Deliverable D3.10

Working Paper on Dialogue Management Evaluation

April 1999

Esprit Long-Term Research Concerted Action No. 24823

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

DISC

TITLE	**D3.10 Working paper on dialogue management evaluation.**
PROJECT	DISC (Esprit Long-Term Research Concerted Action No. 24823)
EDITORS	-
AUTHORS	Niels Ole Bernsen, Natural Interactive Systems Laboratory, University of Southern Denmark, Denmark nob@nis.sdu.dk
ISSUE DATE	28.4.1999
DOCUMENT ID	wp3d3.10
VERSION	1
STATUS	Draft
NO OF PAGES	34
WP NUMBER	3
LOCATION	
KEYWORDS	WP3, dialogue management, best practice, evaluation

Document Evolution

Version	Date	Status	Notes
1	1/3/99	Draft	First draft published for review from partners.
2	28.4.99	Draft	Second draft for DISC review

Working Paper on Dialogue Management Evaluation

Abstract

This DISC Working Paper present a first outline of a systematic approach to the evaluation of spoken language dialogue systems (SLDSs) and their components. The approach is illustrated for the case of dialogue manager evaluation. The basic components of the approach are (a) a model of the core properties of the SLDS or component to be evaluated; (b) a set of evaluation criteria which have been systematically generated from the model; (c) a generic template for characterising individual evaluation criteria with respect to, among other things, which core property is being evaluated, the types of evaluation involved, which methods to use, when to evaluate the property during the life-cycle, importance of evaluation, difficulty and cost of evaluation; and (d) filled templates for each individual evaluation criterion. Using this approach, 35 evaluation criteria for dialogue manager evaluation are described.

Contents

1. Introduction

This DISC Working Paper addresses the question: what does it take to turn existing lore on evaluation of spoken language dialogue systems (SLDSs) and their components into a developing applied science? The SLDS component chosen for the purpose of discussion is the dialogue manager. However, the basic issues involved in establishing scientific foundations for (a) dialogue manager evaluation may well be the same for (b) speech recognisers, (c) speech generators, (d) natural language understanding and generation components and, possibly, for (e) system integration and (f) human factors of SLDSs as well. This, at least, is the working hypothesis of the present paper. (a) - (f) comprise the aspects of SLDSs investigated in the DISC project.

So, what does it take to turn existing results and lore on evaluation of dialogue managers into a developing applied science? It seems clear that more is needed than just taking existing software engineering best evaluation practice and, somehow, applying it to dialogue manager evaluation. This will provide a series of useful generic concepts of the software life-cycle in general and of software evaluation in particular but will not succeed in generating the specific forms of evaluation required for dialogue manager evaluation as distinct from evaluation of software with functionalities and roles that are very different from those of dialogue managers. More is also needed than to pick a small number of more or less well-known evaluation criteria for dialogue manager evaluation, analyse them and recommend how they should be applied. This approach will not be able to explain where these criteria come from, why they are more important than all other possible criteria or, indeed, what those other possible criteria are in the first place.

Rather, what seems to be needed to complement the approaches just mentioned, is (1) a comprehensive and systematic theory of what dialogue managers do, or might be doing as part of the SLDSs of which they are components, such as to identify the topic(s) addressed in the most recent user input or to maintain a linguistic dialogue history. Such a theory is presented in the form of the DISC 'grid' in [Bernsen et al. 1998a] as an elaboration of the theory in [Bernsen et al. 1998b]. The theory in [Bernsen et al. 1998a] is based on in-depth analysis by several partners in DISC of a variety of dialogue managers from SLDSs developed in industry and research. The analysis produced a comprehensive model of the *properties* which dialogue managers need to have, or may have, depending on factors such as application goal, task complexity, domain, intended users etc. [Kuppevelt et al. 1999].

Based on (1) considered as an ordered list of the possible core properties of dialogue managers, it is possible to generate (2) a systematic overview of the *criteria* which may be used in dialogue manager evaluation. For each property characterising a particular dialogue manager, the question for evaluation is, roughly: is the property needed for the application and is it adequate or sufficient, or is it not? If, strictly speaking, the property is not needed in the system being evaluated, this may be a case of over-engineering. The terms 'adequacy' and 'sufficiency' serve as place-holders for any other highest-level evaluation concept, such as "does the component do its work properly" and the like. In other words, once we know the core properties of dialogue managers, we know what could be evaluated about them.

However, knowing *what* could be evaluated about a particular dialogue manager can be far from knowing *how* to evaluate a particular property, *when* to do it in the software life-cycle, which *type* of evaluation one is dealing with, how *important* evaluation is etc. Such questions can be asked with respect to any software evaluation criterion including any evaluation

criterion for dialogue managers. This leads to the idea of creating (3) an *evaluation template* which explains the things one needs to know about a particular evaluation criterion (i.e. the when, the how etc.) in order to apply it correctly for the evaluation of a particular property of some dialogue manager. In addition, to make the evaluation template reasonably self-sufficient for practical use, it could introduce some of the most important concepts relevant to understanding software evaluation, such as different types of evaluation, different methods of evaluation, different phases in the software life-cycle etc. Finally (4), as the evaluation template itself is a generic one it might turn out to be applicable not only to dialogue manager evaluation but to the evaluation of all aspects of SLDSs.

To summarise the argument of the paper: (1) based on a model for what dialogue managers do, or may be doing, any SLDS dialogue manager can be viewed as a structured collection of core properties where the particular structure and nature of the properties depend on many factors, such as application goal, domain, task, intended users etc. (2) Dialogue manager evaluation consists in evaluating those core properties, or the most important of them, in terms of their adequacy or sufficiency. (3) The evaluation of a particular dialogue manager property can be described through filling an evaluation template which seeks to answer such questions as: when should evaluation be done, how should it be done, how costly and difficult is it to do etc.? (4) If this approach works for dialogue managers, the hypothesis is that it might work for each component of SLDSs, for SLDSs as integrated systems and for the human factors of SLDSs. As (1) has been described elsewhere [Bernsen et al. 1998a], this paper will focus mainly on (2) and (3). In what follows, Section 2 briefly characterises the dialogue manager of an SLDS. Section 3 proposes a draft evaluation template for dialogue manager evaluation. Section 4 discussed the generation of evaluation criteria for dialogue managers. Section 5 applies the template to the evaluation of dialogue manager properties. Section 6 concludes the paper.

It is emphasised that the present paper is a first iteration on the described approach. Several iterations and tests will obviously be needed to bring the approach in a shape which will enable SLDS and component developers to benefit from it.

2. Dialogue Management: a Birds Eye View

This section very briefly presents the dialogue manager model which is described in detail in [Bernsen et al. 1998a].

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

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

- use the knowledge of the current dialogue context and the local and global focus of attention it may possess to:

- map from the semantically significant units in the user's most recent input (if any), as conveyed by the speech and language layers, onto the sub-task(s) (if any) addressed by the user;

- analyse the user's specific sub-task contribution(s) (if any);

- use the user's sub-task contribution to:

- execute a series of preparatory actions (consistency checking, input verification, input completion, history checking, database retrieval, etc.) usually leading to:

- the generation of output to the user, either by the dialogue manager itself or through output language and speech layers.

The dialogue management activities just described include:

- sub-task identification;
- advanced linguistic processing;
- domain communication;
- meta-communication;
- other forms of communication;
- expression of meaning;
- error loops and graceful degradation;
- feedback;
- closing the dialogue;
- novice and expert users, user groups;
- other relevant user properties.

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

- change or update its representation of the current dialogue context; and

- generate whatever constraint-based support it may provide to the speech and language layers.

The dialogue management activities just described include:

- support from the dialogue manager to the speech and language layers;
- controlling user input;
- input prediction/prior focus;
- histories.

At a high level of abstraction, what the dialogue manager has to do thus is to apply sets of decision - action rules, possibly complemented by statistical techniques, to get from (context + user input) to (preparatory actions + output generation + context revision + speech and language layer support). For simple tasks, this may reduce to the execution of a transformation from, e.g., (user input keywords from the speech layer) to (minimum preparatory actions + dialogue manager-based output generation including repair meta-communication) without the use of (input or output) language layers, context representations and speech and language layer support. Different dialogue managers may be represented in terms of which increased-complexity properties they add to this simple model of a dialogue manager.

