamn

S: jag söker båtar som går till sandhamn

S: Var ifrån vill du åka.

U: finns det den

S: Jag förstår inte.

U: finns det hotellen

S: information om hotell visas i tabellen

U: tack

S: Tack ska du ha.

21 October 1993

S: Tack för att du vill delta i det här experimentet. Du kommer att få genomföra ett antal uppgifter, som gäller båt-trafik med mera i Stock-olms SJÄRGÅD. Du talar till datorn och datorn ger svar. Ibland går det litet långsamt och då får du vänta. Din röst kommer att spelas in. Tryck på start, med hjälp av musen innan du börjar tala och håll den intryckt så länge du talar. Då det står, "VÄNTA", i rutan måste du vänta.

S: För att göra dig bekant med inspelningsproceduren kommer datorn att be dig säga ett antal saker. Om du inte vill säga ditt eget namn kan du välja något annat. Tryck på knappen för att spela in. Släpp knappen för att avsluta inspelningen.

S: Var så god att börja. Tryck på knappen för att spela in.

S: Kan du repetera.

S: Du befinner dig i Stockholm en vacker SÅMARhyDA:G. Du bestämmer dig för att åka till Waxholm. Din uppgift är att ta reda på när båtarna går dit i kväll.

S: Var så god att börja. Tryck på knappen för att spela in.

U: jag vill åka till vaxholm ikväll .

S: MMMMM Var ifrån vill du åka.

U: från stockholm .

S: MMMMM vilken veckodag vill du åka.

U: på torsdag .

S: Programmet söker nu i databasen .

S: detta är en tabell över de båtar som går från stockholm till vaxholm på en torsdag .

FÖLJANDE BÅTAR GÅR FRÅN STOCKHOLM TILL VAXHOLM PÅ TORSDAGAR :

BÅT AVGÅNG ANKOMST AVG.HAMN

SKARPÖ 17.30 18.40 STRÖMKAJEN

NORRSKÄR 18.30 19.45 STRÖMKAJEN

S: Vad vill du nu veta.

U: när kan man åka hem.

S: Jag förstår inte.

U: när går båtarna tillbaka till stockholm.

S: Programmet söker nu i databasen .

S: det finns inga båtar som går från stockholm till stockholm på en torsdag .

U: när går båtarna från vaxholm till stockholm på torsdag kväll .

S: Programmet söker nu i databasen .

S: Det finns många båtar, som går från vaxholm till stockholm på en torsdag. När vill du åka på dagen?

U: på kvällen .

S: Vänta, jag letar nu.

S: detta är en tabell över de båtar som går från vaxholm till stockholm på en torsdag .

FÖLJANDE BÅTAR GÅR FRÅN VAXHOLM TILL STOCKHOLM PÅ TORSDAGAR:

BÅT AVGÅNG ANKOMST ANK.HAMN

VÄDDÖ 17.55 18.50 STRÖMKAJEN

SJÖBRIS 18.00 18.45 NYBROKAJEN

VÄSTAN 18.55 20.10 STRÖMKAJEN

WAXHOLMII 19.45 20.55 STRÖMKAJEN

NORRSKÄR 20.30 21.40 STRÖMKAJEN

VÄRMDÖ 20.45 21.30 STRÖMKAJEN

STORSKÄR 21.00 22.10 STRÖMKAJEN

VÅNÖ 21.10 21.55 STRÖMKAJEN

S: Vad vill du nu veta.

U: tack nu räcker det .

S: Du vill åka ut i skärgården under helgen. För att ha det bekvämt vill du bo på hotell. Du slutar klockan 15 på fredagen och börjar klockan 10 på måndagen. Ta reda på var man kan bo och när du kan åka.

S: Var så god att börja. Tryck på knappen för att spela in.

U: var finns det hotell i skärgården .

S: Informationen om hotell visas i tabellen .

FÖLJANDE HAMNAR HAR HOTELL:

DALARÖ

FURUSUND

GRUVBRYGGAN

NORRALAGNÖ

NORRTÄLJE

NYNÄSHAMN

SALTSJÖBADEN

SANDHAMN

STAVSNÄS

VAXHOLM

VÄSBYSTRAND

U: var ligger saltsjöbaden .

S: saltsjöbaden finns på denna karta .

U: när går båtarna från stockholm till saltsjöbaden på fredag kväll .

S: Programmet söker nu i data-basen .

S: detta är en tabell över de båtar som går från stockholm till saltsjöbaden på en fredag .

FÖLJANDE BÅTAR GÅR FRÅN STOCKHOLM TILL SALTSJÖBADEN PÅ FREDAGAR:

BÅT AVGÅNG ANKOMST AVG.HAMN

ROSLAGEN 8.40 9.40 STRÖMKAJEN

ÖSTAN 11.00 12.00 STRÖMKAJEN

ÖSTAN 16.45 17.45 STRÖMKAJEN

S: Vad vill du nu veta.

U: kan man åka tillbaka på måndag morgon .

S: Programmet söker nu i data-basen .

S: detta är en tabell över de båtar som går från stockholm till saltsjöbaden på en måndag .

