A Methodology for Evaluating Spoken Language Dialogue Systems and Their Components

Niels Ole Bernsen and Laila Dybkjær

Natural Interactive Systems Laboratory,
Science Park 10, 5230 Odense M, Denmark
nob@nis.sdu.dk, laila@nis.sdu.dk

Abstract

As spoken language dialogue systems (SLDSs) proliferate in the market place, the issue of SLDS evaluation has come to attract wide interest from research and industry alike. Yet it is only recently that spoken dialogue engineering researchers have come to face SLDSs evaluation in its full complexity. This paper presents results of the European DISC project concerning technical evaluation and usability evaluation of SLDSs and their components. The paper presents a methodology for complete and correct evaluation of SLDSs and components together with a generic evaluation template for describing the evaluation criteria needed.

1. Introduction

Spoken language dialogue systems (SLDSs) are, finally, proliferating on the market for a large variety of applications and in an increasing number of languages. Current commercial SLDSs constitute an application paradigm in the field of speech technologies, and that is a major step forward. It means that existing systems can be copied, ported, localised, maintained and modified to fit a range of customer and end-user needs without having to bother about fundamental innovation. This is what creates an emerging industry. Many research issues remain to be solved, however, one of which is now becoming critically important, i.e. the issue of SLDS evaluation.

Only recently have (spoken) dialogue engineering researchers begun to face systems evaluation in its full complexity (Fraser, 1995; Bernsen, Dybkjær and Dybkjær, 1998; Gilbert et al., 1999, DARPA Communicator http://fofoca.mitre.org/). Roughly, systems evaluation decomposes into (i) technical evaluation of systems and their components, (ii) usability evaluation of the system, and (iii) customer evaluation of systems and components. Although (i)-(iii) are not completely dissociated, a technically excellent system integrating excellent components may have poor usability whilst a technically secondary system may score highly in terms of user satisfaction. And the customer may prefer yet a third system for reasons of, say, cost and platform compatibility which have little to do with technical perfection or end-user satisfaction.

The European Esprit Long-Term Research project Spoken Language Dialogue Systems and Components – Best practice in development and evaluation (DISC) was launched in 1997 in order to develop a first dialogue engineering best practice methodology. In the course of DISC, it became clear that SLDS evaluation deserved even more attention and effort than originally planned. This paper describes two general results. In the terminology introduced above, DISC presently addresses technical evaluation of systems and components and usability evaluation whereas evaluation from the customer's point of view has not been fully addressed, mainly because of the scarcity of data from systems deployment available to the consortium from the literature or otherwise. Within this scope, the first result to be presented below is a methodology for *complete and correct evaluation* of SLDSs and components (Section 3). The second result is a generic *evaluation template* for describing the evaluation criteria needed in complete and correct evaluation of SLDSs and components (Section 4). Section 2 briefly presents the DISC approach and Section 5 concludes the paper.

2. Grounding of the DISC Results

This section briefly reviews the empirical basis for the DISC best practice methodology and presents the core concepts of the DISC approach to best practice methodology development. In addition, of course, to general experience and the literature, the DISC results are based on in-depth analysis of the following SLDSs and components: The French LE Arise system on telephone accessed train time-table information systems (den Os et al., 1999), the CMU Phoenix parser (Ward and Issar, 1995), the Daimler-Benz dialogue manager (Heisterkamp and McGlashan, 1996), the Daimler-Benz parser (Mecklenburg, Hanrieder and Heisterkamp, 1995), the Danish Dialogue System for flight ticket reservation (Bernsen, Dybkjær and Dybkjær, 1998), the Vocalis Operetta automated call routing system (Fraser, Salmon and Thomas, 1996), the Vocalis Voice Activated Dialling system (http://www.vocalis.com/products/speechtel/info-frame.html), the Verbmobil spoken language dialogue translation system (http://www.dfki.de/verbmobil/), and the multimodal Waxholm tourist boat information system (http://www.speech.kth.se/waxholm/waxholm.html). The common methodology followed in analysing the above *DISC exemplars* is described in (Dybkjær et al., 1998). As can be seen, some of the DISC exemplars are research prototypes whereas others are commercial. Except for Waxholm, all exemplars are, or are components in, *speech-only* SLDSs. SLDS evaluation, as discussed below, therefore only addresses evaluation of speech-only systems and components.

