Usability Issues in Spoken Language Dialogue Systems

Laila Dybkjær and Niels Ole Bernsen
Natural Interactive Systems Laboratory, University of Southern Denmark
Science Park 10, 5230 Odense M, Denmark
laila@nis.sdu.dk, nob@nis.sdu.dk

Abstract

Whilst spoken language dialogue systems (SLDSs) technology has made good progress in recent years, the issue of SLDS usability is still lagging behind both theoretically and in actual SLDS development and evaluation. However, as more products reach the market and competition intensifies, there is growing recognition of the importance of systematically understanding the factors which must be taken into account in order to optimise SLDS usability. Ideally, this understanding should be comprehensive, i.e. include all major human factors perspectives on SLDSs, and exhaustive, i.e. describe each perspective as it pertains to the detailed development and evaluation of any possible SLDS. This paper addresses the requirement of comprehensiveness by decomposing the complex space of SLDS usability best practice into eleven issues which should be considered by developers during specification, design, development and evaluation. The discussion of each issue is aimed to support the developer in building SLDSs which are likely to generate user satisfaction, which are perceived to be easy to understand and control, and which enable smooth user-system interaction. Based on the best practice issues discussed, criteria for evaluating SLDS usability are proposed. Several limits to our current understanding of SLDS usability are highlighted.

1. Introduction

Spoken language dialogue systems (SLDSs) have been on the market for about a decade and have an even longer history in research laboratories. An SLDS is an interactive system which conducts spoken dialogue with its users. To do so, the SLDS must not only recognise what the user says but also interpret it and generate an appropriate spoken response. Even the simplest kind of SLDS needs, at least, *speech recognition* in order to match the acoustic input signal to its vocabulary of words or phrases, *dialogue management* in order to decide what to do in view of the input message received, and *output generation* in order to produce a spoken response to the user. This includes the ability to produce an appropriate response when the user's input is not being recognised, or, in more complex SLDSs, when the recognised input does not make sense in context. Thus, more complex SLDSs need *language understanding* in order to understand what the recognised input means in context.

The first commercial SLDSs were simple speaker-independent, single word recognition systems with a vocabulary of a few words (Bossemeyer and Schwab 1991, Franco 1993). Even very simple SLDSs which merely understand the words "yes" and "no" and their equivalents in a speaker-independent way can do useful tasks, such as automating collect call telephone services. Today's commercial SLDSs span a much wider range. There is now a rich variety of small-vocabulary systems, cf. the examples at http://www.speechworks.com/demos/demos.htm and http://www.nuance.com/index.htma?SCREEN=demos. The most advanced commercial systems have a vocabulary of several thousand words; understand speaker-independent spontaneous speech; do complex linguistic processing of the user's input; handle shifts in initiative; have quite complex dialogue management abilities including, e.g., reasoning based on the user's input, consultation of the recorded history of the dialogue so far, and graceful degradation of the dialogue when faced with users who are difficult to understand; carry out linguistic processing of the output to be generated; solve several tasks and not just one; and robustly carry out medium-length dialogues to provide the user with, for instance, train timetable information on the departures and arrivals of trains between hundreds of cities. Examples of advanced commercial systems are described in , e.g. (Aust, Oerder, Seide and Steinbiss 1995, Peng and Vital 1996). The following is an English translation of a dialogue with the Philips system.

S1: Hello. You are connected to a spoken dialogue system prototype. This is the automatic train timetable information. You may inquire about train connections to one thousand German stations. From where to where would you like to go?
U1: From Siegen to Cologne.
S2: When would you like to go from Siegen to Cologne?
U2: On 29th June.
S3: At which time would you like to leave on Saturday 29th June?
U3: At nine o'clock.
S4: So you would like to leave at nine o'clock in the morning?
U4: Yes.
S5: There is the following connection with express train 3312 from Siegen at 9:09, arrival in Cologne at 10:39. Would you like to have the connection repeated?
U5: No.
S6: Would you like another connection?
U6: No.
S7: Then I wish you a pleasant trip.

Figure 1. A dialogue with the Philips train information system. U is user and S is system.

A huge amount of work has been invested in the component technologies that are integral to SLDSs. Major results of these efforts are that speech recognisers and synthesisers can now be bought as off-the-shelf software. The language processing and dialogue management parts of SLDS technology are less mature but platforms are emerging which facilitate the building of relatively simple language processing and dialogue management modules for SLDSs. Examples of such platforms and environments are the CSLU Toolkit (http://cslu.cse.ogi.edu/toolkit/) which is freely available, SpeechMania (http://www.speech.philips.com/ud/get/Pages/05fr.htm) from Philips and NLSA (http://www.unisys.com/marketplace/nlu/) from Unisys both of which are commercial software. Multimodal SLDSs are gaining ground. Such systems process user input other than, and in addition to, speech, such as lip movements, facial expression or gesture, and/or generate output in addition to speech, such as animated lips and faces, gesture, written text, images, non-speech sound etc. (Benoit, Martin, Pelachaud, Schomaker and Suhm 2000).

Whilst SLDSs *technology* has made good progress — even if much still remains to be done — the issue of SLDS *usability* remains an area with as many questions as there are solid answers. Far less resources have been invested in human factors for SLDSs than in SLDS component technologies. There has been surprisingly little research in important user-related issues, such as user reactions to SLDSs in the field, users' linguistic behaviour, or the main factors which determine overall user satisfaction. Similarly, human factors have often been neglected in SLDS development and evaluation. However, there seems to be growing recognition that usability is as important as, and partly independent of, the technical quality of any SLDS component and that quality human factors constitute an important competitive parameter.

In general terms, a usable SLDS must satisfy user needs which are similar to those which must be satisfied by other interactive systems. Thus, what users need are SLDSs with which they are generally satisfied in the overall context of use and which they feel are easy to understand and interact with. Interaction should be smooth rather than bumpy and error-prone, and the user should feel in control throughout the dialogue with the system. It is the task of the SLDS developer to meet those user needs considered as overall human factors design goals. However, SLDSs are very different from more traditional interactive systems whose human factors aspects have been investigated for decades, such as systems controlled through graphical user interfaces involving screen, keyboard and mouse. Perhaps the most important difference is that speech is perceptually transient rather than static. This means that the user must pick up the *output* information provided by the system the moment it is being provided or else miss it altogether. It also means that the user has no way of inspecting the interface prior to interaction. If the interface is not self-evident all the way through the dialogue it must be learnt by trial-and-error through repeated interaction, which is bad news for the casual walk-up-and-use user. Secondly, the processing (recognition, language understanding, dialogue management) of spoken *input* remains difficult to design and error-prone in execution, which is why SLDSs must be crafted with extreme care to ensure that users do not produce spoken input which the system is incapable of handling.

Building on human factors results achieved in the European DISC project (http://www.disc2.dk) on best practice in the development and evaluation of SLDSs (Failenschmid, Williams, Dybkjær and Bernsen 1999) we propose that, for the time being, the overall design goals for creating usable interactive SLDSs mentioned in the preceding paragraph may be systematically pursued by focusing on a comprehensive set of eleven usability issues which include all major human factors perspectives on SLDSs. Each best practice issue provides a focal point for optimising SLDS usability during development and evaluation. Some of the issues may appear rather obvious, such as the need for quality speech recognition. Given the comprehensiveness requirement, however, quality speech recognition will have to be included among the best practice issues as a matter of course. Partial obviousness notwithstanding, we are not aware of similar attempts to systematically describe the usability issues which must be faced by today's SLDS developers. In fact, DISC found that SLDS development projects tend to select and apply but a small subset of the evaluation criteria to be described in Section 2.11 below (Heid, Bernsen, Dybkjær and van Kuppevelt 1998).

To focus the paper within the space available, we focus upon *walk-up-and-use SLDSs for shared-goal tasks.* Walk-up-and-use SLDSs should require no prior training of their users and therefore impose the strictest usability requirements on their developers. By contrast, mastery of SLDSs for routine or professional use may be acquired through trial-and-error and written documentation even if those systems have relatively poor usability characteristics. However, if users cannot use walk-up-and-use SLDSs more or less immediately, they are not likely to come back to try again. SLDSs for shared-goal tasks, or task-oriented SLDSs, may be contrasted with conversational SLDSs. The former are built to help users perform one or several particular and well-circumscribed tasks on the assumption that user and system have the unique and shared goal of accomplishing the task as efficiently as possible. Tasks include, e.g., reserving concert tickets, doing home banking over the telephone, getting travel information, accessing email over the phone, getting connected to the right person in a company, obtaining information on car insurance, and providing information on the amount of water and electricity used during a certain period. In fact, all existing commercial systems are shared-goal systems. Real conversational systems which are able to conduct meaningful dialogue with their users without being constrained by the shared goal and common task assumptions, remain research challenges so far. Within the limitations just noted, the best practice issues described below would seem to apply up to and including systems of a complexity well beyond current commercial applications (cf. Figure 1). The increasing use of speech in multimodal and natural interactivity contexts is only partially addressed (in Section 2.2). Finally, in what follows, the *user* is the system end-user who interacts with an SLDS in order to carry out a particular task, and not, for instance, the system developer, the system maintainer or the SLDS deployer.