FÖLJANDE BÅTAR GÅR FRÅN STOCKHOLM TILL SALTSJÖBADEN PÅ MÅNDAGAR:

BÅT AVGÅNG ANKOMST AVG.HAMN

SKÄRGÅRDEN 8.40 9.40 STRÖMKAJEN

S: Vad vill du nu veta.

U: jag vill åka i andra riktningen .

S: Programmet söker nu i data-basen .

S: detta är en tabell över de båtar som går från stockholm till saltsjöbaden på en måndag .

FÖLJANDE BÅTAR GÅR FRÅN STOCKHOLM TILL SALTSJÖBADEN PÅ MÅNDAGAR:

BÅT AVGÅNG ANKOMST AVG.HAMN

SKÄRGÅRDEN 8.40 9.40 STRÖMKAJEN

S: Vad vill du nu veta.

U: när går båtarna från saltsjöbaden till stockholm på måndag morgon .

S: Programmet söker nu i data-basen .

S: detta är en tabell över de båtar som går från saltsjöbaden till stockholm på en måndag .

FÖLJANDE BÅTAR GÅR FRÅN SALTSJÖBADEN TILL STOCKHOLM PÅ MÅNDAGAR:

BÅT AVGÅNG ANKOMST ANK.HAMN

SKÄRGÅRDEN 19.00 20.00 STRÖMKAJEN

S: Vad vill du nu veta.

U: när går båtarna på söndag .

S: Programmet söker nu i data-basen .

S: detta är en tabell över de båtar som går från saltsjöbaden till stockholm på en söndag .

FÖLJANDE BÅTAR GÅR FRÅN SALTSJÖBADEN TILL STOCKHOLM PÅ SÖNDAGAR:

BÅT AVGÅNG ANKOMST ANK.HAMN

SKÄRGÅRDEN 20.15 21.15 STRÖMKAJEN

S: Vad vill du nu veta.

U: ja tack det räcker för den här gången .

S: Du befinner dig i Vaxholm och vill åka till Vega-bryggan. Det är tisdag och du vill åka efter lunch. Din uppgift är att ta reda på vilka båtar du kan ta.

S: Var så god att börja. Tryck på knappen för att spela in.

U: hur kommer man från vaxholm till vegabryggan idag.

S: Programmet söker nu i data-basen .

S: detta är en tabell över de båtar som går från vaxholm till vegabryggan på en torsdag .

FÖLJANDE BÅTAR GÅR FRÅN VAXHOLM TILL VEGABRYGGAN PÅ TORSDAGAR:

BÅT AVGÅNG ANKOMST

SKARPÖ 10.00 (10.15)

STORSKÄR 13.10 (13.30)

SKARPÖ 14.25 (14.45)

SILVERÖ 16.15 (17.20)

ÖSTAN 16.55 (17.20)

SILVERÖ 18.05 (18.20)

SKARPÖ 18.40 (19.10)

S: Vad vill du nu veta.

U: jag vill åka på tisdag .

S: Programmet söker nu i data-basen .

S: detta är en tabell över de båtar som går från vaxholm till vegabryggan på en tisdag .

FÖLJANDE BÅTAR GÅR FRÅN VAXHOLM TILL VEGABRYGGAN PÅ TISDAGAR:

BÅT AVGÅNG ANKOMST

SKARPÖ 10.00 (10.15)

STORSKÄR 13.10 (13.30)

SKARPÖ 14.25 (14.45)

SILVERÖ 16.15 (17.20)

ÖSTAN 16.55 (17.20)

SILVERÖ 18.05 (18.20)

SKARPÖ 18.40 (19.10)

S: Vad vill du nu veta.

U: ja det var allt jag ville veta tack ska ni ha .

28 February 1994

S: Tack för att du vill delta i det här experimentet. Du kommer att genomföra ett antal uppgifter, som gäller båt-trafik med mera i Stock-olms SJÄRGåD. Du talar till datorn och datorn ger svar. Ibland går det litet långsamt och då får du vänta. Din röst kommer att spelas in. Tryck på start, med hjälp av musen innan du börjar tala och håll den intryckt så länge du talar. Då det står, "VÄNTA", i rutan måste du vänta. För att göra dig bekant med inspelningsproceduren kommer datorn att be dig säga ett antal saker. Om du inte vill säga ditt eget namn kan du välja något annat.

S: Tryck på knappen för att spela in. Släpp knappen för att avsluta inspelningen.

S: Var så god att börja. Tryck på knappen för att spela in.

S: Kan du repetera.

S: Du befinner dig i Stockholm en vacker SÅMARhyD'A:G. Du bestämmer dig för att åka till Waxholm. Din uppgift är att TA reda på när båtarna går dit i kväll.

S: Var så god att börja. Tryck på knappen för att spela in.

U: när går båtarna till stockholm ikväll .

S: Var ifrån vill du åka.

U: från stockholm .