In the DISC approach, an SLDS has six *aspects:* speech recognition, speech generation, natural language understanding and generation, dialogue management, human factors, and systems integration. In simple systems, the natural language understanding and generation aspect may be non-existent but the five other aspects probably must be present for the system to be an

SLDS at all (even low quality human factors are human factors). From the point of view of best practice, an SLDS should be the result of (a) correct choices among the available options, technological and otherwise, within each aspect and (b) correct development (including evaluation) practice.

Based on analysis of the range of existing SLDSs and components referred to above, DISC has developed what we call a *grid* best practice analysis per aspect. Each grid defines a space of aspect-specific *issues* which the developer must, or may have to, face. When developing a dialogue manager, for instance, the developer should decide whether or not the dialogue manager should provide top-down support for input language processing. For each issue, the available *options* are laid out in the grid together with the *pros* and *cons* for choosing a particular option (cf. (a) above). In addition to the grid analyses per aspect, and again based on the DISC exemplars, DISC has developed a *life-cycle* best practice analysis per aspect, which includes recommendations on how the development process should proceed (cf. (b) above). In summary, the way DISC tackles the task of specialising software engineering best development and evaluation practice to the particular purposes of dialogue engineering, is to model an SLDS as having at most six aspects, present each aspect as a space of issues to be addressed, and describe the development process per aspect. At the time of writing, most DISC results are being made available on the DISC Best Practice Guide website (http://www.disc2.dk).

3. Evaluation Completeness and Correctness

It is clear from Section 2 that there is no way of guaranteeing that DISC has identified a complete set of issues per aspect. In fact, this is as unlikely as can be. The sophistication of speech-only SLDSs and their components will no doubt continue to develop for many years to come, leading to even more complex issue spaces than those charted in DISC. Still, one can only evaluate what is there and we hope that DISC has managed to chart at least large fractions of the aspect-specific issue spaces confronting today’s developers.

Based on the grid issue spaces, it is possible to define an aspect-specific notion of evaluation *completeness*. Suppose that, for instance, the dialogue management grid includes 24 issues for consideration by dialogue manager developers, such as which types of dialogue histories to include in a particular application. If the SLDS to be developed is a relatively simple one, not all of the 24 issues are likely to be relevant, so the developer selects options within, say, 14 of the issues and ignores the remaining issues because these are relevant only to more sophisticated dialogue managers than are presently needed. In this case, the developer must apply evaluation criteria on 14 chosen dialogue manager options in order to do a complete evaluation of the dialogue manager aspect of the application. Process and results of generating a complete set of evaluation criteria for human factors in SLDSs are presented in (Dybkjær and Bernsen, 2000).

Evaluation completeness or, more generally speaking, knowing *what* to evaluate, is not enough, however. *How* to evaluate, or evaluation *correctness*, is just as important. To follow best development practice, developers have to evaluate a chosen option at the right time(s) and in the right way(s). Thus, evaluation correctness is a matter of applying a particular evaluation criterion correctly at the right stages during the development life-cycle.

What the DISC best practice methodology aims to do, in other words, is to support the developer in (a) choosing the right options for the application at hand and (b) properly developing and evaluating an SLDS incorporating those options. Both of these aims come together in the filled evaluation templates to be used in the development of the SLDS and its components.

