Usability Evaluation 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

The paper first addresses a series of issues basic to evaluating the usability of spoken language dialogue systems, including types and purpose of evaluation, when to evaluate and which methods to use, user involvement, how to evaluate and what to evaluate. We then go on to present and discuss a comprehensive set of usability evaluation criteria for spoken language dialogue systems.

1 Introduction

Usability is becoming an increasingly important issue in the development and evaluation of spoken language dialogue systems (SLDSs). Many companies would pay large amounts to know exactly which features make SLDSs attractive to users and how to evaluate whether their system has these features. In spite of its key importance far less resources have been invested in the usability aspect of SLDSs over the years than in SLDS component technologies. The usability aspect has often been neglected in SLDS development and evaluation and 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. However, there now 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 usability constitutes an important competitive parameter.

Most of today's SLDSs are walk-up-and-use systems for shared-goal tasks. Not least in walk-up-and-use systems is usability of utmost importance. Users of such systems cannot be offered to attend a course to learn about the system or be asked to read the user manual before starting to use the system or in case of problems during interaction. Help must be available online and needed as little as at all possible.

There is at present no systematic understanding of which factors must be taken into account to optimise SLDS usability and thus also no consensus as to which usability evaluation criteria to use. Ideally, such an understanding should be comprehensive, i.e. include all major usability 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 aspect of comprehensiveness by proposing a set of usability evaluation criteria. The criteria are derived from a set of usability issues that have resulted from a decomposition of the complex space of SLDS usability best practice.

In the following we first briefly address types and purpose of evaluation (Section 2), when to evaluate and which methods to use (Section 3), user involvement (Section 4), and how to evaluate (Section 5). Section 6 presents the proposed set of evaluation criteria and discusses the usability issues behind these. Section 7 concludes the paper.

2 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 usability 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 usability 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).

3 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.

4 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).

5 How to evaluate

Evaluation, including usability evaluation, is non-trivial and cannot be explained simply by stating what to evaluate (cf. Section 6) and what the developers' options are. One of the most difficult questions in evaluation probably is how to do it properly. We have developed a template which supports consistent and detailed description of each evaluation criterion (Bernsen and Dybkjær 2000). 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 (www.disc2.dk/tools). The idea is that the combined set of (i) design options for SLDS usability, (ii) usability evaluation criteria, and (iii) template-based characterisation of each criterion, will provide developers with sufficient information for proper evaluation of their SLDSs.

6 What to evaluate

In general terms, a usable SLDS must satisfy user needs which are similar to those which must be satisfied by other interactive systems. The SLDS must be easy to understand and interact with. Interaction should be smooth 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 usability design goals. However, SLDSs are very different from more traditional interactive systems whose usability 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. Moreover, 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 unsatisfactory 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.

In the following we present a set of usability evaluation criteria which are based on results achieved in the European DISC project (www.disc2.dk) on best practice in the development and evaluation of SLDSs (Failenschmid et al. 1999). Our claim is that quality usability of SLDSs may be pursued by focusing on a comprehensive set of 15 usability issues which include all major usability perspectives on SLDSs. These usability issues - or a subset thereof, depending on how advanced and complex the system is to be – should be represented in the system specification and should therefore also be reflected in a set of evaluation criteria for the system which would appear mandatory for evaluating the usability of the SLDS. More details on the usability issues can be found in (Dybkjær and Bernsen 2000).

6.1 Modality appropriateness

The majority of task-oriented SLDSs have so far used speech as the only input/output modality. However, an increasing number of systems now combine spoken input/output with other modalities.

It is well-known 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. Few users would e.g. be happy to speak aloud their pin code to the bank teller machine in the street. The 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. Common sense, experimentation and/or the use of the tool SMALTO (www.disc2.dk/tools) may help the developers in making the right modality choice.

6.2 Input recognition adequacy

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 may be used by many different users 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.

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).

6.3 Naturalness of 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. Depending on, e.g., the task and users' experience, what is "natural" input language may vary considerably.

What is being experienced as natural input speech is also highly relative to the system's output phrasing. For example, lengthy and/or overly polite system utterances are likely to invite similar linguistic user behaviour, thereby burdening input recognition and understanding unnecessarily. The system's output language thus should be used to control users' input language so that the latter becomes manageable for the system whilst still feeling natural to the user. 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.

6.4 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.

There are three main types of output speech: recordings of entire system utterances, concatenation of recorded words and phrases, and text-to-speech (TTS). 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 while full recordings are much too inflexible. Moreover, too natural output speech, like full recordings, 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.

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, the voice type has a major influence on how pleasant users find the "system's voice". Voice type includes features

such as male/female, deep/high voice, speaking rate, and emotions.

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.