FÖLJANDE BÅTAR GÅR FRÅN STOCKHOLM TILL STOCKHOLM PÅ MÅNDAGAR:

BÅT AVGÅNG ANKOMST AVG.HAMN ANK.HAMN

ÖSTAN 16.00 (16.00) STRÖMKAJEN SLUSSEN

ROSLAGEN 17.00 (17.00) STRÖMKAJEN SLUSSEN

ROSLAGEN (6.50) 7.45 SLUSSEN STRÖMKAJEN

VÄDDÖ (7.10) 7.40 SLUSSEN STRÖMKAJEN

ÖSTAN (7.15) 7.50 SLUSSEN STRÖMKAJEN

ROSLAGEN (7.30) 8.25 SLUSSEN STRÖMKAJEN

S: detta är en tabell över de båtar som går från stockholm till stockholm på en måndag .

U: vilka båtar går efter 17 .

S: det finns inga båtar som går från stockholm till stockholm på en måndag efter klockan 17 .

U: jag vill åka till vaxholm istället .

FÖLJANDE BÅTAR GÅR FRÅN STOCKHOLM TILL VAXHOLM PÅ MÅNDAGAR :

BÅT AVGÅNG ANKOMST

SKARPÖ 17.30 18.40

NORRSKÄR 18.30 19.45

S: detta är en tabell över de båtar som går från stockholm till vaxholm på en måndag efter klockan 17.

U: jag vill åka den snabbaste båten dit .

FÖLJANDE BÅTAR GÅR FRÅN STOCKHOLM TILL VAXHOLM PÅ MÅNDAGAR :

BÅT AVGÅNG ANKOMST

SKARPÖ 17.30 18.40

NORRSKÄR 18.30 19.45

S: detta är en tabell över de båtar som går från stockholm till vaxholm på en måndag efter klockan 17.

U: jag åker med skarpö .

S: FÖLJANDE BÅTAR GÅR FRÅN STOCKHOLM TILL VAXHOLM PÅ MÅNDAGAR :

BÅT AVGÅNG ANKOMST

SKARPÖ 17.30 18.40

NORRSKÄR 18.30 19.45

S: detta är en tabell över de båtar som går från stockholm till vaxholm på en måndag efter klockan 17 .

U: då åker jag med skarpö avgång 17 och 30 .

FÖLJANDE BÅTAR GÅR FRÅN STOCKHOLM TILL VAXHOLM PÅ MÅNDAGAR :

BÅT AVGÅNG ANKOMST

SKARPÖ 17.30 18.40

NORRSKÄR 18.30 19.45

S: detta är en tabell över de båtar som går från stockholm till vaxholm på en måndag efter klockan 17 .

U: tack .

S: Tack ska du ha.

S: Hitta på en destination och ta reda på om det går att ta sig dit. Ta även reda på var den ligger.

S: Var så god att börja. Tryck på knappen för att spela in.

U: var ligger marö .

S: marö finns på denna karta .

U: jag vill åka till marö ikväll .

S: Var ifrån vill du åka.

U: stockholm .

FÖLJANDE BÅTAR GÅR FRÅN STOCKHOLM TILL MARÖ PÅ MÅNDAGAR:

BÅT AVGÅNG ANKOMST AVG.HAMN

SJÖBRIS 9.00 (12.00) NYBROKAJEN

S: detta är en tabell över de båtar som går från stockholm till marö på en måndag .

U: tack .

S: Tack ska du ha.

S: Du ska hälsa på en vän på Ramsöberg, och vill stanna så länge som möjligt. Ta reda på när båten går.

S: Var så god att börja. Tryck på knappen för att spela in.

U: när går sista båten från ramsberg till stockholm .

S: Var ifrån vill du åka.

U: från ramsöberg .

S: vilken veckodag vill du åka.

U: idag .

FÖLJANDE BÅTAR GÅR FRÅN RAMSÖBERG TILL STOCKHOLM PÅ MÅNDAGAR:

BÅT AVGÅNG ANKOMST ANK.HAMN

SKARPÖ 10.25 12.10 STRÖMKAJEN

SKARPÖ 15.15 16.55 STRÖMKAJEN

S: detta är en tabell över de båtar som går från ramsöberg till stockholm på en måndag .

U: tack .

S: Tack ska du ha.