The above model was used to identify and, partially, structure the evaluation criteria described in Section 5.

3. A Draft Template for Evaluating Aspects of SLDSs

The evaluation template includes seven entries A - G. The template is a generic tool to which correspond an "empty" template version which must be filled in for each property to be evaluated. If and when presented in the web, the template's terminological information will be in the form of hypertext links. This will make the basic template much shorter. The seven entries are:

A. 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. The filled evaluation template should mark whether the criterion concerns a specific or generic property.

B. 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 in 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 considered "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.

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. The individual filled templates generally ignore such *internal comparative* evaluation. An equally important but much more difficult form of comparative evaluation is comparison between different SLDSs or components. The general problem with this *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.

C. Method(s) of evaluation

This entry describes the methods of evaluation that 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 E 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 have been considered [Bernsen et al. 1998a]. Evaluation also consists in checking whether 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 the recogniser is switched off and replaced by 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 before implementation begins. 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. 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.

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.

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, 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.

D. Needs and dependencies

This entry comments on the *need for the property* being evaluated, which may be relative to other factors that are specified, such as the task or the distribution of dialogue initiative among user and system.

E. 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 including requirements and design specification. Pointedly expressed, 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 better 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. This 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 while the end result of this phase should be an implemented and debugged system or component 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 system, such as an 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.

Porting deals with re-using the SLDS or component new purposes.

F. 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 quite crucial to get some property right even if that property, such as speech acts identification, is relevant only to few current systems at this point.

G. Difficulty and cost of evaluation

This entry comments on the difficulties and costs 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;

- the difficulty of evaluation may depend on the existence of *unsolved research problems.* These may be more or less severe;

- 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 done by a consultant; easy evaluation can be costly, for instance because of the volume of data involved.

4. Generating Evaluation Criteria for Dialogue Managers

To each SLDS aspect corresponds a set of evaluation criteria. In this section we illustrate the proposed approach to evaluation through the aspect of dialogue management. The current situation in dialogue manager evaluation as described below is probably worse than with respect to other aspects of SLDSs, such as speech recognisers, but this serves to make dialogue manager evaluation an interesting testing ground for the proposed approach.

In addressing the issue of dialogue manager evaluation, it is important to keep two different situations in mind, i.e. (1) evaluation of the SLDS of which the dialogue manager forms a part, and (2) evaluation of the dialogue manager *per se.* Dialogue manager evaluation is required in both situations. However, (1) obviously makes it harder to exactly distinguish between those parts of the SLDS's performance which are due to the dialogue manager alone, those which are due to the performance of other system components, and those which are due to interaction between the dialogue manager and other system components. Poor speech recognition, for instance, can only to a certain extent be counter-balanced by good dialogue manager design. If the speech recognition is too poor, the users will walk away even if they are faced with a brilliant dialogue manager. (2) may be one in which the dialogue manager is being selectively evaluated as part of SLDS development or it may be one in which the dialogue manager is itself the sole target of development. In both of the latter cases, the evaluation criteria involved are likely to be more dialogue manager specific than those involved in evaluating the dialogue manager as part of an SLDS as a whole. As the dialogue manager influences many different parts of SLDS performance and processing, a full list of dialogue manager evaluation criteria is likely to overlap with evaluation criteria for system integration as well as evaluation criteria for the human factors aspect of the system. This is not a problem in DISC which covers all these aspects of SLDSs but it does imply some vagueness with respect to whether or not a certain evaluation criterion pertains to dialogue management.

Only a few years ago, the field of dialogue management was so new that evaluation criteria hardly existed at all, i.e. no one had really thought about what to test, how to do it etc., and no experience from previous development efforts was available. Evaluation criteria were invented *ad hoc* when the dialogue manager and the SLDS in which it was embedded came up for evaluation. This way of doing things is still quite common, in particular in research projects,

but cannot be recommended because it means that developers have little support during development in terms of criteria that the dialogue manager should fulfil in order to be considered satisfactory and acceptable. The definition, from the start of the life-cycle, of clear, relevant and appropriate evaluation criteria, and the continuous and methodologically sound evaluation of progress with reference to those criteria, should be main characteristics of dialogue manager development and evaluation.

There is still no well-defined set of evaluation criteria to draw upon in the literature. Most criteria in current use are purely quantitative, such as transaction success [Bernsen et al. 1998b]. These can be both useful and relatively easy to apply but provide insufficient information on the real quality of a dialogue manager. Subjective measures from user questionnaires and the like, on the other hand, such as scorings of perceptions of an SLDS on a five-point scale, are often difficult to interpret as input on the quality of the dialogue manager. Objective qualitative evaluations from experts are difficult to come by already because there are so few experts in this field. In this situation, it seems that one thing that might help advance the state-of-the-art is to generate all possible evaluation criteria for dialogue managers, represent each criterion using the template presented in Section 3, and test how this apparatus works in real development projects in industry and research.

Based on in-depth analysis of a series of dialogue managers, a semi-structured list of possible dialogue manager properties was established following a model of significant possible steps in the dialogue manager's processing of input (Section 2, [Bernsen et al. 1998a]). Each dialogue manager property encountered on the list is a possible "what" for evaluation (cf. Section 3, template entry A) and must be described according to the evaluation template presented in Section 3. Somewhat to our surprise, the list of possible dialogue manager properties generated no less than 35 possible evaluation criteria for dialogue managers. The criteria are presented in a structured list which mostly follows the model of significant possible steps in the dialogue manager's processing of input just referred to, as follows:

1. Use of knowledge of the current dialogue context and local and global focus of attention. 3 evaluation criteria. Example: dialogue segmentation adequacy.

2. Map from the semantically significant units in the user's most recent input (if any), as conveyed by the speech and language layers, onto the sub-task(s) (if any) addressed by the user. 5 evaluation criteria. Example: sub-task or topic identification success.

3. Analyse the user's specific sub-task contribution(s) (if any) through the execution of a series of preparatory actions (consistency checking, input verification, input completion, history checking, database retrieval, etc.). 4 evaluation criteria. Example: database information sufficiency.

4. Generation of output to the user, either by the dialogue manager itself or through output language and/or speech layers and/or other output modalities. 10 evaluation criteria. Example: adequacy of on-line information to users on how to interact with the system.

5. Global issues of dialogue management evaluation. 7 evaluation criteria. Example: feedback strategy sufficiency: processing feedback.

6. Global issues of dialogue system evaluation. 7 evaluation criteria. Example: real-time performance.

It is only the criteria belonging to (5) and (6) above which do not correspond to the processing model, illustrating the point made earlier in this section that some dialogue manager evaluation criteria come close to being evaluation criteria for SLDSs as a whole.

5. Criteria for Dialogue Manager Evaluation

5.1 Overview of the criteria

This is an overview of the criteria described in Section 5.3:

1. Use of knowledge of the current dialogue context and local and global focus of attention.

1.1 Dialogue segmentation adequacy.

1.2 Relevance and success of predictions.

1.3 Sufficiency of dialogue histories (linguistic, topic, task, performance) and their use.

2. Map from the semantically significant units in the user's most recent input (if any), as conveyed by the speech and language layers, onto the sub-task(s) (if any) addressed by the user.

2.1 Adequacy of dialogue manager support for dedicated processing of ellipsis.

2.2 Adequacy of dialogue manager support for co-reference interpretation.

2.3 Adequacy of dialogue manager support for indirect speech act interpretation.

2.4 Adequacy of dialogue manager support for multimodal input fusion, i.e. for combining more or less simultaneous input messages expressed in different modalities into a single semantic representation or sub-task contribution.

2.5 Sub-task or topic identification success.

3. Analyse the user's specific sub-task contribution(s) (if any) through the execution of a series of preparatory actions (consistency checking, input verification, input completion, history checking, database retrieval, etc.).