4. A Generic Evaluation Template

Given that the developer knows, per SLDS aspect and for the particular application at hand, what to evaluate, such as how well the dialogue manager handles error loops and graceful degradation, focus can shift to how to do the evaluation. In DISC, we have iteratively developed an evaluation template to support the ‘how’ of evaluation. The template is a model of what the developer needs to know in order to apply an evaluation criterion to a particular property of an SLDS or component, such as the noise models used by the recogniser. This knowledge is specified by the template’s ten entries which are numbered 1 through 10. Depending on the purpose of use of the template, we have developed three different versions of the template including different information, as follows. (A) The *basic template* presents and defines the ten entries. The basic template is not meant to be filled with specific information but has the role of supporting the understanding of how to use the template for evaluation purposes. (B) The *empty template* simply includes the ten entries of the DISC evaluation model. The empty template is meant to be filled with information for specifying particular evaluation criteria. Thus, an empty template has to be filled for each property, i.e. each selected option in aspect-specific issue space, to be evaluated. (C) A *filled template* specifies a particular evaluation criterion.

Figure 1 shows the empty evaluation template. Examples of filled templates can be found on the DISC web site, e.g. at (http://www.disc2.dk/slds/dm/DMevaldetail.html). The basic template is shown below.

1. What is being evaluated
2. System part evaluated
3. Type of evaluation
4. Method(s) of evaluation
5. Symptoms to look for
6. Life-cycle phase(s)
7. Importance of evaluation
8. Difficulty of evaluation
9. Cost of evaluation
10. Tools

Figure 1. The empty template.

- *1. What is being evaluated*

This entry describes the *property or properties* of an SLDS or component that is being evaluated, such as speech recognition success rate. In some cases, an evaluation criterion refers to a *generic property* which covers several different *specific properties.* Dialogue segmentation, for instance, can be done in several different ways depending on the segmentation units involved, such as user and system turns, or dialogue acts. When dealing with generic properties, the evaluators using the template will have to do the appropriate additional specifications of the specific properties which they will be evaluating.

• *2. System part evaluated*

This entry describes which component(s) of an SLDS are being evaluated, if any. This could be, e.g., the parser, the speech generation component or the system as a whole.

• *3. Type of evaluation*

This entry describes the *type* of evaluation, i.e. whether evaluation is quantitative, qualitative or subjective and whether or not evaluation is comparative. Some evaluation criteria are comparative by nature. Many others can in principle be used for comparative evaluation. It is, of course, satisfying to obtain a quantitative score from the evaluation which can be used to measure progress, and which may even be objectively compared to scores obtained from evaluation of other SLDSs. However, many important evaluation issues relating to SLDSs cannot be subjected to quantification. Note that a particular property under evaluation may be subjected to several different types of evaluation.

Terminology

Quantitative evaluation consists in counting something and producing an independently meaningful number, percentage etc. It should be noted that, even if quantitative measures may make little sense in absolute terms, i.e. as independently meaningful numbers or scores, quantitative measures can be useful for *progress evaluation* in which improvements are being measured against, e.g., a test suite. However, we would argue that quantitative progress evaluation is not ”real” quantitative evaluation as long as progress is not being measured against an independently meaningful quantitative standard or target. Independently meaningful scores are not only very important for purposes of comparative evaluation of systems and components, they are also difficult to achieve. For instance, many published speech recogniser recognition success rates suffer from under-specification in terms of factors such as recording environment, microphone quality, corpus selection, corpus size, speaker population details etc.

Qualitative evaluation consists in estimating or judging some property by reference to expert standards and rules. The standards to apply may derive from the literature, from experience or from expert consultants.

Quantitative and qualitative evaluation are both *objective evaluation*.

Subjective evaluation consists in judging some property of an SLDS or, less frequently, component by reference to users’ opinions.