4. Waxholm screen shot(s)

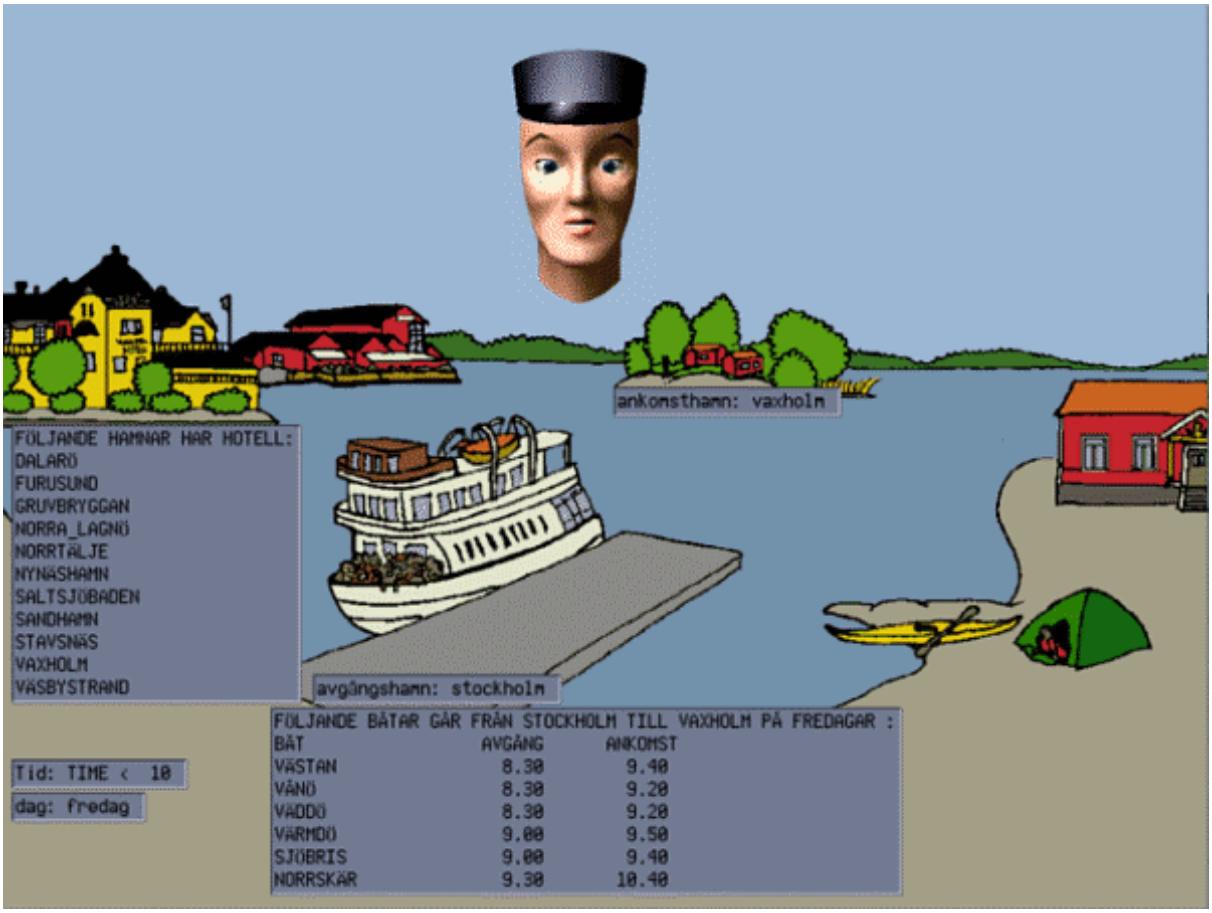


Figure 3. Screen shot from the Waxholm system.

7.12 Waxholm Dialogue Manager Life Cycle

Laila Dybkjær and Niels Ole Bernsen

The Maersk Institute, Odense University
5230 Odense M, Denmark
laila@mip.ou.dk, nob@mip.ou.dk

1. Life cycle description

Overall design goal(s):

What is the general purpose(s) of the design process?

The aim is to build a generic, i.e. partially re-usable, multimodal spoken dialogue system in which speech synthesis and speech recognition can be studied in a human-machine dialogue framework. The system concept is quite advanced for its time and represents an eminently worthwhile goal. The same is true of the objective of achieving re-usability. The emphasis on speech synthesis and speech recognition marks the project as one of exploratory research. There was no particular focus on dialogue management. The effect seems to be that there are fewer detailed analyses and exact measurements of results than one would have expected had this component been a focus point.

Hardware constraints:

Were there any a priori constraints on the hardware to be used in the design process?

A powerful (for that time) computer was needed to be able to do real-time processing. Even then, the vocabulary was limited to about 1000 words. Also the recognition quality put a limit to the vocabulary size. These conditions appear reasonable for the time.

Software constraints:

Were there any a priori constraints on the software to be used in the design process?

An existing synthesiser as well as an existing recogniser were used, neither of them off the shelf products. Rather they were software components that were under continuous development. The effort on the recogniser was put in focus during the project. The main research effort on speech synthesis was carried out outside the Waxholm project and lack of personnel was a major problem.

Customer constraints:

Which constraints does the customer (if any) impose on the system/component? Note that customer constraints may overlap with some of the other constraints. In that case, they should only be inserted once, i.e. under one type of constraint.

No customer constraints were introduced for this research system.

Other constraints:

Were there any other constraints on the design process?

The main limiting factors were personpower and knowledge. Preferably more people should have been involved but it was hard to find good people with the right speech technology background. One major constraint was that it was the first time the developers were involved in such a project. They made some mistakes that slowed them down. These were the kinds of mistakes people make when a domain is new and unexplored.

Design ideas:

Did the designers have any particular design ideas which they would try to realise in the design process?

The dialogue should be described by a grammar; the dialogue manager should be probabilistic, i.e. a matrix of topics and features expressing probabilities of what topic the user is talking about is central to the dialogue management, cf. Figure 7 in [Carlson and Hunnicutt 1995]. The probabilistic dialogue manager based on keywords in the users' input is an interesting innovative feature of Waxholm.

Designer preferences:

Did the designers impose any constraints on the design which were not dictated from elsewhere?

No Lisp or Prolog person was involved so the developers were using C throughout the project (except SQL for database search).

