Deliverable D3.8a

DISC Dialogue Engineering Best Practice Methodology

July 1999

Esprit Long-Term Research Concerted Action No. 24823

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

DISC

TITLE	**D3.8a DISC Dialogue Engineering Best Practice Methodology**
PROJECT	DISC (Esprit Long-Term Research Concerted Action No. 24823)
EDITORS	-
AUTHORS	Niels Ole Bernsen and Laila Dybkjær (NIS) nob@nis.sdu.dk, laila@nis.sdu.dk
ISSUE DATE	31.7.1999
DOCUMENT ID	wp3d3.8a
VERSION	2
STATUS	Final
NO OF PAGES	41
WP NUMBER	3
LOCATION	http://www.disc2.dk
KEYWORDS	WP3, spoken language dialogue systems, dialogue engineering, best practice, development, evaluation

Document Evolution

Version	Date	Status	Notes
1	12.5.99	Draft	First draft published for review from partners.
2	31.7.99	Final	

DISC Dialogue Engineering Best Practice Methodology

Niels Ole Bernsen and Laila Dybkjær

Natural Interactive Systems Laboratory
University of Southern Denmark - Odense

Abstract

This DISC Working Paper D3.8a presents an overview of, and a reading guide to, the DISC work package 2 and work package 3 deliverables. It is the successor to deliverable D1.8 which served as a reading guide to and summary document of DISC work package WP1 [Heid et al 1998]. The report starts by reviewing WP1 on current practice in spoken language dialogue systems and components development and evaluation. It continues by presenting the state-of-the-art surveys of existing tools as well as the best practice tools developed by DISC partners themselves in WP2. The report then presents the WP3 best practice drafts for the six DISC aspects: speech recognition, speech synthesis, language understanding and generation, dialogue management, human factors, and system integration issues. Finally, some major remaining challenges to be addressed in DISC are discussed.

Contents

1. Introduction

DISC aims to develop a first best practice methodology for spoken language dialogue systems (SLDSs) and components.

This document, DISC Deliverable D3.8a, primarily serves as a reading guide to, and summary document of, the DISC Work Packages WP2 and WP3 and thus is a successor to DISC Deliverable D1.8 which served as a reading guide to, and summary document of, DISC Work Package WP1 [Heid et al 1998]. However, as D3.8a is the final deliverable of DISC, it seems appropriate to start by reviewing WP1 as well as those parts of the DISC approach which have been adhered to throughout (Section 2). Following that, an overview is provided of WP2 and WP3 including the changes that were made to the original DISC Workplan (Section 3). Sections 4 and 5 then provide a reading guide for WP2 and WP3, respectively. Section 6 concludes the document by presenting the challenges which are currently being addressed in the successor project to DISC, DISC-2. Appendix 1 presents the proposed best practice grids for each aspect. A full life-cycle is presented in Appendix 2. The DISC evaluation template and the evaluation criteria per aspect are shown in Appendix 3.

2. DISC Basics and Current Practice

This section briefly presents the DISC basic analytical apparatus (Section 2.1) and the DISC analysis of current practice in SLDSs development and evaluation (Section 2.2).

2.1 DISC Basics

To provide a thorough analysis of a complex software system, such as an SLDS, it is practical and useful to take the logical system architecture as a starting point, breaking up the complex system into aspects which can be more easily described individually, as well as with respect to their interaction.

The different aspects are represented as a series of layers in Figure 1, called performance, speech, language, control, and context, respectively. Depending on the individual system at hand, some of these layers indeed correspond, more or less, to components or modules of the system. This is true, in particular, of speech recognition, natural language analysis, dialogue management, natural language generation and speech synthesis. At a more abstract level, any designer of an SLDS has to deal with issues of user interaction, cooperativity or, more generally, human factors, as well as, on the technical side, with questions of system integration, system architecture, and the interaction between individual components.

Taking into account the fact that with many SLDSs, there is a certain asymmetry between the ‘analysis’ side and the ‘generation’ side, we have combined the language analysis and generation components into one aspect. Thus, whilst many systems include very sophisticated components for the analysis of user input, they tend to have much less sophisticated generation facilities which are often based on a set of more or less pre-constructed system reactions. By consequence, we have divided up our analytical work on SLDSs into the following sub-areas, the first four of which roughly correspond to the speech, language, and control, and context layers contained in Figure 1, or to the corresponding components or subsystems. Very roughly, the fifth area corresponds to the performance layer of Figure 1:

1. speech recognition;

2. speech synthesis;
3. language understanding and generation;
4. dialogue management;
5. human factors;
6. system integration issues.

The listed areas are the six SLDSs *aspects* addressed in DISC. Although not really embodied in individual system components, analysis from the point of view of human factors as well as analysis of architectural problems, implementational issues and questions of system integration have been included among the SLDSs aspects. Both aspects cut across all or most components and are of particular importance when it comes to assessment of an SLDS as a whole.

In addition to the logical system architecture shown in Figure 1 and the six SLDS aspects, DISC is based on the software engineering life-cycle model, adapted for the purpose of the analysis of SLDSs, sketched in Figure 2.