Comparative evaluation consists in comparing quantitative, qualitative or subjective evaluations for different SLDSs and components. Comparative evaluation is often done *internally* in a development process in order to measure progress (progress evaluation). In most cases, this does not produce independently meaningful scores which can be used in comparisons with other SLDSs or components. An equally important but much more difficult form of comparative evaluation is comparison between different SLDSs or components. The general problem with *external comparative* evaluation of SLDSs and components is that it can be difficult to ensure evaluation under strictly identical conditions, such as same task, same test suite, same-sized user population etc. As a rule, the easier it is to ensure strictly identical conditions, the more specific is the property being evaluated. However, customers and end-users tend to be more interested in global evaluations that take into account many different properties, asking: which SLDS or component among several is globally the best one? Such evaluations are at best qualitative and often include subjective elements.

• *4. Method(s) of evaluation*

This entry describes the methods of evaluation which may be used at various stages in the life-cycle. In early design and specification, evaluation tends to be conceptual rather than based on real data. Later in the life-cycle, data capture and analysis dominate the evaluator’s activities (see 5 below).

Terminology

Design analysis consists in using experience and common sense, thinking hard when exploring the design space during the specification and design phases, doing walkthroughs of models, comparing with similar systems, browsing the literature, applying existing theory, guidelines and design support tools, if any, involving experts and future users, the procurer etc. The completeness of the requirements specification may be judged by checking whether all relevant entries in the DISC grid(s) have been considered, see (http://www.disc2.dk). Evaluation also consists in checking whether the design goals and constraints are sound, non-contradictory and feasible given the resources available. Note that design analysis can be performed at any time during the life-cycle, not only during the early design phase. For instance, a customer considering alternative offers may want to analyse the requirement specifications and design specifications of the products on offer.

Wizard of Oz data analysis consists in analysing problems posed by phenomena observed in data from simulated user-system interactions. The simulations are performed by one or several humans and address the non-implemented parts of the system. These may range from the entire system to a single sub-module, such as a fully implemented system in which only, e.g., the recogniser is switched off and replaced by a simulation. The advantage of simulations is that, if done extensively and analysed carefully, a large number of problems with design concepts and the phenomena that will be present in the deployed application can be spotted early in the development process. Their disadvantage is the cost of setting up and running several simulations, and of analysing the generated data. The perception of the SLDS or component by the users involved in the simulations can be investigated through methods such as questionnaires and interviews.

No standards exist for which questions to ask in *questionnaires and interviews*. No standards exist on how to interpret the results of questionnaires and interviews. Still, these methods can give crucial insights into the users' perception of the system.

For *questionnaires,* a standard procedure is to ask users to express their subjective perceptions of the SLDS as a series of properties on a five-point scale. Questionnaires should contain a "free-style comments" section.

Post-trial interviews are useful for capturing user observations which might otherwise have been missed and which might have implications for virtually any kind of system deficiency. Interviews are often a good complement to questionnaires.

Diagnostic evaluation is of central importance in the early development process but should require less effort in the final phase by which time most errors should have been removed. During debugging of the implemented SLDS or component, two typical types of test are glassbox tests and blackbox tests.

A *glassbox test* is a test in which the internal system representation can be inspected. The evaluator should ensure that reasonable test suites, i.e. data sets, can be constructed that will activate all loops and conditions of the program being tested.

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

Test suites are useful for evaluating one or several sub-components independently of the rest of the system. Use of test suites for component evaluation should always be accompanied by rigorous and explicit consideration of the match between the test-suite evaluation conditions and the actual operating conditions for the component in the integrated system. Any mismatch, such as lack of representativeness of the test suite data or of the acoustic signal conditions, may render the test suite evaluation results irrelevant to judging the appropriateness of the component for the task it is to perform in the integrated system. Test suites are a natural part of glass-box and black-box evaluation.