3.1 Adequacy of domain inferences.

3.2 Database information sufficiency.

3.3 Adequacy of strategy for identifying and responding to out-of-task and/or out-of-domain input.

3.4 Robustness - wrt. unexpected (user) deviations from the dialogue plan.

4. Generation of output to the user, either by the dialogue manager itself or through output language and/or speech layers and/or other output modalities.

4.1 Adequacy of on-line information to users on how to interact with the system.

4.2 Adequacy of on-line information to users on the system's domain and task coverage.

4.3 Feedback strategy sufficiency: information feedback.

4.4 Sufficiency of meta-communication facilities: system-initiated repair, system-initiated clarification, user-initiated repair, user-initiated clarification.

4.5 Robustness - wrt. error loops and graceful degradation.

4.6 Adequacy of operator fallback strategy.

4.7 Conformance of system phrases to the cooperativity guidelines (individually as well as in context) / dialogue interaction problems caused by flawed system utterance design.

4.8 User model adequacy (adequacy of (non-) distinction between novice and expert users, etc.).

4.9 Adequacy of initiative distribution among user and system relative to the task.

4.10 Adequacy of output distribution over speech and other output modalities in multimodal SLDSs.

5. Global issues of dialogue management evaluation.

5.1 Complexity of the interaction model expressed, e.g., in terms of number of nodes if a graph representation is used.

5.2 Task and domain model coverage: are these sufficient? Are they delineated in a principled and intuitive way?

5.3 Degree of utilisation of the knowledge sources available to the dialogue manager.

5.4 Feedback strategy sufficiency: processing feedback.

5.5 Ease of maintenance/modification of the dialogue manager and/or of its individual modules.

5.6 Portability (re-usability) of the dialogue manager and/or of its individual modules.

6. Global issues of dialogue system evaluation

6.1 Average time for task completion as compared to other ways (not using an SLDS) of solving the same task.

6.2 Average cost per transaction as compared to other ways of solving the same task (not using an SLDS).

6.3 Average number of turns to complete a task compared to human-human interaction.

6.4 Real-time performance.

6.5 Transaction success rate.

6.6 Translation success rate of spoken language translation (support) systems.

6.7 User satisfaction and other subjective parameters (explain).

5.2 An empty template

The empty template used in describing the 35 dialogue manager evaluation criteria in Section 5.3 looks as follows:

A. What is being evaluated

B. Type of evaluation

C. Method(s) of evaluation

D. Needs and dependencies

E. Life-cycle phase(s)

F. Importance of evaluation

G. Difficulty and cost of evaluation

5.3 Criteria for dialogue manager evaluation

This section presents a systematically generated series of evaluation criteria that are relevant to dialogue manager evaluation. Whilst not all of them are applicable to all systems, each criterion can help throw light on the adequacy and performance of some dialogue managers. They need not, of course, all be included in the "official" list of evaluation criteria stated in the requirements specification. Even if not included, they can still be highly relevant for evaluating how good or bad the dialogue manager is and what progress is being made during its development.

1. Use of knowledge of the current dialogue context and local and global focus of attention.

1.1 Dialogue segmentation adequacy.

A. What is being evaluated: is the number of segmentation units distinguished by the dialogue manager, such as user and system turns, a set of dialogue acts, domain communication and meta-communication, start, main body of dialogue and end-part of dialogue, etc., sufficient for the application? Can the system correctly distinguish among the units when processing real dialogues? Generic property.

B. Type of evaluation: qualitative, possibility for obtaining quantitative percentages for unit application (correct vs. incorrect application).

C. Method(s) of evaluation: design analysis, glassbox test, blackbox test. For high-complexity segmentation in research: Wizard of Oz data analysis, controlled experiments, field studies.

D. Needs and dependencies: the need for dialogue segmentation into different types of unit is relative to, among many other things, the task, user input complexity and the distribution of initiative among user and system.

E. Life-cycle phase(s): early design, implementation. For high-complexity segmentation in research: simulation, field evaluation. Evaluation during early design is mandatory. If the adopted segmentation strategy is inadequate, the dialogue manager will not be able to do its job. This is especially true if a crucial distinction between units is missing. If the appropriate units are being distinguished but the distinction is semantically flawed, the dialogue manager will in some cases not have appropriate unit tokens to work with. During implementation, evaluation takes the form of testing that the adopted segmentation strategy is adequate.

F. Importance of evaluation: medium. Current commercial systems work with few types of segmentation unit, such as the turn. This is normally rather easy for the system to do. When more units are being used, evaluation importance becomes high.

G. Difficulty and cost of evaluation: easy and low cost for simple systems which only need to distinguish between few types of unit. From then on, difficulty and cost increase.

1.2 Relevance and success of predictions.

A. What is being evaluated: is it a good solution for the SLDSs to be using predictions? Are predictions of the next user input relevant and correct? Note that prediction can pertain to several different aspects of the user's input, such as: which topic will it address? Which topic will it not address? Which vocabulary will it involve? Which grammar will it involve? Etc. Generic property.

B. Type of evaluation: qualitative, possibility for obtaining quantitative percentages for correctness (correct vs. incorrect prediction).

C. Method(s) of evaluation: design analysis, glassbox test, blackbox test. For high-complexity predictions: simulation, field evaluation. Wizard of Oz data analysis, controlled experiments, field studies.

D. Needs and dependencies: the need for predictions is relative to, among other things, the complexity of the user input and the complexity of the task.

E. Life-cycle phase(s): early design, implementation. For high-complexity predictions: simulation, field evaluation. Evaluation during early design is mandatory. In the early design phase it must be evaluated if some form(s) of prediction is helpful and feasible. During implementation, evaluation focuses on the relevance and success of predictions.

F. Importance of evaluation: low. Most current commercial systems do not use predictions. If the dialogue manager uses prediction, it is crucial to make sure that the prediction strategy works properly. The penalty is that relevant but unpredicted user input will not be understood. This will leave the system (and the user) in a limbo.

G. Difficulty and cost of evaluation: relatively easy for simple system-directed dialogue systems, more difficult and costly as task complexity and user input complexity grow.

1.3 Sufficiency of dialogue histories (linguistic, topic, task, performance) and their use.

A. What is being evaluated: the dialogue manager as well as other parts of the SLDS may be using (creating, maintaining as well as drawing upon) one or several dialogue histories, and may be using those histories for an open-ended variety of purposes. In each case, the questions for evaluation are: does the system have the dialogue histories it needs? Is the particular dialogue history that is being used up to the job(s) for which it has been created? Is it being properly used by the dialogue manager? Is it being properly maintained by the dialogue manager? Generic property.

B. Type of evaluation: qualitative.

C. Method(s) of evaluation: design analysis, glassbox test, blackbox test. For high-complexity histories: Wizard of Oz data analysis, controlled experiments, field studies.

D. Needs and dependencies: the number and nature of the dialogue histories which the dialogue manager should use depend on numerous factors, such as the task, the distribution of initiative among user and system, user input complexity etc.

E. Life-cycle phase(s): early design, implementation. For high-complexity histories: simulation, field evaluation. Evaluation during early design is mandatory. Dialogue histories may be drawn upon by many different processes performed by the dialogue manager as well as, in some cases, processes performed in the natural language and speech layers of the system, such as processing of ellipsis, co-reference and speech acts. During implementation, evaluation focuses on how the dialogue manager interacts with the implemented dialogue histories.

F. Importance of evaluation: medium. Most current commercial systems need at least a task history whose working should be flawless.

G. Difficulty and cost of evaluation: relatively easy for simple systems, more difficult and costly as the number of histories involved, and the purposes for which they are being used, grow.

2. Map from the semantically significant units in the user's most recent input (if any), as conveyed by the speech and language layers, onto the sub-task(s) (if any) addressed by the user.

2.1 Adequacy of dialogue manager support for dedicated processing of ellipsis.

A. What is being evaluated: are there sufficient reasons for the system to do ellipsis processing? Does the dialogue manager maintain a linguistic dialogue history which provides appropriate context for ellipsis processing? Does the dialogue manager provide appropriate support for ellipsis processing (cf. 1.3)? Specific property.