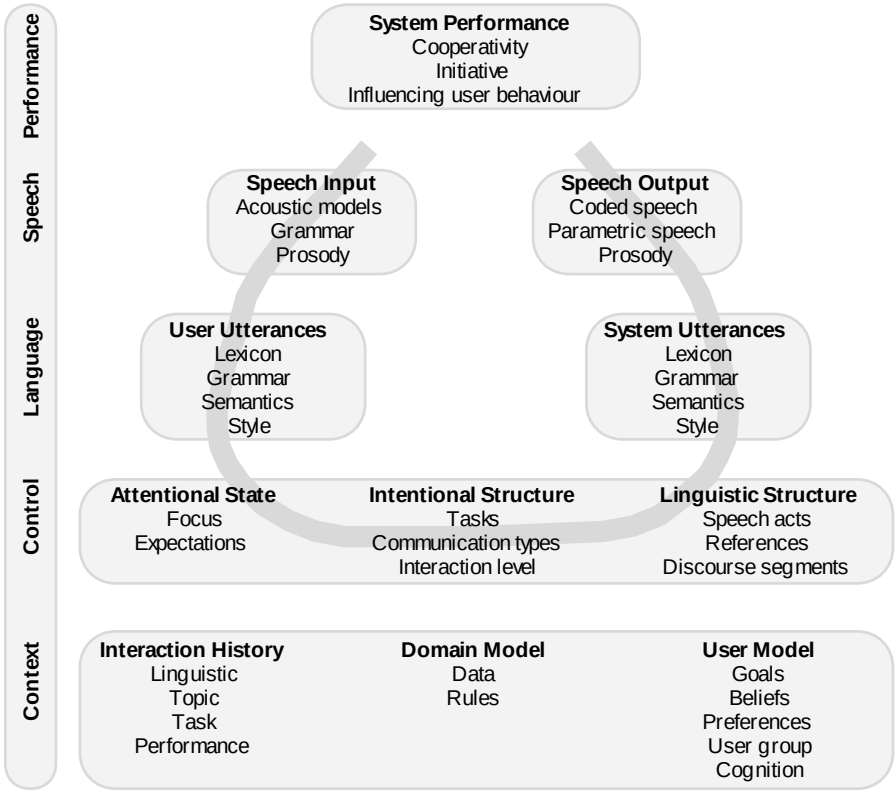

Figure 1. Elements of an interactive speech theory [Bernsen et al 1998]. Element types are shown in boldface. The dark grey band and the grey boxes reflect the logical architecture of interactive speech systems.

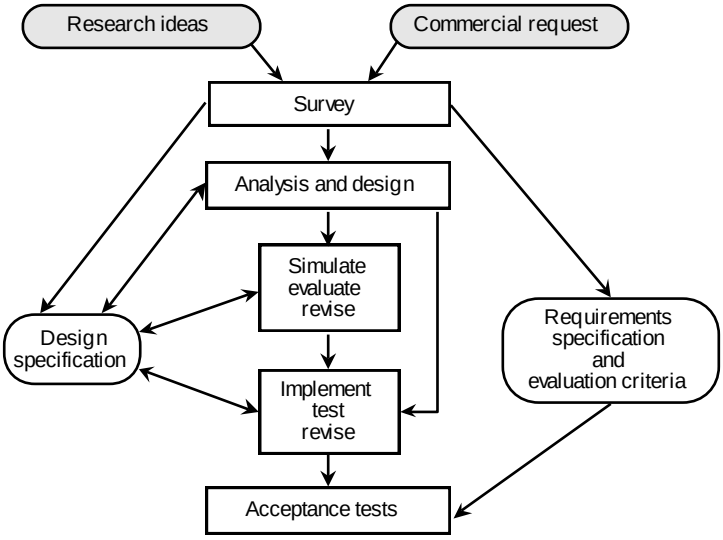

Figure 2. A software engineering life-cycle model for the development and evaluation of interactive speech systems [Bernsen et al. 1998]. Rectangular boxes show process phases. The development and evaluation process is iterative within each phase and across phases. Arrows linking process phases indicate the overall course of the process. The requirements specifications and evaluation criteria, and the design specification (rounded white boxes) are used throughout the development process. The rounded grey boxes indicate that the system to be developed may be either a research system or a commercial system.

Looking at both the development of an SLDS and its evaluation with the software engineering life-cycle model in mind, analysis of an individual SLDS must take two dimensions into account:

1. the choices which have been made in the design and implementation of the individual layers and components of the system; and
2. the way in which the development and evaluation of the system were carried out.

The former can be considered mainly technical information about the system, whereas the latter is information about the working procedures, the steps followed in the design and implementation of the system, design goals, constraints, and procedures relevant to development and evaluation practice.

In the terminology of DISC, we call the first analytical dimension a *grid* analysis of an SLDS and the second dimension a *life-cycle* analysis of an SLDS. Note that the grid analysis includes human factors (cf. the six aspects mentioned earlier in this section).