User-system interaction data analysis consists in analysis of data from the interaction between the fully implemented system and real users, either in *controlled experiments* with selected users and scenarios which they have to perform, or in *field studies* where the SLDS or component is being exposed to uncontrolled user interaction. User-system interaction data is useful or even necessary in many cases, i.e. when too little is known in advance about the phenomena that will be present in the deployed application. This data, if comprehensive, has high reliability because of deriving from a test corpus of sufficient size and realism wrt. task and user behaviour. Unfortunately, the data cannot be obtained until late in the development of the system. User-system interaction data analysis, if performed extensively rather than cursorily, is costly. This kind of analysis can be partly replaced by Wizard of Oz data analysis which is costly as well but which happens early enough in the life-cycle to enable prevention of gross errors. Since there is significant cost in both cases, cost which is only offset by corresponding risks, this is where (early) design support tools are most desirable.

• 5. *Symptoms to look for*

This entry describes the symptoms the evaluator should look for in the data. These could be, e.g., lack of understanding by the system, apparently irrelevant system responses, or user complaints in a questionnaire.

• 6. *Life-cycle phase(s)*

This entry describes the life-cycle phases in which evaluation of the property in question should be performed. In general, the earlier evaluation can start, the better. Distinction is made between early design, simulation, implementation, field evaluation, final evaluation, maintenance and porting.

Terminology

Early design includes requirements and design specification. This is the most important life-cycle phase for system and component evaluation. However difficult this may be to do in any formal way, it is essential to carry out a systematic and explicit evaluation of whether the design goals and constraints are reasonable, feasible and non-contradictory. Caught at this stage, errors due to rash design decisions will not be causing trouble later on. There is no substitute for qualitative evaluation and sound judgement during early design. This explains the importance of applied theory, guidelines and tools in support of early design.

Simulation and implementation. These are the life-cycle phases in which modules, such as the dialogue manager and its sub-modules, should be severely tested. To begin with, (part of) the SLDS or component may be simulated. The end result of this phase should be an implemented and debugged system or component which is ready for external trials. Simulation-before-implementation can be advisable in many cases, not least with respect to dialogue manager development. Applied theory and guidelines are at this stage mainly used in support of scenario and test suite development.

Field evaluation is performed by exposing the SLDS or component to uncontrolled interaction with users. Field evaluation may precede the final acceptance test.

Final evaluation may consist in an *acceptance test*, i.e. a more or less formal and controlled evaluation experiment which should decide if the SLDS meets the evaluation criteria specified as part of the requirements specification. What is primarily being evaluated is the behaviour of the system as a whole. In addition to controlled experiments, final evaluation may include design analysis and blackbox tests. The evaluation methods used during final evaluation may also be used for *customer evaluation* in which a potential customer wants to understand the positive and negative sides of an SLDS or component.

Maintenance deals with updating the SLDS or component in various ways, such as updating the database linked to the dialogue manager.

Modification deals with re-using the SLDS or component for new purposes. This includes localisation, customisation, additions and other kinds of changes.

• 7. Importance of evaluation

This entry comments on the importance of evaluating a certain property. Note that importance is a multi-faceted concept and may depend on, among other things:

- is evaluation of this property *relevant to all or only some* current systems or components?
- if the system or component has the property under consideration, *how crucial* is it to get the property right? What are the penalties?

Evaluation importance can be described as *low*, *medium* or *high* together with a statement of the reasons for the grading. Stating those reasons is important to understanding the grading proposed. For instance, it may be crucial to get some property right even if that property, such as speech acts identification, is relevant only to few current systems.

• 8. Difficulty of evaluation

This entry comments on the difficulties involved in performing the evaluation.

- the difficulty of evaluation may depend on various forms of *complexity*, such as task complexity, user input complexity, dialogue manager complexity, or overall system complexity, see (http://www.disc2.dk);
- the difficulty of evaluation may depend on the existence of *unsolved research problems*. These may be more or less severe.

• 9. Cost of evaluation

This entry comments on the costs involved in performing the evaluation.

- evaluation is more or less *costly* to perform in terms of time, manpower, or skilled labour;
- difficult evaluation may be relatively uncostly, for instance if it can be done quickly by an external expert (the problem is to find and motivate the expert); easy evaluation can be costly, for instance because of the volume of data involved;
- Wizard of Oz simulations, field studies and their associated data analysis are costly.