B. Type of evaluation: qualitative. Counting success rates in absolute terms would seem to make little sense at the moment.

C. Method(s) of evaluation: design analysis, glassbox test, blackbox test. Current natural language processing techniques should be sufficient for producing adequate blackbox tests.

D. Needs and dependencies: the need for dedicated processing of ellipsis depends on, among other things, the task and the user input complexity. If the dialogue manager is involved in the dedicated processing of ellipsis, this is because it maintains a linguistic dialogue history which is being used in the linguistic processing of the user's input.

E. Life-cycle phase(s): early design, implementation. During early design, it must be evaluated if the system to be designed does need dedicated processing of ellipsis.

F. Importance of evaluation: low. Except for advanced spoken language translation systems, most task-oriented SLDSs do not need dedicated processing of ellipsis. No current commercial SLDS performs dedicated processing of ellipsis. Few research systems do. The many ellipsis phenomena that standardly occur in spoken discourse are normally taken into account in the system's grammar.

G. Difficulty and cost of evaluation: difficult. Modest conceptual difficulties remain to be resolved in research.

2.2 Adequacy of dialogue manager support for co-reference interpretation.

A. What is being evaluated: are there sufficient reasons for the system to do co-reference interpretation? Does the dialogue manager maintain a linguistic dialogue history which provides appropriate context for co-reference interpretation? Does the dialogue manager provide appropriate support for co-reference interpretation (cf. 1.3)? Specific property.

B. Type of evaluation: qualitative. Counting success rates in absolute terms would seem to make little sense at the moment.

C. Method(s) of evaluation: design analysis, Wizard of Oz data analysis, glassbox test, blackbox test, controlled experiments, field studies.

D. Needs and dependencies: the need for co-reference interpretation depends on, among other things, the task and user input complexity. If the dialogue manager is involved in co-reference interpretation, this is because it maintains a linguistic dialogue history which is being used in the linguistic processing of the user's input.

E. Life-cycle phase(s): early design, simulation, implementation, final evaluation, field evaluation. During early design, it must be evaluated if the system to be designed does need co-reference interpretation.

F. Importance of evaluation: low. No current commercial SLDSs perform co-reference interpretation.

G. Difficulty and cost of evaluation: difficult. Significant conceptual difficulties remain to be resolved in research.

2.3 Adequacy of dialogue manager support for indirect speech act interpretation.

A. What is being evaluated: are there sufficient reasons for the system to do indirect speech acts interpretation? Does the dialogue manager maintain a linguistic dialogue history which provides appropriate context for indirect speech acts interpretation? Does the dialogue manager provide appropriate support for indirect speech acts interpretation (cf. 1.3)? Specific property.

B. Type of evaluation: qualitative. Counting success rates in absolute terms would seem to make little sense at the moment.

C. Method(s) of evaluation: design analysis, Wizard of Oz data analysis, glassbox test, blackbox test, controlled experiments, field studies.

D. Needs and dependencies: the need for indirect speech act interpretation depends on, among other things, the task and user input complexity. If the dialogue manager is involved in indirect speech acts interpretation, this is because it maintains one or more dialogue histories which are used in the linguistic processing of the user's input.

E. Life-cycle phase(s): early design, simulation, implementation, final evaluation, field evaluation. During early design, it must be evaluated if the system to be designed does need indirect speech acts interpretation.

F. Importance of evaluation: low. No current commercial SLDSs perform indirect speech acts interpretation.

G. Difficulty and cost of evaluation: difficult. Significant conceptual difficulties remain to be resolved in research.

2.4 Adequacy of dialogue manager support for multimodal input fusion, i.e. for combining more or less simultaneous input messages expressed in different modalities into a single semantic representation or sub-task contribution.

A. What is being evaluated: potentially, this is an extremely wide and complex area of evaluation because 'multimodal input' covers a wide range of possible input combinations. A

well-known example is the combination of speech and pointing gesture through mouse clicks. No current system seems to require dialogue manager support but this may change in the future. Little is known at present about the nature of the dialogue manager support that will be needed. Generic property.

B. Type of evaluation: qualitative.

C. Method(s) of evaluation: design analysis, Wizard of Oz data analysis, glassbox test, blackbox test, controlled experiments, field studies.

D. Needs and dependencies: the need for multimodal input fusion depends on numerous factors, such as the task, the application etc. Given the large variety of available input modalities, the term 'multimodal input fusion' is a very broad one. In the future, the present entry (2.4) will have to be expanded into several different, modality-specific entries. As for 2.1 through 2.3 above, multimodal input fusion is likely to be done primarily outside the dialogue manager which may contribute, for instance, various forms of history support.

E. Life-cycle phase(s): early design, simulation, implementation, final evaluation, field evaluation.

F. Importance of evaluation: low but increasing. No current commercial SLDSs perform multimodal input fusion.

G. Difficulty and cost of evaluation: difficult. Significant conceptual difficulties remain to be resolved in research.

2.5 Sub-task or topic identification success.

A. What is being evaluated: the extent to which the dialogue manager succeeds in identifying the sub-task(s) or topic(s) addressed in the user's utterances. Specific property.

B. Type of evaluation: qualitative. Quantitative evaluation is possible in SLDSs in which task slots or translation templates are being filled directly from identified sub-task(s) or topic(s). One then measures hits vs. misses. In other SLDSs, the dialogue manager may get the sub-task(s) or topic(s) right whilst still failing to get the user's contribution to the sub-task(s) or topic(s) right.

C. Method(s) of evaluation: design analysis, glassbox test, blackbox test. For high-complexity identification: Wizard of Oz data analysis, controlled experiments, field studies.

D. Needs and dependencies: virtually all SLDSs need to do sub-task or topic identification, and this can be done in many different ways. Is sometimes linked to prediction success (cf. 1.2).

E. Life-cycle phase(s): early design, implementation. For high-complexity identification: simulation, final evaluation, field evaluation.

F. Importance of evaluation: high. Needed for virtually all SLDSs. This is a core measure of the successful working of a dialogue manager.

G. Difficulty and cost of evaluation: difficulty and cost grow with the complexity of the task(s), user input complexity etc.

3. Analyse the user's specific sub-task contribution(s) (if any) through the execution of a series of preparatory actions (consistency checking, input verification, input completion, history checking, database retrieval, etc.).

3.1 Adequacy of domain inferences.

A. What is being evaluated: the extent to which the system is able to perform relevant and correct inferencing over the user's input, such as inferring an absolute date from a reference to a day of the week. Specific property.

B. Type of evaluation: qualitative, subjective.

C. Method(s) of evaluation: design analysis, blackbox test. For high-complexity inferential setups: Wizard of Oz data analysis, controlled experiments, field studies, questionnaires, interviews.

D. Needs and dependencies: the domain inferences needed by a particular SLDS are highly dependent on task and domain. Note that it is often difficult to circumscribe the domain inferences needed by the application.

E. Life-cycle phase(s): early design, implementation. For high-complexity inferencing: simulation, final evaluation, field evaluation.

F. Importance of evaluation: medium/high. All but simple SLDSs need some amount of domain inferencing.

G. Difficulty and cost of evaluation: relatively easy for simple systems. The difficulty and cost grow rapidly with the complexity of the task(s), the domain, user input complexity etc. because of the need for a test corpus of sufficient size and realism wrt. task, domain and user behaviour.

3.2 Database information sufficiency.

A. What is being evaluated: whether the system's database (if any) contains all the correct data needed for the application. Specific property.

B. Type of evaluation: qualitative.

C. Method(s) of evaluation: design analysis, to a lesser extent blackbox test.

D. Needs and dependencies: the database information needed by a particular SLDS (if any) is highly dependent on task and domain.

E. Life-cycle phase(s): early design, to a lesser extent implementation, maintenance.

F. Importance of evaluation: medium/high. Many SLDSs need a database of domain facts and rules. The database may need continuous or regular updating. It is essential that the database be correct and complete relative to the application.

G. Difficulty and cost of evaluation: it is often rather easy to identify the database information needed by the application.