2. Human factors best practice issues for SLDSs

The eleven human factors best practice issues for SLDSs to be presented are aimed to carve the complex space of SLDS usability into intuitively satisfactory and complementary segments. The issues are otherwise of several different types. Two issues, i.e. 2.1 on specification and 2.11 on evaluation, concern particular aspects of the SLDS development and evaluation process itself. These issues are orthogonal to the rest of the best practice issues and include the latter within their scope. Most issues address requirements on particular SLDS components, including 2.3 on speech recognition, 2.4 on user input language, 2.5 on output speech, 2.6 on output language, and 2.7, 2.8., 2.9 and 2.10 on various aspects of interaction optimisation wrt. which the dialogue manager has a central role. Finally, 2.2 on when to use speech highlights the fact that speech is not always the right choice of modality for interactive systems.

Even if the best practice issues are hypothetically claimed to cover all major human factors perspectives on SLDSs, their presentation below is far from being exhaustive in the sense of describing each issue as it pertains to the detailed development and evaluation of any possible SLDS. For one thing, too much is already known about some of the conditions for SLDS usability. All we can do about that within the limitations of this paper is to explain and exemplify the issues involved and refer the reader to more extensive presentations elsewhere, for instance on the DISC SLDSs Best Practice website (http://www.disc2.dk). Typically, those presentations elaborate and further illustrate how to address a particular issue. Secondly, there are still many unknowns in the area of human factors for SLDSs, some of which are pointed out below.

2.1 Full specification of human factors

It is essential to gather knowledge about the needs, expectations, behaviour (linguistic and otherwise) and environment of the intended users of an SLDS. This must be done as early as possible in the specification and design process by involving representative users from the target user group(s). The collected information contributes to deciding which human factors requirements to include in the requirements specification. Useful sources for information collection are, e.g., observation of human-human communication on the same task whenever possible, site visits which help understand organisational and collaborative requirements, and interviews with prospective users. Given the fact that SLDSs must be carefully crafted to fit their users, lack of early user involvement could easily lead to a demand for substantial redesign later or, even worse, to a final system which the intended users do not want to use. How to involve the users from early on is well described in the literature, see, e.g. (Bernsen, Dybkjær and Dybkjær 1998, Williams and Cheepen 1998) and the DISC Best Practice website (http://www.disc2.dk). Some best practice advice on user involvement from an evaluation point of view is provided in Section 2.11.

Important human factors to consider at an early stage include, i.a., ease of use (for instance, you don't have to remember particular keywords and procedures), the capability of the system to perform a dialogue which is natural (just express yourself to get the task done), flexible (you don't need a system conformant plan to get the task done) and robust (if something goes wrong the system helps you getting back on track) within its domain, availability of sufficient meta-communication facilities (it is easy to correct mistakes), sufficient task domain coverage (the system shares your conception of the task), awareness of the needs of different user groups, contextual factors in the user's organisation and in the deployment of the SLDS. In fact, all of the best practice issues presented in Sections 2.2 through 2.11 should be considered in the specification phase as regards their relevance for the system to be designed. Issue relevance should be made operational by spelling out the implications for the SLDS's behaviour and by including those implications in the specification. For instance, given an on-line database of air flight schedules and fares from a particular airline company, it has now become relatively straightforward to build SLDSs enabling users to book extremely complex round-trips. However, some users will want to use a different carrier; others will be prepared to revise their time schedule to obtain reduced fares; and yet others will want to know if other carriers could provide a simpler or faster itinerary. If such preferences are not discovered and taken into account from early on, most users are likely to find the SLDS untrustworthy and likely to offer overly expensive fares and unnecessarily complex itineraries.

Among the human factors listed in the preceding paragraph, some of the most difficult usability issues concern contextual adequacy, i.e. adequacy of the full set of *contextual* factors which contribute to making an SLDS acceptable to its users. These factors remain insufficiently explored both as regards which they are and as regards their individual contributions to user satisfaction. It is possible that contextual factors, such as service improvements or economical benefits, are among the most important factors influencing users' satisfaction with SLDSs. *Service improvement* may consist in longer opening hours so that users can call the service around the clock and on week-ends rather than during normal office hours only. Another service improvement may be the introduction of a brand new service which makes life easier because, e.g., customers can call the system instead of having to go to some far-away office. Also, the fact that SLDSs can help avoid phone queues to overloaded human agents is likely to be felt as an important improvement of efficiency. And SLDSs replacing voice response systems may make life easier for some mobile phone users who do not have to switch the phone back and forth between their ears and their eyes, as well as for people having rotation dial phones which cannot be used with touch-tone systems. *Economical benefits* will usually also be positively received by users. Users may, e.g., pay less for a service if they use an SLDS instead of calling a person, or the SLDS may be set up on a free phone number.

Once the relevant and operationalised human factors have been included in the requirements specification, their feasibility must be subjected to early evaluation together with all other parts of the specification. 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 included in the requirements specification are reasonable, sufficient, feasible and non-contradictory. For example, it is important to assess whether the resources available in the project in terms of time, money, man-power and expertise, are sufficient to develop the system as specified, and whether the specified system is likely to satisfy actual user needs and preferences. If the requirements specification is not properly evaluated and discrepancies turn up later between technical feasibility and usability requirements, this may result in re-specification and redesign, or, even worse, in an SLDS in which insufficient hacks damage the system's usability.

All evaluation criteria to be used in evaluating the final system, including those relating to human factors, should be included in the requirements specification. The evaluation criteria state the parameters which will be measured or otherwise evaluated and the results that should be achieved for the final system to be acceptable. For the time being, there is no agreed set of human factors evaluation criteria (cf. Section 2.11). Nevertheless, the definition, from early on during development, of clear, relevant and appropriate evaluation criteria, and the continuous and methodologically sound evaluation of progress with reference to those criteria, is crucial to good development practice. If the human factors evaluation criteria become invented and applied as the project proceeds and no decisions are made up front concerning which criteria the final system or component must satisfy, developers have little support in determining whether they are on the right track when observing the emerging behaviours of the system and its components.

2.2 Speech is the right thing

The large majority of task-oriented spoken language dialogue systems use speech-only in their interaction with the users. This is changing, however, and we will be seeing an increasing number of systems which combine spoken human-system dialogue with other modalities for information representation and exchange. Examples are: speech input and output combined with graphics output (Wyard, Appleby, Kaneen, Williams and Preston 1995), combined speech and pen input (Oviatt 1997), combined speech and mouse pointing gesture input (Roth, Chuah, Kerpedjiev, Kolojejchick and Lucas 1997). More examples can be found in (Benoit et al. 2000). For instance, static graphics output is useful for rendering information which is too lengthy for being presented through speech, such as long lists of flight connections, or which is virtually impossible to present through speech, such as the detailed contents of images. It is a well-known fact that speech-only interaction is not appropriate for all tasks and applications, and the same is true for any particular modality combination which includes speech input and speech output for spoken dialogue with the users. To mention a simple example, few users would be happy if they had to speak aloud their pin code to the bank teller machine in the street.

In other words, before embarking on an SLDS development project developers should attempt to make sure that spoken input and output, possibly combined with other input/output modalities, is an appropriate modality choice for the planned application. If the chosen modalities are inappropriate, chances are that the users either will not accept the application or will refrain from using some of the modalities it offers.