• 10. Tools

This entry references software tools and other kinds of support which may be of help in performing the evaluation.

In DISC, we have produced draft filled templates for a wide range of evaluation criteria of importance to the evaluation of most aspects of SLDSs. These criteria are still being refined. For several reasons, as has transpired above, an evaluation criterion represented as a filled template is not a self-explanatory construct. First, it uses but does not itself explain the terminology explained in the basic template above. Secondly, because the evaluation criterion has been generated from issues/options/pros and cons in some DISC grid, it must be interpreted by reference to that grid. This is why the hypertext structure of the DISC Best Practice Guide website has been found particularly helpful for representing the DISC results, see (http://www.disc2.dk).

5. Conclusion

In this paper we have presented the DISC approach to generating complete and correct evaluation criteria for SLDSs and components. It is probably uncontroversial that the SDLS global community could benefit strongly from having best practice standards for the development and evaluation of SLDSs and their components. The DISC Best Practice Guide website is an attempt to address this need. The website will continue to be developed and we would like to take this opportunity to invite all readers to send us their comments and criticisms on what is there or what is not there but should be. In addition to continued improvement of the DISC Best Practice Guide website according to the DISC agenda, future plans include extension to best practice for the development and evaluation of multimodal SLDSs.

6. References

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

den Os, E., Boves, L., Lamel, L. and Baggia, P. 1999. Overview of the ARISE Project. In *Proceedings of the European Conference on Speech Technology, EuroSpeech*, 1527-1530.

DARPA Communicator: http://fofoca.mitre.org/

DISC Best Practice Guide: http://www.disc2.dk

Dybkjær, L. and Bernsen, N. O., 2000. Issues in Making Usable Spoken Language Dialogue Systems. To appear in *Natural Language Engineering*.

Dybkjær, L., Bernsen, N. O., Carlson, R., Chase, L., Dahlbäck, N., Failenschmid, K., Heid, U., Heisterkamp, P., Jönsson, A., Kamp, H., Karlsson, I., Kuppevelt, J.v., Lamel, L., Paroubek, P., and Williams, D., 1998. The DISC Approach to Spoken Language Systems Development and Evaluation. In A: Rubio, N. Gallardo, R. Castro, and A: Tejada (eds.): *Proceedings of the First International Conference on Language Resources and Evaluation*, Granada, 1998. Paris: The European Language Resources Association, 185-189.

Fraser, N. M., 1995. Quality Standards for Spoken Language Dialogue Systems: A Report on Progress in EAGLES. In *Proceedings of the ESCA Conference on Spoken Dialogue Systems, Theories and Applications*, Vigsø, 157-160.

Fraser, N. M., Salmon, B. and Thomas, T., 1996. Call Routing by Name Recognition: Field Trial Results for the Operetta(TM) System. *IVTTA'96*, NJ, USA.

Gilbert, N., Cheepen, C., Failenschmid, K. and Williams, D., 1999. *Guidelines for Advanced Spoken Dialogue Design*. http://www.soc.surrey.ac.uk/research/guidelines.

Heisterkamp, P. and McGlashan, S., 1996. Units of dialogue management: an example. In *Proceedings of ICSLP'96*, Philadelphia, 200-203.

Mecklenburg, K., Hanrieder, G. and Heisterkamp, P., 1995. A Robust parser for continuous spoken language using PROLOG. In *Proceedings of Natural Language Understanding and Logic Programming 1995*, Lisbon, Portugal, 127-141.

Verbmobil: http://www.dfki.de/verbmobil/

Voice Activated Dialling: http://www.vocalis.com/-products/speechtel/infoframe.html

Ward, W. and Issar, S., 1995. The CMU ATIS System. In *Proceedings of the ARPA Workshop on Spoken Language Technology*, 249-251.

Waxholm: http://www.speech.kth.se/waxholm/waxholm.-html