3.3 Adequacy of strategy for identifying and responding to out-of-task and/or out-of-domain input.

A. What is being evaluated: whether there are good reasons for including this property in the system. The extent to which the SLDS is able to identify and respond properly to out-of-task and/or out-of-domain input to which the developers want it to respond. Specific property.

B. Type of evaluation: qualitative, subjective.

C. Method(s) of evaluation: design analysis, blackbox test. For high-complexity issues: Wizard of Oz data analysis, controlled experiments, field studies, questionnaires, interviews.

D. Needs and dependencies: when the user input is out-of-task and/or out-of-domain, a special reason is needed for ensuring that information survives the semantic parsing and reaches the dialogue manager. So this is a case of deciding, during early design, if the users need informative feedback on why the system cannot solve the sub-task addressed in their input. One such reason could be that it is not obvious why the system's task has been delineated exactly the way it has. Otherwise, out-of-task and/or out-of-domain input will be treated the same way as unrecognised input or input that is inconsistent with the system's grammar.

E. Life-cycle phase(s): early design, implementation. For high-complexity issues: simulation, final evaluation, field evaluation.

F. Importance of evaluation: low. In most cases, ways should be found to regiment the users' relevant input so that it does not go beyond the system's task and domain.

G. Difficulty and cost of evaluation: relatively easy to do once it has been decided to what out-of-task and/or out-of-domain input the system should provide informative feedback.

3.4 Robustness - wrt. unpredicted (user) deviations from the dialogue plan.

A. What is being evaluated: the extent to which the system is able to correctly respond to unpredicted (user) deviations from the dialogue plan. Specific property.

B. Type of evaluation: qualitative, subjective.

C. Method(s) of evaluation: design analysis, blackbox test. For high-complexity issues: Wizard of Oz data analysis, controlled experiments, field studies, questionnaires, interviews.

D. Needs and dependencies: by contrast with 3.3, in 3.4 the user remains within the task and the domain. The dialogue manager uses local and/or global focus (the dialogue plan). The task does not have to be complex but, in fact, often is because that is when local and/or global focus tend to be used.

E. Life-cycle phase(s): early design, implementation. For high-complexity issues: simulation, final evaluation, field evaluation.

F. Importance of evaluation: low. No current commercial system appears to have this kind of robustness, i.e. they treat unpredicted (user) deviations from the dialogue plan the same way as unrecognised input or input that is inconsistent with the system's grammar.

G. Difficulty and cost of evaluation: potentially difficult and costly. Requires a test corpus of sufficient size and realism wrt. task, domain and user behaviour. Difficulty and cost grow with the complexity of the task(s), the domain, user input complexity etc.

4. Generation of output to the user, either by the dialogue manager itself or through output language and/or speech layers and/or other output modalities.

4.1 Adequacy of on-line information to users on how to interact with the system.

A. What is being evaluated: the extent to which the system provides the user with correct and sufficient information on how to interact with it. Specific property.

B. Type of evaluation: qualitative, subjective.

C. Method(s) of evaluation: design analysis, possibly Wizard of Oz data analysis, controlled experiments, field studies, questionnaires, interviews.

D. Needs and dependencies: the need for, and the nature of, on-line information to users on how to interact with the system, depend primarily on the user group.

E. Life-cycle phase(s): early design, possibly simulation, final evaluation, field evaluation.

F. Importance of evaluation: high. Most SLDSs need some form of on-line information to the users on how to interact with the system. This information must be carefully crafted to fit the targeted user group(s). If the information is incomplete, wrong, misleading etc., this may seriously affect the users' perception of the usability of the system.

G. Difficulty and cost of evaluation: potentially difficult and costly. In many cases, a test corpus of sufficient size and realism wrt. user behaviour is required to verify that the on-line information provided is OK.

4.2 Adequacy of on-line information to users on the system's domain and task coverage.

A. What is being evaluated: the extent to which the system provides the user with correct and sufficient information on its domain and task coverage. Specific property.

B. Type of evaluation: qualitative, subjective.

C. Method(s) of evaluation: design analysis, possibly Wizard of Oz data analysis, controlled experiments, field studies, questionnaires, interviews.

D. Needs and dependencies: the need for, and the nature of, on-line information to the users on the system's domain and task coverage, depend primarily on the task and domain as well as on the user group.

E. Life-cycle phase(s): early design, possibly simulation, final evaluation, field evaluation.

F. Importance of evaluation: high. All SLDSs need to provide some form of on-line information to the users on the system's domain and task coverage. This information must be carefully crafted to fit the targeted user group(s). If the information is incomplete, wrong, misleading etc., this may seriously affect the users' perception of the usability of the system. 4.2 is related to 4.7.

G. Difficulty and cost of evaluation: relatively easy to do provided that the system's domain and task coverage is intuitively easy to grasp. In some cases, however, a test corpus of sufficient size and realism wrt. user behaviour may be required for verification.

4.3 Feedback strategy sufficiency: information feedback.

A. What is being evaluated: the extent to which the system's provision of feedback to users on the information provided by them is sufficient to avoiding undesired user input. Specific property.

B. Type of evaluation: qualitative.

C. Method(s) of evaluation: design analysis, possibly Wizard of Oz data analysis, controlled experiments, field studies.

D. Needs and dependencies: the nature of the information feedback to be provided by the system depends primarily on the task(s) but may also depend on the user group.

E. Life-cycle phase(s): early design, possibly simulation, final evaluation, field evaluation.

F. Importance of evaluation: high. All SLDSs need some form of information feedback in order to inform the user about what the system has understood or about what the system is now doing in response to the user's input. Information feedback may be provided in many different ways and not necessarily through speech output. Information feedback seems to be extremely important to the user's perception of the system. 4.3 is related to 4.7.

G. Difficulty and cost of evaluation: can be complex and costly, and may involve comparative empirical studies of different feedback strategies.

4.4 Sufficiency and adequacy of meta-communication facilities: system-initiated repair, system-initiated clarification, user-initiated repair, user-initiated clarification.

A. What is being evaluated: the extent to which the meta-communication facilities offered by the system are up to the system's purpose. Thus, meta-communication sufficiency concerns (a) whether the system has the generic meta-communication facilities it should have, and (b) whether these facilities are individually sufficient for their purposes. Generic property.

B. Type of evaluation: qualitative, subjective.

C. Method(s) of evaluation: design analysis, Wizard of Oz data analysis, blackbox test, often also controlled experiments, field studies, questionnaires, interviews. Note that realistic system meta-communication behaviour can be difficult to simulate by humans.

D. Needs and dependencies: most repair meta-communication, whether system-initiated or user-initiated, can be thought of as task independent whereas most or all clarification meta-communication, whether system-initiated or user-initiated, is task dependent. Meta-communication can also be user group dependent.

E. Life-cycle phase(s): early design, simulation, implementation, final evaluation, field evaluation.

F. Importance of evaluation: high. Virtually all SLDSs need some form of meta-communication, in particular, system-initiated and user-initiated repair meta-communication. The system's – existing or missing - facilities for meta-communication and the degree of success with which they work, are important to the user's perception of the system.

G. Difficulty and cost of evaluation: difficult and potentially costly except for simple systems. A test corpus of sufficient size and realism wrt. user behaviour is often required. Tool support is desirable in order to reduce cost and risk.

4.5 Robustness - wrt. error loops and graceful degradation.

A. What is being evaluated: how easy and elegantly the dialogue manager is able to get itself out of error loops involving the user, including how the dialogue manager gracefully degrades its level of communication with the user in order to reach a level at which the communication is able to proceed (and not, e.g., stop or being handed over to a human operator). Specific property.

B. Type of evaluation: qualitative, subjective.

C. Method(s) of evaluation: design analysis, Wizard of Oz data analysis, blackbox test, potentially controlled experiments, field studies, questionnaires, interviews.

D. Needs and dependencies: 4.5 can be viewed as a continuation of the criteria on meta-communication (4.4), that is, 4.5. is for the situation in which the "front line" 4.4 mechanisms fail to work, in particular, the "front line" mechanisms for system-initiated repair meta-communication.