There are several approaches to the issue of modality choice. One approach to ensuring an appropriate choice of modalities is to use common sense. Common sense, however, is far from being infallible. The reason is the sheer complexity of the problem of modality choice. One study found that the choice of whether or not to use spoken interaction depends on how the following variables are instantiated by the planned application: *generic task* (e.g. text editing), *speech act* (e.g. alarm), *user group* (e.g. the blind), *interaction mode* (e.g. wireless device), *work environment* (e.g. public spaces), *generic system* (e.g. ATMs), *performance parameter* (e.g. speed, efficiency), *learning parameter* (e.g. no learning overhead) and *cognitive property* (e.g. reduction in visual workload) (Bernsen 1997). Change the instantiation of one of these variables, and the modalities to be used in the application may change completely. Common sense is not always a good predictor of the complex interrelationships involved. A second approach is scientific experimentation which often serves to complement and correct common sense. However, because of the complexity of the modality choice problem, developers cannot count on the existence of experimental results which can be unambiguously applied to the design cases at hand. And it is often difficult or impossible to carry out elaborate controlled experiments as part of application development projects. A third option is for the developer to consult a recently developed experimental design support tool called SMALTO which supports early design reasoning about whether or not to use speech. The tool was developed in DISC, cf. (http://www.disc2.dk/tools, Bernsen and Luz 1999). SMALTO is based on the observation that a limited number of modality properties, such as that "speech is omnidirectional", have proved to provide powerful support to reasoning about modality appropriateness. SMALTO interactively shows how those modality properties work in evaluating particular claims about the use of speech in a large variety of applications. The modality properties we currently use can be found at the DISC SLDSs Best Practice website.

2.3 Good speech recognition

From the user's point of view, good speech recognition means that the system rarely gets the user's spoken input wrong or fails to recognise what the user just said. Recognition success, as perceived by the user, not only depends on recogniser quality but also on how other parts of the SLDS handle the user's input. Good recogniser quality nevertheless remains the key factor in making users confident that the system will successfully get what they say.

Walk-up-and-use systems are likely to encounter a large variety of users who, moreover, may address the system in highly different environments. The speech recogniser, therefore, and depending on more specific information on its intended users and environments of interaction, must be trained to recognise a variety of dialects and accents, speakers of different gender, age and voice quality, speaking with a low or a loud voice, in noisy or quiet environments, and with varying channel quality. All or most recognisers must cater for both female and male voices, and sometimes for children's voices as well. As to the environment, an over-the-phone taxi ordering application, for instance, is likely to demand more robustness to background noise than an application which will be addressed most of the time by users speaking from their office. In many cases, speech from different speakers may overlap. The telephone line itself holds omnipresent and time-variant noise as well. Additional sources of variation are the way in which users hold the speaker relative to the mouth and the extra-linguistic noise speakers may produce, such as tongue clicks (Waibel 1996, Baggia, Gerbino, Giachin and Rullent 1994). Based on information on the target user population and their environments of interaction, the early design specification should make clear which of the above issues the recogniser must be able to handle.

It has often been observed that SLDSs do not need perfect speech recognition because they benefit from one of the fundamental characteristics of human dialogue, i.e. that errors committed may automatically become the subject of the dialogue itself until they have been corrected. Moreover, using language models and other constraints, many SLDSs are able to remove false recogniser hypotheses about what the user just said. This is why, say, 82% recognition success can lead to +90% transaction success for the system as a whole. Despite the important points made by these observations, it remains true that too many misrecognitions may cause users to turn down the system. If users are misunderstood or not understood by the system they may initiate meta-communication to correct the problem (see 2.10). However, they may also simply repeat the utterance which the system did not recognise, or they may rephrase their input in a more complex manner, thereby making matters worse. The fact that the system has some form of meta-communication capability will not necessarily solve the problem. Moreover, meta-communication comes at a price, as we shall see.

It remains a non-trivial task to optimise speech recognition for the application at hand. Adequate information on users and environments is essential input to the selection and creation of training data. To assess the quality of the system's recognition capabilities prior to running the full system, speech recognition accuracy may be tested on the recogniser with users from the target group(s).

2.4 Natural user speech

Speaking to an SLDS should feel as easy and natural as possible. It does not help the user that the system's speech recognition is perfect in principle if the input vocabulary and grammar expected from the user are not the ones which the user is likely to use and thus cannot be understood. This makes the question of which kind of input language the system should accept and understand, a very important one. The point is *not* that all but fully conversational SLDSs are unnatural. Rather, what constitutes natural user speech is a relative, not an absolute notion. Depending on, i.a., the task and users' experience, what is "natural" input language may vary considerably. In principle, an SLDS only needs to be able to discriminate if the user does or does not produce a sound in response to a system question. For most tasks, however, such 'grunt detect' systems will be perceived as highly unnatural. Moreover, it is hardly feasible, for instance, to build a train time-table information system this way. This system at least needs to be able to recognise train stations, dates, times and indications as to whether the customer wants to go to, or depart from, a particular station. In addition, many tasks require that users have at least some of the dialogue initiative and sometimes most of it as in, e.g., email and calendar handling (see 2.8). Whenever the user has the initiative, longer utterances are to be expected compared to answers to closed system questions (see 2.6). In other words, the task imposes certain minimal constraints on how simplistic the user's input language could be.

What is being experienced as natural input speech is also highly relative to the system's output phrasing. For example, if a flight ticket reservation system asks: "Where does the journey start?", some users might naturally reply by providing the name of the town or suburb in which they live rather than the name of an airport, which is likely to constitute a problem for the recogniser. Re-phrasing of the system's question might remove the problem. In the same way, lengthy and/or overly polite system utterances are likely to invite similar linguistic behaviour on the part of the user, thereby burdening input recognition and understanding unnecessarily. The system's output language thus can (indeed, should) be used to control users' input language so that the latter becomes manageable for the system whilst still feeling natural to the user (cf. 2.6). If the minimal constraints imposed by the task are satisfied and the system's output language adequately controls the user's input language, users may well feel that the dialogue is natural even if they are not inclined to engage in lengthy conversation.

Analysis of data from system simulations, questionnaires and interviews is a useful tool for obtaining information on users' input language and on what they perceive as being natural input language.

2.5 Good output voice quality

From the user's point of view, good SLDS output voice quality means that the system's speech is clear and intelligible, does not demand an extra listening effort, is not particularly noise sensitive or distorted by clicks and other extraneous sounds, has natural intonation and prosody, uses an appropriate speaking rate, and is pleasant to listen to (Karlsson 1999). Taken together, these requirements are difficult to meet today.

Types of output speech

There are three main types of output speech: recordings of entire system utterances, concatenation of recorded words and phrases, and synthesised speech.

The highest and most human-like output voice quality is obtained by playing recordings of entire system utterances. In most cases, however, it is practically impossible to record all the utterances which the system may have to produce, such as all possible utterances concerning bank account statements. An alternative is to record and store utterance parts, i.e. words and phrases, and concatenate the parts on-line to produce the needed output utterances. The price paid for this gain in flexibility is that it is very difficult to get intonation completely right in concatenated speech because the intonation of a particular word changes according to that word's position in a sentence. It is usually too cumbersome to record all possible intonations and, even if this were to be done, difficult to control the correct concatenation in context. Moreover, it is difficult to make the 'seams' between the concatenated parts completely continuous and avoid any kind of distortion. These are the reasons why concatenated speech tends to exhibit more or less unnatural prosody. Still, concatenated speech can be made clear, intelligible and reasonably pleasant to listen to. However, a problem common to all forms of recorded speech is that the person whose voice has been recorded should be available at all times for new recordings, should these become necessary! It can easily be impossible to find a second voice which is indistinguishable from the original. This is a distinct disadvantage particularly in systems whose output repertoire needs regular updating, such as events information SLDSs.

The most flexible way of generating output speech is through text-to-speech (TTS). TTS makes it easy to add new words and phrases because no recordings of a particular speaker are involved. Although steady progress is being made, TTS quality remains modest with regard to clarity, intelligibility, prosody and pleasantness. Moreover, TTS systems tend to have a series of particular limitations, such as in the pronunciation of proper names or phrases borrowed from other languages.

Concatenated speech is the most frequently used type of speech in today's SLDSs. For walk-up-and-use systems in particular, TTS may simply be too difficult to understand for infrequent users. Even if, inevitably, the lack of perfection of today's output speech technology may adversely affect users' evaluation of the system, there is one advantage. It is that too natural output speech, such as in recordings of entire system utterances, may suggest to users that the system is far more capable and human-like than it actually is, encouraging them to address the system in a way which is more conversational and talkative than it can handle.

Type of voice

The type of output voice chosen is likely to affect users' perception of the system as a whole. In particular, and together with the quality of the speech output (cf. above), the voice type has a major influence on how pleasant users find the "system's voice". If recordings are to be used it must be decided whether or not to involve a professional speaker. Even if it may be desirable to use a professional speaker to obtain good voice quality, experiments have shown that users sometimes prefer non-professional voices. Similarly, it must be decided whether to use a male or a female voice, whether the voice should be deep or high, and what the speaking rate should be. Men appear to prefer female voices in many cases whereas women appear to be more indifferent. A slow speaking rate may be useful for interaction with novices or in noisy environments whilst a faster speaking rate may be appropriate for experienced users. Many issues concerning voice type are still open. For example, how will the system's ability to change speaking style or express emotion through prosody affect the interaction with users? These are becoming real issues since some TTS systems already have those abilities.