Design process type:

What is the nature of the design process?

Exploratory research mainly on speech recognition in human-machine dialogue.

Development process type:

How was the system/component developed?

The development of the system may be described in terms of four phases. 1. The idea was conceived and initial preparations done. 2. First WOZ experiments were performed with about 30 users. 3. A second series of WOZ experiments were performed with about 40 users. 4. The full system was implemented.

The initial preparations included interviews with 3-4 people from the Waxholm company and a timetable was collected from the company. The developers were not allowed to make recordings in the Waxholm company. The initial system was based on what the 5-6 people in the project could come up with as regards lexicon, grammar, dialogue model etc. During WOZ, text (from the wizard's typing) and speech data were collected running the system with a wizard replacing the speech recognition module. Data were eventually collected from users interacting with the final system. After the first subjects were recorded, both the grammar and the dialogue model could be based on empirical data. The lexicon was expanded and the network probabilities were trained. After the second phase some major revisions were made based on observed interaction problems. No particular development methodology appears to have been followed. A new phase of development was entered when the team believed that the system had been sufficiently improved based on data from the previous phase.

Requirements and design specification documentation:

Is one or both of these specifications documented?

No. However, during project meetings goals were agreed upon. Whether the goals had been reached was discussed during the following meetings.

Development process representation:

Has the development process itself been explicitly represented in some way? How?

Initially the system was designed based on intuition and discussions within the group. At an early stage the block diagram in Figure 2 was made that later on has been presented in many papers. One person had the task of implementing the information flow between modules (represented as boxes in the figure), and the data format was negotiated between the different people that were responsible for modules that should be connected.

The project group had a weekly meeting with a written protocol with numbered tasks: small and big ones, from "fix the loudspeaker" to "improve the recognition". At each meeting the list was gone through and the numbered tasks were taken off when finished. During these meetings design issues played an important role. The protocols exist but are not publicly available.

See also the Waxholm history below. The absence of any explicit development process representation means that Waxholm will be more difficult than needed to re-design and maintain, especially for people who were not involved in its development. Explicit design representation is a good support for newcomers.

Realism criteria:

Will the system/component meet real user needs, will it meet them better, in some sense to be explained, than known alternatives, is the system/component "just" meant for exploring specific possibilities (explain), other (explain)?

Realism and meeting real user needs was desired though not the main goal. The main goal was to explore speech-based dialogue systems and specifically speech components and their interaction. The Waxholm idea was chosen because it seemed a realistic application and was open-ended in the sense that the domain could easily be extended. The Waxholm travel agents provide person dependent information (e.g. depending on age). This is not being done by the system. It can hardly be claimed that the system will meet user needs better than the Waxholm travel agents. The Waxholm system was only in a loose sense meant to meet real user needs and relatively little effort was spent on ensuring that it did so. Thus the project had no: extended end-user contact, extensive work on domain delimitation, clear up-front performance criteria, final adequacy criteria, extended quantitative and qualitative evaluation throughout the development process, an explicit development methodology.

Functionality criteria:

Which functionalities should the system/component have (this entry expands the overall design goals)?

The demonstrator application gives information on boat traffic in the Stockholm archipelago, including information about port locations, hotels, camping sites and restaurants. The user may obtain departure and arrival times for boats, have a map displayed with the place of interest, get an overview of lodging and dining possibilities, get a presentation of possible places to visit. This list of Waxholm functionalities represents a sub-section of the information users may

want to have before going on a boat trip in the Stockholm archipelago. The developers themselves found evidence that this sub-section will not necessarily satisfy real users.

Usability criteria:

What are the aims in terms of usability?

The system is a walk-up-and-use system which requires no previous training of its users. The system should be able to perform a smooth dialogue with its users. Dialogue "smoothness" was never defined as an operational parameter in the Waxholm project, e.g. in terms of the development methodology, approach to task domain delimitation, system co-operativity parameters, or other evaluation criteria.

Organisational aspects:

Will the system/component have to fit into some organisation or other, how?

N/A. Waxholm might have been installed as an independent information booth.

Customer(s):

Who is the customer for the system/component (if any)?

There is no customer. The Waxholm company just thought it was fun. They have no intention to push the system to a final product that is robust enough for public use. No recordings were allowed by the company but the developers were allowed to interview 3-4 travel agents once. Later in the process the developers have contacted the Waxholm company several times to discuss specific issues. The company was not involved in domain delimitation.

Users:

Who are the intended users of the system/component?

Users must speak Swedish; walk-up-and-use users. Walk-up-and-use is an appropriate paradigm for the application.

Developers:

How many people took significant part in the development? Did that cause any significant problems, such as time delays, loss of information, other (explain)? Characterise each person who took part in terms of novice/intermediate/expert wrt. developing the system/component in question and in terms of relevant background (e.g., novice phonetician, skilled human factors specialist, intermediate electrical engineer).

Development of the dialogue component was made mainly by two engineers, one of which had long speech technology experience. Neither of them had a formal human factors training.

Development time:

When was the system developed? What was the actual development time for the system/component (estimated in person/months)? Was that more or less than planned? Why?