E. Life-cycle phase(s): early design, simulation, implementation, possibly final evaluation, field evaluation.

F. Importance of evaluation: medium. It seems likely that most SLDSs could benefit from some forms of handling of error loops and graceful degradation. On the other hand, there are limits to the extent to which users will accept graceful degradation before being switched to an operator.

G. Difficulty and cost of evaluation: can be difficult and costly even for medium-complexity graceful degradation. A test corpus of sufficient size and realism wrt. user behaviour is often required.

4.6 Adequacy of operator fallback strategy.

A. What is being evaluated: if the system uses operator fallback, the question is whether the adopted strategy is adequate. Users may get the operator too early for the application to be cost-effective, or too late for the application to gain user acceptance, or in the wrong way. Specific property.

B. Type of evaluation: qualitative, subjective.

C. Method(s) of evaluation: design analysis, often also Wizard of Oz data analysis, controlled experiments, field studies, questionnaires, interviews.

D. Needs and dependencies: 4.6 can be viewed as a "last resort" when system-initiated repair meta-communication does not work, perhaps not even with graceful degradation thrown in.

E. Life-cycle phase(s): early design, implementation, possibly simulation, final evaluation, field evaluation.

F. Importance of evaluation: medium. Only some SLDSs can possibly have operator fallback. However, when operator fallback is present it highly important to get it right.

G. Difficulty and cost of evaluation: crucial details can be difficult and costly to evaluate. Empirical comparison among strategies may be needed including user test corpus analysis and subjective evaluations.

4.7 Conformance of system phrases to the cooperativity guidelines (individually as well as in context) / dialogue interaction problems caused by flawed system utterance design.

A. What is being evaluated: the extent to which the system-generated phrases meet the guidelines for cooperative system behaviour. The nature and number of dialogue interaction problems caused by flawed system utterance design. Specific property.

B. Type of evaluation: qualitative, partly quantitative, partly externally comparative: it is possible to quantify types of guideline violations in a test corpus and generate a percentage which has some independent meaning to it. The qualitative part of the evaluation is to judge the relative severity of different violations of rules for cooperative system utterance design.

C. Method(s) of evaluation: design analysis, Wizard of Oz data analysis, controlled experiments, field studies, questionnaires, interviews.

D. Needs and dependencies: task independent, user independent in principle, i.e. unless one wants to offer professional users a less-than-optimal set of system utterances because they will learn to understand those used by the system anyway (i.e. the hard way through trial-and-error). Affects all aspects of the system's utterances, in domain communication, feedback and meta-communication.

E. Life-cycle phase(s): early design, simulation, field evaluation, final evaluation.

F. Importance of evaluation: high. The purpose of the cooperativity guidelines is to help dialogue designers ensure a smooth user-system dialogue and avoid unnecessary meta-communication. This is likely to strongly affect the user's perception of the system.

G. Difficulty and cost of evaluation: difficult. Takes experts to do at the moment. The difficulty is offset by the potential cost savings by having to find out that the implemented system is less-than-fully usable. A DISC support tool is being developed that should make evaluation easy to do during early design.

4.8 User model adequacy (adequacy of (non-) distinction between novice and expert users, etc.).

A. What is being evaluated: (a) whether or not the system incorporates the (run-time) user models it needs to meet crucial differences in user preferences and behaviour, and (b) whether the incorporated user models work adequately. Generic property.

B. Type of evaluation: qualitative, subjective.

C. Method(s) of evaluation: design analysis, Wizard of Oz data analysis, controlled experiments, field studies, questionnaires, interviews.

D. Needs and dependencies: the need for run-time distinctions between the needs of different users is already apparent in 4.5 (error loops and graceful degradation) and 4.6 (operator fallback) above. So, as soon as users with different needs can be anticipated, run-time distinctions between them should be considered.

E. Life-cycle phase(s): early design, simulation, field evaluation, final evaluation.

F. Importance of evaluation: medium. All SLDSs need a design-time model of the intended users. Models distinguishing in simple ways between the needs of different user groups at run-time, possibly using a performance history for the purpose (see 1.3), are beginning to appear. This is where the dialogue manager comes in. The presence, of lack, of crucial run-time

distinctions between user needs may be an important factor in the user's perception of the system.

G. Difficulty and cost of evaluation: fairly easy for simple mechanisms, more complex from there onwards. A test corpus of sufficient size and realism wrt. user behaviour may be required.

4.9 Adequacy of initiative distribution among user and system relative to the task.

A. What is being evaluated: whether or not the distribution of initiative between user and system is appropriate to the interactive user-system task. Specific property.

B. Type of evaluation: qualitative, subjective.

C. Method(s) of evaluation: design analysis, for innovative applications: Wizard of Oz data analysis, controlled experiments, field studies, questionnaires, interviews.

D. Needs and dependencies: dependent on the task, as stated.

E. Life-cycle phase(s): early design, potentially simulation, implementation, final evaluation, field evaluation.

F. Importance of evaluation: high. All SLDSs embody an initiative distribution among user and system. The evaluator should keep in mind that mixed initiative is not always "the best" solution. Depending on the task, user-driven or system-driven dialogue may be perfectly satisfactory. Wrong design choices can have strong repercussions wrt. users' perception of the system.

G. Difficulty and cost of evaluation: fairly easy for many tasks of low complexity.

4.10 Adequacy of output distribution over speech and, in multimodal SLDSs, other output modalities.

A. What is being evaluated: the correctness of the choices of output modalities for expressing information to users, such as combining output speech and output graphics. Speech-only is not always an adequate output modality and does not always combine well with other output modalities. Generic property.

B. Type of evaluation: qualitative.

C. Method(s) of evaluation: design analysis. For innovative applications: Wizard of Oz data analysis, controlled experiments, field studies.

D. Needs and dependencies: dependent on the task, the users, the environment, and virtually every other variable one could think of.

E. Life-cycle phase(s): early design. For innovative applications: simulation, final evaluation, field evaluation.

F. Importance of evaluation: medium. Getting this right, or not too wrong, can easily decide the life or death of the application.

G. Difficulty and cost of evaluation: can be difficult for innovative applications. In most such cases, trained experts are needed at the moment. A DISC tool is being built to assist design and evaluation.

5. Global issues of dialogue management evaluation.

5.1 Complexity of the interaction model expressed, e.g., in terms of number of nodes if a graph representation is used.

A. What is being evaluated: the complexity of the dialogue structure managed by the dialogue manager. Specific property.

B. Type of evaluation: quantitative, qualitative. The qualitative aspect comes in when judging whether the measured quantity is satisfactory.

C. Method(s) of evaluation: design analysis, often also Wizard of Oz data analysis, blackbox test, controlled experiments, field studies because, despite the best design efforts, interaction model complexity may be forced to grow throughout the life-cycle.

D. Needs and dependencies: this is a typical relational evaluation criterion. Without a specific purpose or context for the evaluation, the complexity "number" is meaningless. One possible purpose could be to reflect on the relationship with real-time performance. Another could be comparison with other SLDSs solving similar or different tasks. Note also that interaction models are not necessarily expressed as graphs, in which cases a quantitative evaluation can be difficult or impossible.

E. Life-cycle phase(s): early design, often also final evaluation, field evaluation.

F. Importance of evaluation: medium. Depends on the purpose or context involved. Not always obvious.

G. Difficulty and cost of evaluation: the qualitative part of the evaluation can be quite difficult. Low cost.

5.2 Task and domain model coverage: are these sufficient? Are they delineated in a principled and intuitive way?

A. What is being evaluated: the adequacy of the underlying models of the task and the domain. Was any meaningful and important user move within the task and/or domain overlooked? Does the task model correspond to the users' model of the task or was the system's task model curtailed in some way? Specific property.

B. Type of evaluation: qualitative, subjective.

C. Method(s) of evaluation: design analysis, Wizard of Oz data analysis, blackbox test, controlled experiments, field studies, questionnaires, interviews.

D. Needs and dependencies: relationship with domain inferences (3.1) and the information provided by the system on its task(s) and domain (4.2).