In order to gather input on user preferences with respect to the system's output voice, representative users of the system under development may be asked to listen to different "system voices" and provide feedback on which one they prefer and what they like and dislike about each of them.

2.6 Adequate output phrasing

SLDSs speak to their users about many different topics in order to advance the task, provide feedback, engage in meta-communication, welcome the user or say goodbye, etc. Regardless of the topic, the system should express itself co-operatively in order to maximise the likelihood that the task is achieved as smoothly and efficiently as possible. To facilitate successful interaction, the system's output should be correct, relevant and sufficiently informative without being over-informative. Expressions should be clear and unambiguous, and language and, as far as possible, terminology should be consistent and familiar to the user (Bernsen et al. 1998). Failure to provide adequate (or co-operative) output phrasing may generate many different kinds of miscommunication some of which could lead to interaction failure. A tool based on Cooperativity Theory, CODIAL, has been developed in DISC to support the design of co-operative system dialogue (Dybkjær 1999, http://www.disc2.dk/tools). The purpose of CODIAL is to help prevent system miscommunication during early dialogue design. The main ideas are explained in this and the following sections. For more details the interested reader is referred to the DISC SLDSs Best Practice website. In addition to co-operativity, this section discusses input control which was briefly mentioned in Section 2.4.

Contents

It is crucial that the user can trust what the system says. Users have good reason for dissatisfaction if the system provides false information on, e.g., departure times, prices or meeting venues. Still, this may happen, for instance if the database is not being properly updated.

Lack of relevance of system output caused by, e.g., misrecognition, will typically lead to meta-communication dialogue. The system's utterance may be perfectly relevant given its interpretation of what the user said but totally irrelevant given what the user actually said. If, for instance, a user has asked for information on train connections between Ulm and Stuttgart and the system replies: "When do you want to go from Ulm to Coburg?", then obviously the user will initiate some kind of repair, such as "No, no, not to Coburg, to Stuttgart.", which many systems will have difficulty coping with.

System output should be sufficiently informative. Otherwise, misunderstandings may occur which are only detected much later during interaction, if at all, or which, at best, lead to immediate requests for clarification by the user. A typical example of information insufficiency is a system which, when asked for a discount ticket on a certain departure on which no discount is available, merely replies that discount is not possible without offering the user a normal fare for the same departure. Conversely, the system should not provide too much or overly verbose information. In response, users may either become inattentive or try to take the dialogue initiative in ways which the system cannot handle. And even attentive users are likely to forget lengthy messages. Also, users may become confused about the discourse focus and what is the point of relevance, initiating clarification meta-communication as a result.

Form and language

System output must be clear and unambiguous. Unclarity naturally leads to uncertainty and need for clarification. So does ambiguity *if* detected by the user. If undetected, as often happens, the effects of ambiguity can be severe. If the user unknowingly selects a non-intended meaning of a word or phrase uttered by the system, all sorts of things can go wrong. An example of multiple ambiguity is a system which says that “Flight 658 will arrive at 9”. As long as the user does not know if this is am or pm and does not know if this is arrival according to the regular flight plan or expected actual arrival time, the user would be advised to ask a few question before acting on the information provided. To help avoid ambiguity it is, moreover, advisable to use the same expressions for the same purposes throughout the dialogue.

Users prefer the system to speak their native language in most applications. Just as importantly, the system preferably should not use terms and expressions which are not familiar to most or all of its users. If the system must do that, unfamiliar terminology should be explained either proactively (before users ask) or through adequate measures for clarification meta-communication. For instance, a car sales information system should be prepared to handle the clarification meta-communication that will inevitably result (cf. 2.10).

Input control

It is important to realise that the system’s output language tends to have a massive priming effect on the user’s language. Humans are extremely good at adapting (automatically, unconsciously) their vocabulary, grammar and style to those of their dialogue partner, even if the partner happens to be an SLDS (Amalberti, Carbonell and Falzon 1993, Gustafson, Larsson, Carlson and Hellman 1997, Zoltan-Ford 1991). It is, therefore, crucial that the words and grammar used in system output can be recognised and understood by the system itself. Similarly, the system should have a speaking style which induces users to provide input that is to the point and can be handled by the system.

Closed system questions impose strong input language control. For instance, the system’s question “Please state your city of departure” is likely to elicit a one-word answer such as “Copenhagen”. “When do you want to leave” is a slightly more open question which will probably elicit longer user input, such as an utterance containing a date, a date and a time, or a time. An entirely open question, such as “How may I help you”, may invite any kind of user input much of which may be difficult to handle for current SLDSs.

It may be desirable to use a fairly restrictive kind of output language in error handling situations (cf. 2.10) and when interacting with novice users. For instance, relatively closed system questions or listings of options may often serve to guide novice users through the dialogue in a natural way. This is relatively easy to do for well-structured tasks where user and system share a model of the sub-tasks to be addressed and possibly in which order the sub-tasks should be dealt with. Ill-structured tasks pose more challenges but may, e.g., be handled via menu-like structures (cf. 2.8). To cater also for expert users who know exactly which input the system needs, the system may accept not only an answer to the specific question it asks but other pieces of information as well. For instance, when interacting with a train information system which asks “Where do you want to travel from”, it should be possible not only to answer this question but also to provide more complete information, such as “I want to go from Copenhagen to Odense tomorrow morning around eight o’clock”. This will speed up the interaction for experienced users while still providing guidance to novice users.

Depending on the task and the system’s output language, natural user input language may include complex phenomena, such as cross-sentence co-reference, ellipsis, discontinuous user input involving large gaps in the sequence of dialogue acts expected by the system, and indirect dialogue acts. Any such phenomenon, when constituting a natural and frequently occurring part of the input, has to be either handled by the system in a satisfactory way or eliminated through redesign of the system’s output.

Exactly how terse or how polite the system’s output style should be depends on several factors. Terse system output will encourage users to use a terse style which is easier for the system to handle than a style which is lengthy and conversational. Furthermore, terseness speeds up task performance and often appears to promote user satisfaction, in particular in case of repeated use of the system. Politeness phrases may in such contexts be perceived as superfluous and contributing to making the dialogue long-winded, see e.g. (Williams and Cheepen 1998). However, depending on, i.a., the users (first-time users or expert users), their culture, and the organisation which owns the system (and which may want a friendly and polite system), some amount of politeness may be desirable.

Lack of co-operativity in the system's output may be diagnosed from the occurrence of communication problems in simulated or real user-system interaction. Data capture and analysis is costly, however, especially because large amounts of data may be needed for triggering most of the communication problems which the system is likely to cause. To reduce cost, and to help identify those kinds of lack of cooperativity which are less likely to cause communication problems, CODIAL may be used both for walk-throughs through the interaction design prior to data capture and for the actual data analysis. Interaction data analysis is needed to assess the efficiency of the input control strategies adopted. User contacts through interviews and questionnaires are good means for obtaining early input on how users experience the system's output.

2.7 Adequate feedback

Adequate feedback is essential for users to feel in control during interaction. The user must feel confident that the system has understood the information input in the way it was intended, and the user must be told which actions the system has taken and what the system is currently doing. Only by being told can the user take corrective action when needed. Moreover, *telling* the user is not always good enough – the user must be told in such a way that the user *notices* what the system says. It follows that it may be a good thing for SLDSs to provide several different kinds of feedback to their users. It also follows that the task of ensuring adequate feedback can be a difficult one. We distinguish between process feedback and information feedback.

Process feedback

When the system processes information received from the user and hence may not be speaking for a while, process feedback keeps the user informed on what is going on. Many SLDSs may benefit from offering this kind of feedback. A user who is uncertain about what the system is doing, if anything, is liable to produce unwanted input or to believe that the system has crashed and decide to hang up. Moreover, the uncertainty itself is likely to affect negatively the user's satisfaction with the system. Process feedback can be provided in many different ways. Right now, the field is in a state of experimentation in which different kinds of process feedback are being tested. The best process feedback needs not be spoken words or phrases describing what the system is doing but could consist in grunts or ehm's, tones, melodies, or appropriate earcons.