6.5 Output phrasing adequacy

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 contents of the system's output should be correct, relevant and sufficiently informative without being over-informative. Users have good reason for dissatisfaction if the system provides false information, e.g., 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. 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. Conversely, the system should not provide too much or overly verbose information. Users may then e.g. become inattentive, try to take the dialogue initiative, or become confused and initiate clarification meta-communication.

The form of system 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). 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. To help avoid ambiguity it is, moreover, advisable to use the same expressions for the same purposes throughout the dialogue. 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.

Developers may use CODIAL - a tool based on Cooperativity Theory – as support for the design and evaluation of co-operative system dialogue (www.disc2.dk/tools).

It is important to realise that the system's output language tends to have a massive priming effect on the user's language. 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.

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.

6.6 Feedback adequacy

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. A difficult thing is that *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 may therefore be a good thing for SLDSs to provide several different kinds of feedback to their users. We distinguish between process feedback and information feedback.

When the system processes information received from the user and hence may not be speaking for a while, process feedback – which may be provided in many different ways - keeps the user informed on what is going on. 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.

Feedback on the system's understanding of what the user just said and on the actions taken by the system helps ensure that, throughout the dialogue, the user is left in no doubt as to what the system has understood and is doing.

Information feedback can be provided in different ways and more or less explicitly. The amount and nature of the information feedback depends e.g. on the cost and risk involved in the user-system transaction. 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.

6.7 Adequacy of dialogue initiative

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.

Spoken human-human dialogue is prototypically mixed-initiative. However, many task-oriented dialogues tend to be directed primarily by one of the interlocutors. 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. 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 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.

6.8 Naturalness of the dialogue structure

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.

6.9 Sufficiency of 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.

6.10 Sufficiency of the system's reasoning capabilities

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. 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.

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.

6.11 Sufficiency of 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. 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; help facilities; and highlighting of non-obvious system behaviour.

Barge-in allows the user to speed up the interaction, e.g. by interrupting already familiar instruction prompts. 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 or implicit. If the user starts speaking while the system is still listening but 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.

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. The introduction should not be too long because then users cannot remember the instructions. Moreover, the instructions must be feasible for the user. 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.

Providing useful help mechanisms is a difficult interaction design task. Help may be an implicit part of the dialogue, 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.

Sufficiency of interaction guidance should be carefully evaluated by exposing the SLDS to interaction with representative users.

6.12 Error handling adequacy

Even if the best practice issues discussed so far 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. Also as regards error handling 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.*

System-initiated repair meta-communication is needed whenever the system either did not understand or was uncertain that it understood correctly what was said. 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. In case of a repeated misunderstanding the system may either choose to fall back on a human operator, close the dialogue, or, better, start graceful degradation, i.e. change the level of interaction into a simpler one. If users simply fail to respond, then the system should tell that it is expecting their input. Users may also be understood by the system to have said something which is false and hence needs to be corrected. *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, but since users may express their corrections in many different ways this is very difficult. Some systems require the user to use specifically designed keywords. 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 where the user simply repeats his input until the system has received the message. 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.

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. *System-initiated* clarification 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 an expression is inconsistent. *User-initiated* clarification is needed whenever the system produces inconsistent or ambiguous utterances, or uses terms with which the user is not familiar. Unfortunately, handling user clarification 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. Due to the nature of their domain, some tasks inherently require facilities for clarifying the terminology used because it may not be a practical option for e.g. a car sales system to explain all domain terms as it goes along.

Most SLDSs need abilities for handling system- and user-initiated repair, and many SLDSs need system-initiated clarification abilities. There is no simple decision procedure for deciding which mechanisms to include in a particular SLDS. 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.

6.13 Sufficiency of adaptation to user differences

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 and thus falls outside the group of SLDSs considered in this paper. 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.

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 e.g. introductions which can be skipped easily through barge-in or explicit de-selection.

6.14 Number of interaction problems

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.

6.15 User satisfaction

Objectively measured quality, technical and otherwise, does have an impact on user satisfaction, but this is far from being the whole story. User satisfaction is inherently subjective, building on personal preferences and contextual factors. Unfortunately, some of the most difficult usability issues exactly 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.

Much still remains to be discovered about how the behaviour of SLDSs affect the satisfaction of their users. Therefore, subjective evaluation remains a cornerstone in SLDS evaluation. User questionnaires and interviews remain core tools for gathering information on user satisfaction.

7 Conclusion

We have presented a brief guide to practical evaluation of SLDSs followed by a set of usability evaluation criteria. 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.

References

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 Dybkjær, L. 2000. A Methodology for Evaluating Spoken Language Dialogue Systems and Their Components. *Proceedings of LREC 2000*, Athens, May 2000, 183-188.

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. and Bernsen, N.O. 2000. Usability Issues in Spoken Language Dialogue Systems. In *Natural Language Engineering*, Special Issue on Best Practice in Spoken Language Dialogue System Engineering, Volume 6 Parts 3 & 4 September 2000, 243-272.

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.

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

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

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.