E. Life-cycle phase(s): early design, simulation, implementation, final evaluation, field evaluation.

F. Importance of evaluation: high. Very important for judging the overall quality of the application. If not supported by adequate on-line information to users, a non-intuitive task and domain model coverage will inevitably generate usability problems.

G. Difficulty and cost of evaluation: difficult. Requires either a shrewd and experienced analytical mind deeply familiar with the task and the domain, or a test corpus of sufficient size and realism wrt. the actual tasks that users want solved (or both).

5.3 Degree of utilisation of the knowledge sources available to the dialogue manager.

A. What is being evaluated: the extent to which the dialogue manager actually uses the implemented knowledge sources rather than showing signs of over-engineering. Generic property.

B. Type of evaluation: qualitative.

C. Method(s) of evaluation: design analysis, glassbox test, blackbox test.

D. Needs and dependencies: none.

E. Life-cycle phase(s): early design, implementation.

F. Importance of evaluation: low. Insufficiently used knowledge sources demands an explanation. For instance, an unused linguistic history is a waste due to over-engineering.

G. Difficulty and cost of evaluation: relatively easy to do.

5.4 Feedback strategy sufficiency: processing feedback.

A. What is being evaluated: the extent to which the system's provision of feedback to users on its current processing task is sufficient to avoiding undesired user input. One part consists in using the system and asking such questions as: when is there processing feedback? What does it consist in? Is it sufficiently informative? Another part consists in letting several users answer these questions, because different users may have different preferences wrt. processing feedback. Specific property.

B. Type of evaluation: qualitative, subjective.

C. Method(s) of evaluation: design analysis, Wizard of Oz data analysis, controlled experiments, field studies, questionnaires, interviews.

D. Needs and dependencies: depends on the task and on the system's real-time behaviour.

E. Life-cycle phase(s): early design, simulation, implementation, field evaluation, final evaluation.

F. Importance of evaluation: high. The quality of the processing feedback is likely to influence the user's perception of the system.

G. Difficulty and cost of evaluation: relatively easy to do.

5.5 Ease of maintenance/modification of the dialogue manager and/or of its individual modules.

A. What is being evaluated: the effort needed to maintain the dialogue manager and/or its individual modules. Generic property.

B. Type of evaluation: qualitative, rarely quantitative.

C. Method(s) of evaluation: design analysis.

D. Needs and dependencies: the ease of maintenance/modification of the dialogue manager and/or of its individual modules depends on the application and the skill level of the people required to do the maintenance and/or modification.

E. Life-cycle phase(s): early design, final evaluation, maintenance.

F. Importance of evaluation: medium. Important when ease of maintenance/modification of the dialogue manager and/or of its individual modules is required of the application, such as when non-specialist users should be able to easily change the database.

G. Difficulty and cost of evaluation: sometimes difficult. It can be difficult to judge the claims made by the developers and hard to (be allowed to try to) do modifications oneself during evaluation. On the other hand, it is relatively easy to evaluate ease of maintenance when the intended maintainers are non-experts.

5.6 Portability (re-usability) of the dialogue manager and/or of its individual modules.

A. What is being evaluated: the effort needed to port the dialogue manager and/or its individual modules to support different applications. Generic property.

B. Type of evaluation: qualitative, rarely quantitative.

C. Method(s) of evaluation: design analysis.

D. Needs and dependencies: none.

E. Life-cycle phase(s): early design, final evaluation, porting.

F. Importance of evaluation: medium. In many cases of dialogue manager evaluation, portability is not an issue. Can be crucially important, though, especially if one is purchasing a dialogue manager that is claimed to be easily portable to novel applications.

G. Difficulty and cost of evaluation: difficult. The best clue is reliable and detailed information on the person/hours used to port the dialogue manager at hand to a series of relevant and relevantly different applications. It is important to ask exactly what those person/hours did accomplish. In the absence of such information, portability can be rather difficult to judge. The problem is that most dialogue managers are claimed to be portable but mostly without provision of the reliable and detailed information needed.

6. Global issues of dialogue system evaluation

6.1 Average time for task completion as compared to other ways (not using an SLDS) of solving the same task.

A. What is being evaluated: the average time it takes representative users to complete representative tasks using the SLDS as compared to the average time it takes representative users to complete representative tasks in other ways, such as by communicating with a human operator or using interactive modalities other than speech. Specific property.

B. Type of evaluation: quantitative, qualitative, can be externally comparative..

C. Method(s) of evaluation: design analysis, Wizard of Oz data analysis, controlled experiments, field studies. Design analysis and Wizard of Oz data analysis are generally less reliable than user-system interaction analysis but can important for the qualitative evaluation of task completion times prior to deciding if an interactive speech application is worth building.

D. Needs and dependencies: this kind of evaluation assumes the existence of other ways of doing the same task. Such alternatives do not always exist because SLDSs may be introduced for tasks that were simply not done before. Also, competing, non-SLDS system solutions may conceivably exist for solving the task.

E. Life-cycle phase(s): early design, simulation, field evaluation.

F. Importance of evaluation: medium. Can be important to the user's perception of the system. If doing the task using an SLDS takes much longer than what the user is accustomed to, the user's perception could be negative.

G. Difficulty and cost of evaluation: relatively easy to do given representative and realistic samples of user task performance using, resp. not using, the SLDS. One problem, however, is that one might want to know the answer before actually developing the SLDS or its potential alternatives. In such cases, measurement becomes difficult as it has to be based on simulation or mock-ups, introducing elements of qualitative evaluation.

6.2 Average cost per transaction as compared to other ways of solving the same task (not using an SLDS).

A. What is being evaluated: the total cost per transaction using resp. not using an SLDS. Specific property.

B. Type of evaluation: quantitative, can be externally comparative.

C. Method(s) of evaluation: design analysis. Comparison of business plan costs with current business costs.

D. Needs and dependencies: this kind of evaluation assumes the existence of other ways of doing the same task. Average time for task completion (7.1) may contribute to the calculations.

E. Life-cycle phase(s): early design. The comparison can of course be made at any time after early design but is most consequential at this early stage.

F. Importance of evaluation: high. Transaction cost can be a decisive factor when deciding whether or not to build or buy and SLDS.

G. Difficulty and cost of evaluation: from easy to difficult and uncertain, depending on factors such as: how easy is it to decide which costs to include in the comparison; is the application of a novel type or is it well-established technology; what are the uncertainties wrt. development costs, number of final users etc. Even though partly responsible for the enterprise, the present author never figured out exactly how much was saved using a commercialised text translation system as compared to the standard (all-human) way of doing the translations.

6.3 Average number of turns to complete a task compared to human-human interaction.

A. What is being evaluated: how many turns it takes to complete the task on average using the SLDS resp. using human-human interaction. Specific property.

B. Type of evaluation: quantitative.

C. Method(s) of evaluation: Wizard of Oz data analysis, controlled experiments, field studies.

D. Needs and dependencies: the average number of turns needed for task completion is rather uninformative in itself but acquires significance when compared to the number of turns that are needed for task completion in human-human interaction. Another important use of average turn numbers is when these are sub-divided into (a) turns used for domain communication and (b) turns used for repair meta-communication and other undesirable forms of meta-communication. As the proportion of (undesirable) meta-communication rises, the system is likely to be perceived as flawed by its users.

E. Life-cycle phase(s): simulation, field evaluation, final evaluation.

F. Importance of evaluation: high. Is important to the user's perception of the system, which is negatively affected by comparatively lengthy turn sequences as well as by frequent repair meta-communication and other undesirable forms of meta-communication.

G. Difficulty and cost of evaluation: easy to do given a realistic and representative corpus of user task performance dialogues.

6.4 Real-time performance.

A. What is being evaluated: the extent to which the SLDS responds in real-time to the user's input. Specific property.

B. Type of evaluation: quantitative, qualitative, subjective. The qualitative part is to judge the significance of the quantitative findings.