Information feedback

Feedback on the system's understanding of what the user just said helps ensure that, throughout the dialogue, the user is left in no doubt as to what the system has understood. The same effect of building the user's trust in the system is produced by feedback on the actions taken by the system, particularly if those actions cannot be perceived by the user, such as when a money transfer has been made from one account to another. All SLDSs therefore need to provide information feedback. A user who is uncertain as to what the system has understood, or done, is liable to produce unwanted input and to react negatively to the way the system works.

Information feedback can be provided in different ways. In many cases, the system may simply carry out the action requested by the user in a way which the user can perceive, such as reading aloud a requested voice mail, thus demonstrating that it has understood the user's input. This way of providing feedback is not always possible. Some actions cannot be perceived by users during interaction with the system, as in the money transfer example above. Also, several pieces of information may be required from the user (e.g. when booking a flight ticket). As long as the system does not have all the information it needs, it must ask for the missing parts before taking further action. Moreover, even when the system does have the information it needs for taking a certain action, such as providing information about a train departure, it may not be clear to the user that the system actually is talking about the train the user wants unless adequate feedback is provided. In such cases the system should make the user aware of what it has understood. The difficulty lies in finding the best way to do this. Consider the dialogue snippets in Figure 2. The system misrecognises 'Hamburg' as 'Hanover'. However, the user has no chance of spotting the error from the answer shown in S2a. A better solution is the answer shown in S2b in which the system explicitly mentions the city names and the date it has understood. This offers the carefully listening user the possibility of detecting the error and initiating meta-communication in order to correct the mistake (U2b). However, such *implicit* information feedback does not always work because users may not listen carefully enough. If the intended users turn out to be prone to ignore the implicit feedback, the more burdensome *explicit* feedback strategy shown in S2c might be considered. Experience has shown that the strategy illustrated in S2c is more robust than the strategy illustrated in S2b (Sturm, den Os and Boves 1999). Explicit feedback does not come for free, however. The price to pay for using explicit feedback is that user and system have to spend more dialogue turns to solve the task.

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

S2a: 5.35 AM.
S2b: The first train from Frankfurt to Hanover on 3rd May 1999 leaves at 5.35 AM.
S2c: You want to go from Frankfurt to Hanover tomorrow morning?

U2a: Many thanks. Goodbye.
U2b+c: [Initiates correction.]
...

Figure 2. Different ways of providing information feedback. U is user and S is system.

The amount and nature of the information feedback the system should provide also depends on factors such as the cost and risk involved in the user-system transaction. Obviously, feedback on bank transfers or travel bookings are more critical than feedback on which email the system should be reading to the user next. Even travel information, if the user gets it wrong, can have serious consequences for that user. Current opinion on information feedback during transactions of medium to high significance probably is that the system developer should prefer the safer among the two most relevant feedback options. Building the user's trust and confidence in the system is more important to user satisfaction than reducing the average number of turns needed to complete the transaction.

For important transactions, an additional safeguard is to give the user a summary of the agreed transaction at the end of the dialogue, preceded by a request that the user listens to it carefully. If the request is not there, the user who has already ignored crucial feedback once, may do so again.

2.8 Adequate system interaction

An SLDS may handle one or several tasks, and tasks may be well-structured or ill-structured. In general, the system should make the user understand clearly which task(s) the system can carry out and how they are structured, accessed and addressed. To support natural interaction, an SLDS needs a reasonable choice of dialogue initiative, an appropriate dialogue structure, sufficient task and domain coverage, and sufficient reasoning capabilities.

Dialogue initiative and structure

Spoken human-human dialogue is prototypically mixed-initiative, the partners in dialogue negotiating and exchanging dialogue initiative as they go along. In fact, however, many task-oriented dialogues tend to be directed primarily by one of the interlocutors. This fact can be exploited when designing human-system interaction. Users may even feel satisfied with less initiative when interacting with an SLDS than when talking to a person as long as the dialogue initiative distribution fits the task(s) the system and the user must solve together, and provided that the rest of the best practice issues proposed in this paper are properly attended to. Thus, *system directed dialogue* can work well for tasks in which the system simply requires a series of specific pieces of information from the user, in particular if the user is new to the system. The robust way to do this is for the system to ask for one piece of information at a time until the task has been completed. The price to pay is that more experienced users are likely to miss the opportunity for providing all the necessary pieces of information in a single utterance. To satisfy experienced users, the system may have to be able to cope with the larger packages of input information which are natural to these users. In a similar way, the rigid system directed menu-based approach, which is often used when several unrelated tasks are available, may be softened by, e.g., the introduction of pseudo sub-menus. Pseudo sub-menus enable the experienced user to access all functions directly by speaking the right command without having to be guided through the series of sub-menus which are available to the inexperienced user.

In principle, a (mainly) *user directed dialogue* is as much of an aberration from mixed initiative dialogue as is the (mainly) system directed dialogue. Currently, user directed dialogue would seem to be appropriate primarily for applications designed for experienced users who know how to make themselves understood by the system. Unless supported by screen graphics or other additional modalities, inexperienced users are likely to address the system in ways it cannot cope with.

Mixed initiative dialogue, i.e. a mixture of system and user initiative, is often both desirable and technically feasible. At some points in the dialogue it may be appropriate that the system takes the initiative to guide the user, obtain missing information, or handle an error. At other points, such as when the user needs information from the system, is already familiar with the system or wants to correct an error, it is appropriate for the user to take the initiative.

As long as we cannot build fully conversational systems, dialogue designers may have to impose some kind of structure onto the dialogue, determining which topics (or sub-tasks) could be addressed when. It is important that the structure imposed on the dialogue is natural to the user, reflecting the user's intuitive expectations, especially in system directed dialogue in which the user is not supposed to interfere with the dialogue structure. Unnatural dialogue structure will often cause users to try to take the initiative in ways which the system cannot cope with.

Task and domain coverage

Sufficient task and domain coverage is also crucial to natural interaction. Even if unfamiliar with SLDSs, users normally have rather detailed expectations to the information or service which they should be able to obtain from the system. It is important that the system meet these expectations. If, for some reason, the system is not able to perform a certain sub-task which users would expect the system to handle, this has to be stated clearly. Even then, user satisfaction is likely to suffer. For instance, if two people want to travel together on a roundtrip, it is standard for human travel agents to book for both of them in parallel, making sure that they get adjacent seats on all legs of the itinerary. Users are therefore likely to expect the system to be able to do just that. If the system has been designed to book for one person at a time, users must be told explicitly that this is the way the system works. And they may not like the considerable amount of extra dialogue turns they have to go through in order to book what to them is a simple twosome journey.

Reasoning

Contextually adequate reasoning is a classical problem in the design of natural interaction. Even when users have been appropriately primed to expect a rather primitive interlocutor, they tend to assume that the system is able to perform the bits and pieces of reasoning which humans are able to do without thinking and which are inseparable parts of natural dialogue about the task. Typically, therefore, SLDSs must incorporate both facts and inferences about the task as well as general world knowledge in order to act as adequate interlocutors. If, for instance, the task has a temporal dimension, the system must be able to infer which date the user is talking about when saying "tomorrow" or "on Friday". Defining which kinds of reasoning the system must be capable of is part and parcel of defining the system's task and domain coverage and subject to similarly difficult decisions on task delimitation (cf. the preceding paragraph). For instance, whereas it may be obvious that the system should be able to define an absolute date based on the user's "tomorrow", it may be less obvious that the system should be able to do the same for "Christmas Day", "Pentecost" or even "The opening day of Wimbledon".

It is possible to get rough ideas on initiative distribution, users' models of the task, and how to delimit the domain from studying recorded human-human dialogues on tasks similar to those which the system is intended to cover. However, the recordings should only be considered possible starting points. In particular, as task complexity grows, developers are likely to find themselves forced to adopt more restrictive task delimitations and impose a more rigid dialogue structure than those which they found in the human-human dialogues. Having done that, the resulting interaction model needs early testing and evaluation. In particular, if the developer is into relatively high task complexity compared to the state of the art, early testing is strongly recommended (see 2.11).

2.9 Sufficient interaction guidance

Sufficient interaction guidance is essential for users to feel in control during interaction. Interaction guidance can be particularly hard to get right in speech-only, walk-up-and-use SLDSs. Speech is inappropriate for providing lengthy and complex "user manual" instructions up front for first-time users (cf. 2.6 on over-informativeness and verbosity). Moreover, at any given time some users will already be familiar with the system whereas others will be novices. Issues to consider include cues for turn-taking vs. barge-in; the background and experience of the target users; help facilities; and highlighting of non-obvious system behaviour, such as that the system does not listen when it speaks, needs particularly reduced forms of user input, or handles the task in some non-standard manner.