Roughly one and a half person years were spent on the dialogue component. It was about the time that could be afforded. The Waxholm project was a three year project involving about 6-8 researchers, but it is hard to say what specifically is Waxholm and what is long term research on the basic technology. An initial version of the system based on text input has been in operation since September 1992. The main emphasis was not on dialogue component

development.

Requirements and design specification evaluation:

Were the requirements and/or design specifications themselves subjected to evaluation in some way, prior to system/component implementation? If so, how?

Since there were no requirement or design specifications, this was not possible.

Evaluation criteria:

Which quantitative and qualitative performance measures should the system/component satisfy?

No particular evaluation criteria were set up from the very beginning. The parameters evaluated during the development process were relatively arbitrary and not based on any systematic approach. The parameters evaluated were:

Number of turns: The number of turns needed to get a specific task completed was measured. The measurements were not related to any performance targets and their significance was not evaluated.

Naturalness - mixed initiative dialogue: A natural mixed initiative dialogue, rather than a prompted system-directed one, was given high priority, so the focus came to be more on the naturalness than on reaching the goal as fast as possible. No particular evaluation was made of the success of the mixed initiative dialogue strategy and no performance targets were formulated.

Naturalness - no interaction problems: During WOZ the developers looked out for interaction problems and problems observed were fixed as far as possible. However, no detailed and systematic analysis of user-system interaction and identification of problems was performed.

Robustness - of topic identification: It was evaluated how robust the topic identification was with erroneous recognition and so on.

Transaction success: -

The above measures were discussed at an early stage and formalised later during the project.

Evaluation:

At which stages during design and development was the system/component subjected to testing? How? Describe the results.

Dialogue management: As regards the entire system and in particular the dialogue manager, only data from the Wizard of Oz experiments was evaluated in any detail, see below.

Phases 2 and 3: WOZ scenario-based simulation and progress evaluation: 66 subjects (17 female) each received three scenarios, the first one always being the same, (Figure 1). Scenarios were presented both as text on the screen and in synthetic speech. Users were encouraged also to use the system beyond the scope of the scenarios. A total of 14 scenarios were used. The scenarios were meant to cover different aspects of the system's domain. 14 scenarios were what the developers came up with without starting to repeat themselves. Scenario development was not principled otherwise. A problem was that users tended to re-use the vocabulary from the scenarios. Each scenario required that the user solved from one to four subtasks, a subtask consisting in, e.g., requesting a timetable, a map or a list of facilities. Each subtask required specification of several different constraints, such as departure port, destination port and departure day. Subjects had to provide the system with up to ten such

constraints, with a mean of 4.3, in order to solve a complete scenario.

After the session subjects filled in a questionnaire with questions about weight, height, age, profession, dialect, speaking habits, native tongue, comments about the experiment, etc. Most subjects were department staff or undergraduate students. The issues addressed in the questionnaire mainly pertain to speech. The questionnaire was not developed in a principled way other than it contained a few questions which were considered relevant. The answers to the questionnaire have not been evaluated in detail. Age and position of participants have been extracted from the questionnaires and are shown in graphs presented in papers.

A total of 198 dialogues were recorded and analysed. The dialogues contained 1900 user utterances and 9200 words. The total recording time amounts to 2 hours and 16 minutes. After the first 37 sessions (35 subjects) all system parts went through a major revision. The first phase included approximately 1000 subject utterances. The responses "I do not understand" and "You have to reformulate" occurred in 35.8% of the system responses. In the second phase the dialogue manager was revised as well as the scenarios. In this phase 31 subjects produced 900 utterances. The improved system failed to understand 20.9% of the time. The system responded "I don't understand" 575 times corresponding to 268 occasions where consecutive repetitions are counted as one occasion. In 50% of the cases the system recNT COLOR="#000000"** [We might move the following (Complete parses, Perplexity, Word error rate) to a "reserve" file, inserting any significant measurements in the grid? It is not about dialogue management.]

Complete parses: The parser has been evaluated in several different ways. Most tests used a deletion estimation procedure. In the WOZ experiments 62% of 700 all people address within-domain issues. The vocabulary contains words from adjacent domains in order for the system to recognise such input and therefore give more precise answers, such as "I cannot make hotel reservations".

About 700 utterances are simple answers to system questions while the rest, 1200, can be regarded as user initiatives.

Word recognition accuracy was measured at 76.0% and later improved to 78.6% (laboratory).

The average utterance length was 5.6 words. The average length of the first utterance in the dialogues was 8.8 words. The utterance length distribution shows one maximum at two words and one at five words (corresponding to ellipses versus full sentences). Less than 3% of the utterances contain restarts like repetition of a word or a phrase or changes of a word. About one fourth of the restarts occur in interrupted words, that is, in words that are not phonetically completed.

Most system questions occurred when the system understood that the subject wanted a timetable displayed. If information was missing, the system took the initiative to ask for this information. The subjects answered the system questions in 95.4% of the cases. Thus subjects were co-operative. Only about 1% changed the topic during the system-controlled dialogue.