C. Method(s) of evaluation: blackbox test, controlled experiments, field studies, questionnaires, interviews.

D. Needs and dependencies: related in complex ways to how humans solve the same task together. Presumably, users have nothing against an SLDS which is *faster* than a human interlocutor. Users also tend to accept the fact that human operators sometimes need quite some time to solve particular sub-tasks. On the other hand, depending on the task and the contents of a particular user input, there are important limits to the users' patience.

E. Life-cycle phase(s): implementation, field evaluation, final evaluation.

F. Importance of evaluation: high. It is important to make sure that the users' patience is not stretched beyond what is being perceived as acceptable.

G. Difficulty and cost of evaluation: relatively easy and intuitive. Does not require large and representative corpora.

6.5 Transaction success rate.

A. What is being evaluated: the percentage of successful transactions between user and system. Successful transactions are those in which the user achieves the expressed task goal(s) through interacting with the system, i.e. the user gets from the system exactly what the user actually asks for. Specific property.

B. Type of evaluation: quantitative, comparative. Qualitative evaluation should be avoided as far as possible.

C. Method(s) of evaluation: controlled experiments, field studies.

D. Needs and dependencies: none.

E. Life-cycle phase(s): field evaluation, final evaluation.

F. Importance of evaluation: high. Transaction success rate is important to user acceptance.

G. Difficulty and cost of evaluation: relatively easy but somewhat costly. Requires a realistic and representative corpus of user-system interaction. Requires clear definitions of what counts resp. does not count as a 'transaction success'. There are several pitfalls here. The basic idea is that the user should get from the system exactly what the user actually asks for. If the user does not get all of it, or if the user gets something else, the transaction is a failure even if the user gave up before having tried everything. If a task is only completed after a lengthy struggle by a persistent user, it still counts as a transaction success. A user's attempt at completing a

task ends when the user hangs up, the system breaks down or the user goes on to addressing a new task. Other pitfalls concern user requests that are partially beyond what the system can deliver. For such cases, a rational solution must be found to a question such as: if the system gave the user what it could, does that count as a transaction success or not? This is why transaction success evaluation may involve judgments of difficult cases by experts.

6.6 Translation success rate.

A. What is being evaluated: the percentage of successful translations made by spoken language translation (support) systems. Specific property.

B. Type of evaluation: quantitative, qualitative, comparative.

C. Method(s) of evaluation: blackbox evaluation, controlled experiments, field studies.

D. Needs and dependencies: none.

E. Life-cycle phase(s): implementation, field evaluation, final evaluation.

F. Importance of evaluation: high. Translation success rate is important to user acceptance.

G. Difficulty and cost of evaluation: relatively easy but somewhat costly. Requires large test suites and a realistic and representative corpus of user-system interaction. For all but the simplest tasks, translation success rate evaluation has an important qualitative aspect to it. This is due to the fact that utterance translation can be more or less apt and adequate.

6.7 User satisfaction and other subjective parameters (explain).

A. What is being evaluated: users' subjective assessments of SLDS properties, such as: acceptability, satisfactoriness, speed, efficiency, flexibility, politeness, stimulatingness, simplicity, predictability, reliability, desirability, usefulness, future potential, error-proneness, linguistic freedom of the user, ease of making corrections, ease of task performance etc. Generic property.

B. Type of evaluation: subjective.

C. Method(s) of evaluation: questionnaires, interviews. As no standard list of properties for subjective evaluation is available at present, different evaluators tend to use partly different properties in their questionnaires and ask partly different questions in their interviews, even for similar applications.

D. Needs and dependencies: none.

E. Life-cycle phase(s): simulation, field evaluation, final evaluation.

F. Importance of evaluation: high. User perception is a crucial factor in determining the fate of an SLDS.

G. Difficulty and cost of evaluation: mostly difficult, not inexpensive if many external users are involved. Anyone can write a questionnaire or conduct a series of interviews. However, there are no standards for the questions they should contain and it is often hard to interpret their results. More research is needed to identify to which questions users are likely to produce the most informative answers.

6. Concluding Discussion

It seems obvious that a long list of 35 evaluation criteria for dialogue managers is potentially counter-productive. It might be viewed as arguing that, rather than relying on a small and relatively arbitrary set of dialogue manager evaluation criteria as is currently the case, developers should start spending very considerable resources on evaluating their dialogue managers from 35 different points of view whenever appropriate throughout the life-cycle. What is being recommended here is something else, however. Firstly, no existing dialogue manager has all the *possible* properties listed in the DISC dialogue manager processing model (Section 2). For instance, no current commercial system, at any rate, seems to be using indirect speech act recognition. So their developers need not worry about getting indirect speech act recognition right. Secondly, the template presented in Section 3 contains entries which address the importance of evaluation as well as its difficulty and cost. The purpose of these entries is to assist developers in focusing their evaluation efforts on the most important properties of their dialogue manager, including those whose evaluation targets should be included in the requirements specification. It is hoped that these entries will provide helpful guidelines for the focusing process.

Still, we are dealing with a large number of evaluation criteria for potential use. However, the solution to that problem, it appears, is not to ignore evaluation of important dialogue manager properties but to intensify the work on early design support tools which can significantly reduce the amount of errors that would otherwise have to be identified, diagnosed and remedied as a result of evaluation. This work should primarily focus on (a) highly important dialogue manager properties for which (b) current evaluation tends to happen late in the life-cycle and which (c) are relatively costly to properly evaluate.

Some observations on the list in Section 5.3 are that (i) the list of evaluation criteria provides a good illustration of the complexity of dialogue manager evaluation, reflecting the core "hidden" role of the dialogue manager in many SLDSs. (ii) Few of the evaluation criteria in the list are straightforwardly quantitative, at least for the time being. (iii) Some of the criteria which are quantitative, cannot be applied to the SLDSs performance as a whole but must be applied diagnostically, for instance by inspecting interaction logfiles to see whether, e.g., the dialogue manager adequately supports the performance of correct co-reference resolution or ellipsis processing. (iv) Many of the criteria are qualitative. (v) Some criteria, and not the least important ones, are at least partly subjective and must be measured through interviews, questionnaires and other contacts with the users to elicit those among their subjective impressions from interacting with the system that may be attributed to the workings of the dialogue manager.

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

Furthermore, the list is incomplete in at least one important respect: it does not include the, often quantitative, criteria which may derive from particular constraints on a given SLDS, such as that the average user utterance length should be, say, four words maximum. To include such

constraints would make the list unnecessarily complex and overly specific. In developing one's dialogue manager, one may measure many different things which it would make no sense to measure in connection with another dialogue manager development project which is subject to very different constraints.

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

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

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

- in the *third iteration,* the developer plans when to apply the specified criteria during the development process. A checklist is produced;

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

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

- did the developers select all the right evaluation criteria?

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

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

- what were the results?

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

The work presented in this paper obviously needs to be continued in order to (i) test the adequacy of the evaluation template, (ii) test the adequacy of the 35 filled templates, (iii) test whether the presented template, in some revised version, could be used for all aspects of SLDSs. Tests will be done firstly within DISC and secondly with developers and deployers of SLDSs and components outside DISC.

7. Acknowledgements

The work presented was done in the Esprit Long-Term Concerted Action No. 24823, DISC (1997-1999): Spoken Language Dialogue Systems and Components: Best practice in development and evaluation. The support is gratefully acknowledged. I also want to warmly thank Laila Dybkjær, Uli Heid and Jan van Kuppevelt for their comments on an earlier draft, and the DISC partners who have demonstrated their systems and provided detailed information for the grid and life cycle analyses.

8. References

[Bernsen et al. 1998a] Bernsen, N. O., Dybkjær, L., Heid, U., van Kuppevelt, J., Dahlbäck, N., Elmberg, P. and Jönsson, A.: Working Paper on Dialogue Management Current Practice. DISC Deliverable D1.5, 1998.

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

[Kuppevelt et al. 1999] van Kuppevelt, J., Heid, U., Dybkjær, L. and Bernsen, N. O.: The DISC method for describing, comparing and evaluating spoken language dialogue systems. Paper to appear in *IEEE Transactions on Speech and Audio Processing.*