Cues for turn-taking and barge-in

Barge-in or talk-over means that users can interrupt the system whenever they wish and still expect to be recognised and understood, even when the system is speaking or is processing recent input. Barge-in allows the user to speed up the interaction, for instance by interrupting already familiar instruction prompts in order to get on with the task. It is known, however, that many users do not interrupt the system even when they know they can do so. On the other hand, people are used to taking an unfilled pause as a cue to start speaking. Thus, if the system does not allow barge-in, it must provide clear cues for turn-taking, making it completely clear to the user when to speak and when to refrain from speaking because the system does not listen. Cues can be explicit, such as the up-front instruction "Please speak after the tone" followed by a tone each time it is the user's turn to speak, or implicit, such as when the system stops talking.

A major problem is the silence which may occur when the system starts processing the user's input (cf. 2.7 on process feedback). This silence could be taken to indicate that, e.g., the system did not get what was said or that the system needs additional information, and that the user should start speaking again. If the system is still listening whilst processing the previous user input, the user's new input may cause problems for the dialogue manager which has to generate an appropriate response to disjoint pieces of user input. And if the system is not listening any more, important input could be lost in cases when users do not merely repeat themselves.

User background and experience

It is useful to distinguish between four types of user: system expert/domain expert, system expert/domain novice, system novice/domain expert and system novice/domain novice. An SLDS needs not support all four groups, of course. If the target user group is domain and system experts only, then, obviously, the system is not a walk-up-and-use system. In that case, the developer may be able to impose strict task performance order, a relatively large number of mandatory command keywords, and ample use of written user instructions. If the primary target group is system novice users, on-line instructions and other help information is likely to be needed. This need tends to increase even further when the system novices are also domain novices who need explanation of domain technicalities, such as what is a "green departure".

Given the relative simplicity of current SLDSs, walk-up-and-use users may quickly become (system) experts. This means that interaction should be supported and facilitated for both system novices and system experts. Special shortcuts for expert interaction can be a good solution. Such shortcuts include introductions which can be skipped easily through barge-in or explicit de-selection, pseudo sub-menus (cf. 2.8), and progressive help mechanisms which are only being provided when needed. An example of progressive help is that a prompt for experts is followed by progressive help for the inexperienced user. The system may say, e.g., "Which service?" followed by "The services available are …" in case the user does not start speaking right after the first prompt.

Help and other kinds of guidance

General and explicit instructions on what the system can and cannot do and how to interact with it may be provided in a spoken introduction which can be repeated on request or be skipped by experienced users. In fact, most speech-only SLDSs strictly need some up-front introduction to guide interaction. We already mentioned the when-(not)-to-speak issue above. Just as importantly, the system should be perfectly clear about the task(s) which the user can accomplish through interaction. For instance, users build very different expectations from being told (a) "Flight information, how may I help you?" and (b) "This service provides information on British Airways domestic flight departures and arrivals. How may I help you?"

The first few words uttered by the system should not express instructions. Rather, the system might say, e.g., "Hello, you are connected to a spoken dialogue system …" in order to leave users just enough time to realise that they are connected to the right service and are speaking to a system. The longer the spoken introduction itself, the less likely it is that the user will remember and be able to follow the instructions provided. Moreover, some instructions may not be feasible for users at all, such as to remember to use a series of particular command keywords in order to navigate the system. If the instructions needed by the walk-up-and-use user are too many to be presented in the system's introduction, some of them may be relocated for presentation at particular points during interaction and only when needed. This eliminates the burden of having to memorise instructions which are provided long before they are needed, if they are needed at all for some variety of the task.

Providing useful help mechanisms is a difficult interaction design task. Help may be an implicit part of the dialogue, such as the progressive help mentioned above; be available on request by saying “help”; or be automatically enabled if the user is having problems repeatedly, for instance in being recognised. In this case the system may, e.g., propose how to express input or inform the user on what can be said.

Hardcopy instruction, such as quick reference cards, may be used to inform users on what the system can and cannot do, what its dialogue structure is, and to instruct them on how to interact with it, for instance through sample dialogues and a list of available commands. In general, this strategy for interaction guidance primarily makes sense if most users are known in advance and will be using the system repeatedly. Hardcopy instruction should never be the prime source of information because it tends to get lost, not to be at hand when needed, or be obsolete. Walk-up-and-use users often present the additional difficulty that it is impossible to know who they are and hence impossible to provide them with written hardcopy instructions in the first place.

This section has argued that barge-in is usually an advantage for the user. However, more research is needed on the potential problems caused by barge-in. SLDSs have great potential for facilitating interaction for experienced users whilst keeping the novices supported as well. A clear system introduction is normally essential to adequate novice support. It is when this introduction is insufficient that the really intricate problems of providing dynamic help begin, in particular in speech-only SLDSs. For the time being, solutions to those problems should be carefully evaluated by exposing them to interaction with representative users.

2.10 Adequate error handling

Even if the best practice issues 2.1 through 2.9 above have been taken into account carefully during specification, design and implementation, the SLDS and its users will still make errors during dialogue. In human-system interaction, error *prevention* is far preferable to error *correction,* and what those best practice issues do is to help prevent errors from occurring during interaction. Humans are good at error correction during spoken dialogue, which is why most errors are handled seamlessly in shared-goal human-human communication. Also in this respect, however, current SLDSs are far inferior to their human interlocutors. This is why adequate error handling remains a difficult issue in SLDS development. Intuitively, this issue can be decomposed along two dimensions: (a) either the system initiates error-handling meta-communication or the user initiates error-handling meta-communication. And (b) when error-handling meta-communication is initiated, it is either because one party has failed to hear or understand the other or because what was heard or understood is false, or it is because what was heard or understood is somehow in need of clarification. We distinguish, therefore, between *system or user initiated repair meta-communication* and *system or user initiated clarification meta-communication.*

Repair meta-communication

System-initiated repair meta-communication is needed whenever the system either did not understand or was uncertain that it understood correctly what was said, for instance due to low recognition confidence. In such cases, the system must ask for repetition, ask the user to speak louder or modify the way the input is being expressed in other specified ways, or tell the user what it did understand and ask for confirmation or correction. 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.

A common occurrence when the system has made clear that it did not (fully) understand what was said, is that the user simply repeats the utterance which caused the problem, leaving the system in exactly the same uncomprehending situation as before. In such cases, the system may either try again, choose to fall back on a human operator, close the dialogue, or, better, start graceful degradation, i.e. carry on by changing the level of interaction into a simpler one. Depending on the problem at hand and the sophistication of the system, this can be done in different ways, such as by asking focused questions, asking for re-phrasing, asking a simple yes/no question, or asking the user to spell a crucial word.

Users may simply fail to respond. Then the system should make the user aware that it is expecting their input, for instance by repeating its latest utterance.

Users may also be understood by the system to have said something which is false and hence needs to be corrected. This is often simple to do, as when the system replies “You have deleted the emails from yesterday" to the user's "Read the emails from yesterday". Sometimes the system has reason to believe that the user has misunderstood what the system said. A symptom of misunderstanding is that the user's input is meaningless in the task context. For instance, the user may be heard as responding "London" to a question about return date. A simple strategy in such cases is for the system to repeat the question. In the – rather blatant - example just given, most users can be expected to correct themselves on being asked the same question again. This strategy will not always work, however. For instance, if the system only discovers a user's misunderstanding later in the dialogue, more elaborate recovery strategies are likely to be needed, such as back-tracking to the point where the misunderstanding occurred. Also, some systems would not consider the user's "London" as a misunderstanding but rather as a topic shift. On that assumption and depending on the task history, the system might respond, e.g. "You want to travel from London?".

Just as all SLDSs need a strategy for recovering from falsehood and from failure to hear or understand what the user just said, all SLDSs need a strategy for helping the user recover from falsehood and from failure to hear or understand what the system just said. *User-initiated* repair meta-communication can be designed in several different ways. Ideally, users should just initiate repair the same way they would have done in dialogue with a human. Some systems have been designed to allow that, but with varying success, the problem being that users may express their corrections in many different ways (Carlson 1996). Other systems require the user to use specifically designed keywords for this purpose, such as "Repeat" and "Correct" (Bernsen et al. 1998). Keywords are simpler for the system to handle than unrestricted user speech. The problem is that using keywords for correction is unnatural and hence difficult for the user to remember. A third approach is the "eraser" principle. For instance, if the system through misrecognition gets "Frankfurt to Hanover" instead of, as the user said, "Frankfurt to Hamburg", the user simply has to repeat "Frankfurt to Hamburg" until the system has received the message (Aust et al. 1995). Whilst this solution may work well for low-complexity tasks, it may be difficult to keep track of in high-complexity tasks. And it will not work if the system cannot recognise input on any sub-task all the time but only on a selected subset.