Transaction success: The database contains 265 subtasks, about 84% of which were solved by the subjects. In 75% of the cases, 199 out of 265, the subjects had completed a subtask after one to five utterances. The subjects needed about seven utterances to solve one scenario. After the task was completed several subjects continued to ask questions in order to test the system. About three additional utterances per scenario were collected in this way. In 42 cases a scenario had been designed so that it could not be completely solved by a subject, corresponding to an a priori error rate of 21%. In half of these, 21 scenarios, some of the subtasks were solved by the subjects.

Robustness - of topic identification: The topic identification method has been evaluated by using one quarter of the data, about 300 utterances, as test data, and the rest as training data, about 900 utterances. This procedure has been repeated for all quarters. The reported results are the mean values from these four runs. The eight possible topics (TIME-TABLE, SHOW_MAP, EXIST, TRIP_MAP, END_SCENARIO, REPEAT, NO_UNDERSTANDING, OUT_OF_DOMAIN) have a rather uneven distribution in the material with TIME_TABLE occurring 45% of the time. The topic NO_UNDERSTANDING is trained on a set of constructed utterances that are impossible to understand, even for a human. This topic is then used as a model for the system to give an appropriate "no understanding" system response. In principle, these utterances can still have a reasonable parse. However, the topic selection is certainly influenced by a poor parse. Using the unprocessed labelled input transcription yields 12.9% errors. By excluding 55 utterances, about 5% of the test corpus, predicted to be part of the "no understanding" topic, error is reduced by about 4% (8.8). When all extra-linguistic sounds, about 700, are excluded from the input material the number of complete parses increases by about 10%. The prediction result (12.7% error and 8.5% with the "no understanding" utterances excluded) was about the same as in the first experiment. When only utterances giving a complete parse are included errors are reduced to 3.1% (2.9%). It is not known if an increased grammatical coverage will reduce the topic prediction errors.

Complete parses: The parser has been evaluated in several different ways. Most tests used a deletion estimation procedure. In the WOZ experiments 62% of 700 utterances (user answers) gave a complete parse while 48% of 1200 utterances (user initiatives) gave a complete parse. Responses to system questions typically have a very simple syntax. If extra-linguistic sounds such as lip smack, sigh and laughing are excluded from the user initiative material, the result is increased to 60% complete parses. Sentences with incomplete parses are handled by the robust parsing component and frequently affect the desired system response.

Perplexity: The perplexity of the Waxholm data is about 26 using a trained grammar. If only utterances with complete parses are considered the perplexity is 23. On an HP 735 it takes about 17 msec to process an utterance.

Word error rate: The parser has also been evaluated in an N-best list resorting framework. Totally 290 N-best lists with about 10 alternatives each were generated, using an early version of the speech recognition module. Several of the utterances were answers to simple questions and the average utterance length was about five words. The top choice using a bigram grammar as part of the recognition module gave a word accuracy of 76.0%. The mean worst and best possible accuracy in the lists were 48% and 86.1%. After resorting to use the STINA parser the result improved to 78.6% corresponding to about 25% of the possible increase.

Re-usability: -

Multimodal aspects: Most of the graphics was only added after the last WOZ experiments: the face, speech recognition feedback, tables in different places (earlier all information was displayed in the same place). These additions were made to cope with problems observed in the WOZ experiments. The additions have not been evaluated.

Mastery of the development and evaluation process:

Of which parts of the process did the team have sufficient mastery in advance? Of which parts didn't it have such mastery?

The main task of the project was to learn and model, which was a success. The project was largely a competence-building exercise.

Problems during development and evaluation:

Were there any major problems during development and evaluation? Describe these.

No human problems were encountered but several significant technical problems outside the scope of the project, such as system problems and hardware problems. Several acoustic problems were experienced and also system software problems reduced the development speed.

Development and evaluation process sketch:

Please summarise in a couple of pages key points of development and evaluation of the system/component. To be done by the developers.

WOZ experiments were performed to collect a database of user utterances. The duration of the WOZ experiments was a time span of about 4 months. Two iterations were made. The first iteration involved about 30 subjects, the second involved about 40 subjects. Between the iterations, major revisions of all system parts were made on the basis of the collected experiences. On-the-fly modifications were made throughout the experiments. In particular in the beginning the system modules did not perform very well. A wizard replaced the speech recogniser; early versions of the other system modules were used. The subjects are seated in an anechoic room in front of a display. A high-quality microphone was used. The wizard is seated in an adjacent room facing two screens, one displaying what is shown to the subject and the other providing system information. The subjects all knew that the wizard replaced the speech recognition. All utterances were recorded at 16 kHz and stored together with their respective label files. The label files contain orthographic, phonemic, phonetic and durational information. A `label file' is a transcribed and annotated corpus bit. The MIX standard annotation was used which may easily be transformed into wav format. All system information is logged. An experimental session starts with a system introduction presented in text on the screen. The text is also read by speech synthesis, thus permitting the subject to adapt to the synthetic voice. The subjects practised the push-to-talk procedure by reading a sound calibration sentence and eight phonetically rich reference sentences.

Waxholm history:

91-09 Preparatory phase - Block diagram (see Figure 2).

91-12 First grammar rule set in parser.

92-07 Network development + Oracle implementation.

93-03 First pilot recording + Recording protocol + Struggling with audio quality**.**

93-07 Formal start of project in the Language Technology programme

93-09 Data collection starts (first WOZ).