A simple case is when the user detects that the system did not hear anything. It is often sufficient for the user to repeat the input, possibly a bit louder, because if the system does not hear anything this will typically be because the user spoke while the system did not listen or because the user did not speak loudly enough relative to the microphone.

It sometimes happens that users change their minds during the dialogue with the system. In practice, these cases are similar to cases in which the system has misunderstood the user.

Clarification meta-communication

Very roughly speaking, clarification meta-communication is more difficult to design for than repair meta-communication, and user-initiated clarification meta-communication is more difficult to design for than system-initiated clarification meta-communication. Some exceptions to these rules are that it can be hard to design for user misunderstandings (see above) and that it can sometimes be relatively straightforward to anticipate and design for user clarification needs (see below). Clarification is typically of the form "I hear what you say, but what, exactly, do you mean?"

System-initiated clarification meta-communication is needed when the user's input is inconsistent, ambiguous or underspecified. In such cases, the system must ask for clarification, for instance by pointing out that the inconsistent expression "Thursday 9th" may be either "Thursday 8th" or "Friday 9th", asking whether the ambiguous "9 o'clock" is am or pm, or asking at which time of day the user wants to leave to the underspecified "I want to depart on Tuesday.".

User-initiated clarification meta-communication is needed whenever the system produces inconsistent or ambiguous utterances, or uses terms with which the user is not familiar. 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, handling such questions is difficult for SLDSs and the system developers might not have discovered all the potential problems in the first place. If they had, they could have tried to prevent all or most of the problems from occurring through adequate output phrasing or other means. As argued in 2.6 above, smooth dialogue requires that all ambiguities, inconsistencies and, in most SLDSs, terms unknown to users are avoided rather than having to be clarified on the user's initiative. There are exceptions, however. Due to the nature of their domain, some tasks inherently require facilities for clarifying the terminology used. For instance, when interacting with a used cars information system, some users will necessarily be wondering what the system is talking about as soon as it mentions ABS brakes or on-board GPS systems. It is not a practical option for the system to explain all of those domain terms as it goes along. This would be intolerable for the users who are familiar with the domain.

Most SLDSs need abilities for handling system- and user-initiated repair, and many SLDSs need system-initiated clarification abilities. We have described a series of repair and clarification mechanisms above. Even if these mechanisms are in some sense general, i.e. independent of particular domains, tasks and users, there is no simple decision procedure for deciding which of them to include in a particular SLDS. Their generality notwithstanding, sensible decisions very much depend on factors such as domain, task complexity, user population and peculiarities of user behaviour which can only be discovered through interaction data analysis.

2.11 Sufficient and timely evaluation of human factors

Human factors evaluation is necessary for measuring progress towards the human factors goals which the system has to meet. Central issues in human factors evaluation include: when to evaluate, the type of evaluation to use, the purpose of evaluation, the nature of the system version undergoing evaluation (e.g. mock-up, simulation, implemented system), what to evaluate, which and how many users to involve, and how to do the evaluation.

Types and purpose of evaluation

Evaluation can be quantitative or qualitative, subjective or objective. *Quantitative evaluation* consists in quantifying some parameter through an independently meaningful number, percentage etc. which in principle allows comparison across systems. *Qualitative evaluation* consists in estimating or judging some parameter by reference to expert standards and rules. *Subjective evaluation* consists in judging some parameter by reference to users' opinions. *Objective evaluation* produces subject-independent parameter assessment. Ideally, we would like to obtain quantitative and objective progress evaluation scores for usability which can be objectively compared to scores obtained from evaluation of other SLDSs. This is what has been attempted in the PARADISE framework based on the claim that task success and dialogue cost are potentially relevant contributors to user satisfaction (Walker, Litman, Kamm and Abella 1997). However, many important human factors issues cannot be subjected to quantification and objective expert evaluation is sometimes highly uncertain or non-existent.

The purpose of evaluation may be to detect and analyse design and implementation errors (diagnostic evaluation), measure SLDS performance in terms of a set of quantitative and/or qualitative parameters (performance evaluation), or evaluate how well the system fits its purpose and meets actual user needs and expectations (adequacy evaluation), cf. (Hirschmann and Thompson 1996, Gibbon, Moore. and Winski 1997, Bernsen et al. 1998). The latter purpose is the more important one from a human factors point of view although the others are relevant as well. Which type of evaluation to use and for which purpose, depends on the evaluation criterion which is being applied (see below). Other general references to natural language systems evaluation are (EAGLES 1996, Gaizauskas 1997, Sparck Jones and Galliers 1996).

When to evaluate and methods to use

Usability evaluation should start as early as possible and continue throughout development. In general, the earlier design errors are being identified, the easier and cheaper it is to correct them. Different methods of evaluation may have to be applied for evaluating a particular parameter depending on the phase in the lifecycle in which evaluation takes place. Early design evaluation can be based on mock-up experiments with users and on design walk-throughs. Wizard of Oz simulations with representative task scenarios can provide valuable evaluation data. When the system has been implemented, controlled scenario-based tests with representative users and field tests can be used. Recorded dialogues with the (simulated) system should be carefully analysed for indications that the users have problems or expectations which exceed the capabilities of the system. Human-system interaction data should be complemented by interviews and questionnaires to enable assessment of user satisfaction. If users are interacting with the prototype on the basis of scenarios, there are at least two issues to be aware of. Firstly, scenarios should be designed to avoid priming the users on how to interact with the system. Secondly, sub-tasks covered by the scenarios will not necessarily be representative of the sub-tasks which real users (not using scenarios) would expect the system to cover.

The final test of the system is often called the acceptance test. It involves real users and must satisfy the evaluation criteria defined as part of the requirements specification (cf. 2.1).

User involvement

In general, representative users from the target user group(s) should be involved in evaluation from early on. The developers themselves can certainly discover many of the usability problems with the early design and implementation, especially when supported by state-of-the-art usability standards, evaluation criteria and design support tools. The problem is that they know too well how to interact with the system in order to avoid creating interaction problems which the system cannot handle. For the time being, there is no alternative to involving the target users in all or most system evaluation phases and for most usability evaluation purposes. This is costly and complex to do. However, the data analysis which is crucial to benefiting from trials with the system, is as necessary after trials with developers as it is after trials with representative users. Even the early involvement of representative users is no guarantee that the system will ultimately produce sufficient user satisfaction. For one thing, the data distribution they generate may not match the behaviour of the users of the system, once installed. For another, experimental user trials are different from real situations of use in which time, money and trust are really at stake. For these reasons, and particularly when introducing SLDSs which are innovative in some respect, it is necessary to prepare and budget for field trials with the implemented system as well as for the subsequent data analysis and fine-tuning of the system. Users who are "only" involved in a test can be much more indifferent to, or more positive towards, a system with poor usability characteristics than real users who have something to loose if the system lets them down (Bernsen et al. 1998).

What to evaluate

As remarked earlier, there is at present no consensus as to which human factors evaluation criteria to use. However, the best practice issues discussed in the present paper may serve to generate a comprehensive list of usability evaluation criteria which would appear mandatory for evaluating the usability of all or most SLDSs. Even if not all of the criteria below are included in the requirements specification, they are still useful for evaluating how usable the system is and what progress is being made during its development. The evaluation criteria we propose are:

1. Modality appropriateness
2. Input recognition adequacy
3. Naturalness of user speech relative to the task(s) including coverage of user vocabulary and grammar
4. Output voice quality
5. Output phrasing adequacy
6. Feedback adequacy
7. Adequacy of dialogue initiative relative to the task(s)
8. Naturalness of the dialogue structure relative to the task(s)
9. Sufficiency of task and domain coverage
10. Sufficiency of the system's reasoning capabilities
11. Sufficiency of interaction guidance (information about system capabilities, limitations and operations)
12. Error handling adequacy
13. Sufficiency of adaptation to user differences
14. Number of interaction problems (Bernsen et al. 1998)
15. User satisfaction

The developers' options wrt. all criteria except 14 were discussed under the best practice issues above. Criterion 14 refers to the cooperativity guidelines which form the basis of CODIAL introduced in Section 2.6. Most criteria are qualitative. Several are subjective or include subjective judgement when no expert consensus can be found in the state-of-the-art. In particular, user satisfaction is subjective throughout. Nevertheless, in view of how much remains to be discovered about how the behaviour of SLDSs affect the satisfaction of their users, subjective evaluation remains a cornerstone in SLDS evaluation. Space does not permit discussion of user questionnaires and interviews. General references are (Anastasi 1988, Miller 1984, Ericsson and Simon 1985).

How to evaluate

Evaluation, including usability evaluation, is non-trivial and cannot be explained simply by stating what to evaluate (cf. the list of evaluation criteria above) and what the developers' options are (the bulk of this paper). One of the most difficult questions in evaluation probably is how to do it properly. In DISC we have developed a template which supports consistent and detailed description of each evaluation criterion. The template includes the following issues: what is being evaluated (e.g. feedback adequacy), the system part evaluated (e.g. the dialogue manager), type of evaluation (e.g. qualitative), method(s) of evaluation (e.g. controlled user experiments), symptoms to look for (e.g. user clarification questions), life cycle phase(s) (e.g. simulation), importance of evaluation (e.g. crucial), difficulty of evaluation (e.g. easy), cost of evaluation (e.g. expensive), and support tools (e.g. SMALTO), see (http://www.disc2.dk/tools). The idea is that the combined set of (i) design options for SLDS usability, (ii) human factors evaluation criteria, and (iii) template-based characterisation of each criterion, will provide developers with sufficient information for proper evaluation of their SLDSs.

3. Conclusion

In this paper, we have attempted to provide a brief best practice overview of what to consider when specifying, designing, developing and evaluating usable spoken language dialogue systems. We have argued that the DISC approach to best practice in the development and evaluation of SLDSs is on the right track towards developing a comprehensive understanding of SLDS usability, i.e. to start from a thorough description of the issues which are faced by today's developers and the solutions they might consider, followed by a listing of the evaluation criteria they should apply together with a guide to practical evaluation. Within this framework, many issues remain unresolved or even unaddressed. Deployment usability issues are still poorly understood as are the usability issues arising from multimodal and natural interactive applications which integrate speech-only SLDSs into larger systems. Usability questionnaire design remains poorly understood. The same applies to cultural differences in the perception of SLDS usability. Finally, we are aware that, even though issues to do with SLDS user adaptation have been discussed in Sections 2.6, 2.8 and 2.9 above, we have not addressed the issue of user profiles which is of particular importance to non-walk-up-and-use systems.

References

Anastasi, A. 1988. *Psychological testing*. New York, Macmillan.

Amalberti, R., Carbonell, N. and Falzon, P. 1993. User representations of computer systems in human-computer speech interaction. *International Journal of Man-Machine Studies* 38: 547-566.

Aust, H., Oerder, M., Seide, F. and Steinbiss, V. 1995. The Philips automatic train timetable information system. *Speech Communication* 17: 249-262.

Baggia, P., Gerbino, E., Giachin, E. and Rullent, C. 1994. Spontaneous speech phenomena in naive-user interactions. *Proceedings of TWLT8*, 8th Twente Workshop on Speech and Language Engineering, Enschede, The Netherlands, 37-45.

Benoit, C., Martin, J. C, Pelachaud, C., Schomaker, L. and Suhm, B.: Audio-Visual and Multimodal Speech Systems. Article to appear in Gibbon, D., Moore, R. and Winski, R. (Ed.). 2000. *Handbook of Standards and Resources for Spoken Language Systems,* 2nd Edition. Mouton de Gruyter, Berlin, New York.

Bernsen, N. O. 1997. Towards a tool for predicting speech functionality. *Speech Communication* 23: 181-210.

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

Bernsen, N. O. and Luz, S. 1999. SMALTO: Speech functionality advisory tool. *DISC Deliverable D2.9.* http://www.disc2.dk/tools.

Bossemeyer, R. W. and Schwab, E. C. 1991. Automated alternate billing services at Ameritech: Speech recognition and the human interface. *Speech Technology Magazine* 5, 3, 24-30.

Carlson, R. 1996. The dialogue component in the Waxholm system. *Proceedings of TWLT11*, 11th Twente Workshop on Dialogue Management in Natural Language Systems, Enschede, The Netherlands, 209-218.

Cole, R. A., Mariani, J., Uszkoreit, H., Zaenen, A. and Zue, V. W. (Editorial Board), Varile, G. and Zampolli, A. (Managing Editors). 1996. *Survey of the State of the Art in Human Language Technology.* Sponsors: National Science Foundation, Directorate XIII-E of the Commission of the European Communities, Center for Spoken Language Understanding, Oregon Graduate Institute. URL: http://www.cse.ogi.edu/CSLU/HLTsurvey/.

Dybkjær, L. 1999. CODIAL, a tool in support of cooperative dialogue design. *DISC Deliverable D2.8.* http://www.disc2.dk/tools.

EAGLES. 1996. Evaluation of Natural Language Processing Systems. Final Report, EAGLES Document EAG-EWG-PR2. Copenhagen, Center for Sprogteknologi.

Ericsson, K. and Simon, H. 1985. Verbal reports as data. *Psychological Review*, 67, 215-251.

Failenschmid, K., Williams, D., Dybkjær, L. and Bernsen, N. O. 1999. Draft proposal on best practice methods and procedures in human factors. *DISC Deliverable D3.6.* http://www.disc2.dk.

Franco, V. 1993. Automation of operator services at AT&T. *Proceedings of Voice'93,* San Diego.

Gaizauskas, R. (Ed.) 1997. *Proceedings of the SALT Club Workshop on Evaluation in Speech and Language Technology,* Sheffield.

Gibbon, D., Moore, R. and Winski, R. (Eds.) 1997. *Handbook of standards and resources for spoken language systems.* Mouton de Gruyter, Berlin, New York.

Gustafson, J., Larsson, A., Carlson, R. and Hellman, K. 1997. How do system questions influence lexical choices in user answers. *Proceedings of EuroSpeech'97*, Rhodes, 2275-2278.

Heid, U., Bernsen, N. O., Dybkjær, L. and van Kuppevelt, J. 1998. Current practice in the development and evaluation of spoken language dialogue systems. *DISC Deliverable D1.8.* http://www.disc2.dk.

Hirschmann, L. and Thompson, H. S. 1996. Overview of evaluation in speech and natural language processing. In Cole et al. 1996, Section 13.1.

Karlsson, I. 1999. Draft proposal on best practice methods and procedures in speech generation. *DISC Deliverable D3.3.* http://www.disc2.dk

Miller, G. 1984. *Experimental Design and Statistics*. London, Methuen.

Nielsen, J. 1993. *Usability engineering.* New York, Academic Press.

Oviatt, S. 1997. Multimodal interactive maps: Designing for human performance. *Human-Computer Interaction*, Vol.12, No. 1&2: 93-129.

Peng, C. and Vital, F. 1996. Der sprechende Fahrplan. *Output* 10, 92-96.

Roth, S. F., Chuah, M. C., Kerpedjiev, S., Kolojejchick, J. and Lucas, P. 1997. Towards an information visualization workspace: Combining multiple means of expression. *Human-Computer Interaction*, Vol.12, No. 1&2: 131-185.

Sparck Jones, K. and Galliers, J. 1996. Evaluating natural language processing systems. Lecture Notes in Artificial Intelligence 1083. Berlin, Springer.

Sturm, J., den Os, E. and Boves, L. 1999. Issues in spoken dialogue systems: Experiences with the Dutch ARISE system. *Proceedings of ESCA Workshop on Interactive Dialogue in Multi-Modal Systems,* Kloster Irsee, Germany, 1-4.

Waibel, A. 1996. Interactive translation of conversational speech. *IEEE Computer*, 41-48.

Walker, M. A., Litman, D. J., Kamm, C. A. and Abella, A. 1997. Evaluating interactive dialogue systems: Extending component evaluation to integrated system evaluation. *Proceedings of the ACL/EACL Workshop on Spoken Dialog Systems*, Madrid, 1-8.

Williams, D. M. L. and Cheepen, C. 1998. Just speak naturally: Design for naturalness in automated spoken dialogues. *Proceedings of CHI 98*, ACM Press, 243-244.

Wyard, P., Appleby, S., Kaneen, E., Williams, S. and Preston, K. 1995. A combined speech and visual interface to the BT business catalogue. *Proceedings of the ESCA Workshop on Spoken Dialogue Systems*, Vigsø, Denmark, 165-168.

Zoltan-Ford, E. 1991. How to get people to say and type what computers can understand. *Int. Journal of Man-Machine Studies* (34), Academic Press Ltd, 527-547.