A*-search implemented and connected to phoneme neural network.

Lexicon structure, transcription - normative for all modules.

Automatic labelling (MIX standard used for annotation (may easily be transformed to .wav)) - name conventions, non-language symbols.

93-11 All modules including recognition connected.

Data analysis using evaluation software.

Phoneme network trained on Wizard of Oz recordings.

First speaker-independent evaluation.

94-01 Implementation of rule-controlled dialogue.

94-05 Evaluation of collected data:

Recognition performance + Syntax analysis + Dialogue control.

94-07 Improved search method + word network output.

True parse probabilities + improved robust parse.

Labelling and labelling....

94-10 First complete run-time version of the system (video available).

Improvements... improvements...

Rule controlled dialogue.. topic prediction.

Recognition accuracy...

96-06 End of project

96-10 On display at the technical fair in Stockholm.

Component selection/design:

Describe the components and their origins.

All parts of the system were developed in-house. A graphical interface presents the dialogue grammar graphically. Both the syntax and the dialogue networks can be modelled and edited graphically with this tool. There is an interactive development environment (it is possible to study the parsing and the dialogue flow step by step when a graphic tree is being built). It is possible to use log files collected during WOZ as scripts to repeat a specific dialogue, including all graphic displays and acoustic outputs.

A multi-lingual text-to-speech system was modified for the application. A face-synthesis module was built. Both the visual and the speech synthesis are controlled by the same synthesis software.

The STINA parser is inspired by MIT's TINA parser. The first grammar and lexicon was based on experiences from TINA plus pure guesses. STINA is now very different from TINA.

Graphic feedback showing the system's recognition of the user's spoken input was introduced to deal with the problem observed during WOZ that users often did not get enough feedback to be able to decide if the system had the same interpretation of the dialogue as the user.

Robustness:

How robust is the system/component? How has this been measured? What has been done to ensure robustness?

A lot of work was spent to make the system technically robust. No formal measurements have been made except fixing the problems when they were noticed. No particular robustness metrics have been adopted or used.

Maintenance:

How easy is the system to maintain, cost estimates, etc.

Given that the machine is not changed maintenance is at a very low cost. The Waxholm system is not directly under development any more but it is kept running for demonstrations from time to time. Maintenance cost are kept at a minimum, i.e. a couple of days per year. The project never aimed at design for sustained development and re-design.

Portability:

How easily can the system/component be ported?

To adjust to a new machine would be very costly. It would take a lot of work to port the system. Implementation methods should be changed in that case.

Modifications:

What is required if the system is to be modified?

A modification of the domain implies an addition concerning how to handle a new topic. It is the ambition that the implementation and the training procedures should, as much as possible, be kept the same. A modification requires new lexical, grammar, topics description, database and graphics.

Additions, customisation:

Has a customisation of the system been attempted/carried out (e.g. modification of a part of the vocabulary, new domain/task, etc.)? Has there been an attempt to add another language? How easy is it (how much time/effort) to adapt/customise the system to a new task? Is there a strategy for resource updates Is there a tool to enforce that the optimal sequence of update steps is followed ?

This has not been attempted.

Property rights:

Describe the property rights situation for the system/component.

The system/component belongs to KTH.

2. Documentation of the design process

See Figures 1, 2 and 3 in the grid description.

3. References to additional project/system/component documentation

Bertenstam, J. Blomberg, M., Carlson, R., Elenius, K, Granström, B., Gustafson, J., Hunnicutt, S., Högberg, J., Lindell, R., Neovius, L., de Serpa-Leitao, A., Nord, L. and Ström, N. (1995):" Spoken dialogue data collection in the Waxholm project" STL-QPSR 1/1995, pp. 50-73. (paper - 23 pages, Postscript 468kb).

Carlson, R., Hunnicutt, S. and Gustafson, J. (1995):" Dialogue management in the Waxholm system" Proc. Spoken Dialogue Systems, Vigsø (paper - 4 pages, Postscript 109kb).

Bertenstam, J. Blomberg, M., Carlson, R., Elenius, K, Granström, B., Gustafson, J., Hunnicutt, S., Högberg, J., Lindell, R., Neovius, L., de Serpa-Leitao, A., Nord, L. and Ström, N. (1995):" The Waxholm system - a progress report" Proc. Spoken Dialogue Systems, Vigsø (paper - 4 pages, Postscript 951kb).

Carlson, R. (1996):"The Dialog Component in the Waxholm System", Proc. Twente Workshop on Language Technology (TWLT11) Dialogue Management in Natural Language Systems, University of Twente, the Netherlands (paper - 10 pages, Postscript 597kb).

Carlson R. and Hunnicutt S. (1996):"Generic and domain-specific aspects of the Waxholm NLP and Dialog modules". Proc of ICSLP-96, 4th Intl. Conference on Spoken Language

Processing, Philadelphia, USA, Oct. 3-6, 1996 (paper - 4 pages, Postscript 111kb).

Carlson R. and Hunnicutt S. (1995):"The natural language component - STINA". STL-QPSR 1/1995, pp. 29-48.

Personal communication with Rolf Carlson.