To summarise, according to DISC, any SLDS can be analysed in depth by mapping out its grid and its life-cycle. How this was actually done is described in Section 2.2.

2.2 The DISC Current Practice Analysis

The purpose of the first DISC work package (WP1) was to provide an overview of current industry and research practice in the development and evaluation of SLDSs and components.

The DISC current practice approach was to (a) analyse a broad range of SLDSs and components with respect to the six key aspects mentioned above, and (b) map out their respective development and evaluation processes. In order to adequately capture current

practice and overcome various problems primarily relating to insufficient and incomparable information provided for individual systems and components, a common scheme was developed. The scheme consists of the 'grid' and a life-cycle model, mentioned in Section 2.1, both of which are slot-filler structures. The DISC 'grid' enables an in-depth characterisation of the properties of any SLDS or SLDS component. The life-cycle model focuses on the development and evaluation process for SLDSs and their components. The point of departure were the grid and the life-cycle issues presented and discussed in [Bernsen et al. 1998], cf. also Figures 1 and 2. These issues were further developed in WP1 in an iterative refinement process based on exemplar analyses per aspect. Observations from the exemplars analysed contributed to issue refinement as well as to the definition of additional issues, and eventually produced a comprehensive overview of current practice in the development and evaluation of SLDSs and components.

The DISC current practice overview was developed as shown in Figure 1 and explained below.

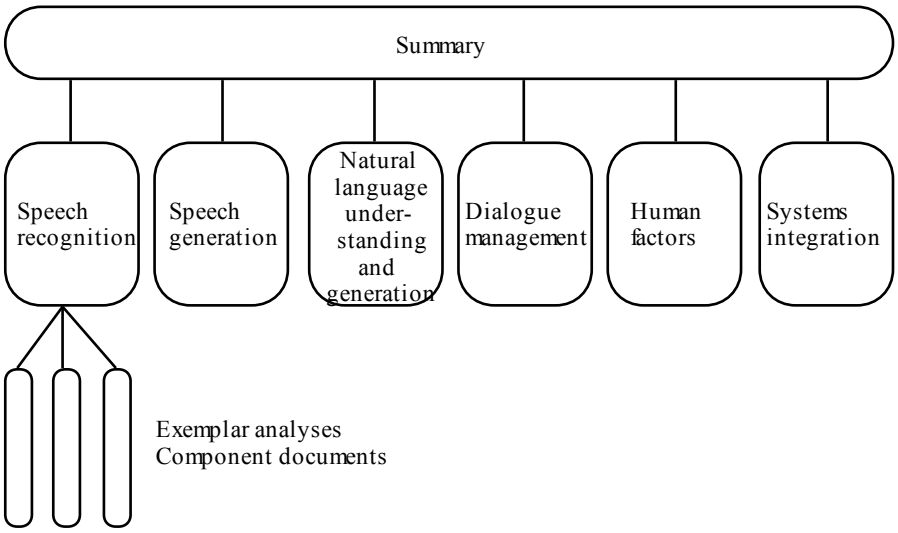

Figure 3. The DISC development of current practice grids and life-cycles [Heid et al 1998]**.**

• For each aspect, a *synthesis working paper* was produced (middle layer of Figure 3). Abstracting from individual observations, each synthesis paper spans the entire range of design and technology choices at hand for the key issues encountered in the exemplars analysed (the grid model), and shows the range of practical approaches followed in the development and evaluation of systems and components (the life-cycle model).

• Each synthesis paper is based on several *exemplar case studies* (about 50 in total), which serve as detailed background information (bottom layer of Figure 3). Most case studies address both grid and life-cycle issues. As a rule, a case study report contains a brief description of the system or component, answers to the grid questions for the system aspect addressed, answers to the life-cycle questions for that aspect, and some concrete examples, such as (annotated) traces, sample dialogues, dictionary entries etc.

• Finally, a *reading guide and summary* document for the synthesis papers was created (top layer of Figure 3). The summary also outlines some general trends in the practical working

procedures followed in development and evaluation, which were observed in the synthesis papers.

Despite the fact that a wealth of scientific papers and reports have been produced on individual SLDSs, it is not a trivial task to produce an overview of current practice in the development and evaluation of SLDSs.

One reason is that the diversity of systems requires a comprehensive descriptive apparatus for adequately dealing with systems ranging from, e.g., call routing and information systems (e.g. of the ATIS type) to more complex systems designed to handle several types of tasks. The grid and life-cycle were developed for this purpose.

A second cause of difficulty is the lack of documentation in the field. The standard published scientific papers and reports mentioned above tend to be radically incomplete as regards important system/component properties and design decisions. It often proved difficult to collect the information needed to complete the grids and life-cycles. Collaboration with the system developers generated much more information about current practice in SLDS development and evaluation than could be gathered from the literature alone.

All analysed exemplars were provided by the DISC partners. The exemplars that were analysed with respect to one or more aspects were: the French LE Arise system for telephone-accessed train time-table information [den Os et al. 1999], the CMU Phoenix parser [Ward and Issar 1995], the Daimler-Benz dialogue manager [Heisterkamp and McGlashan 1996], the Daimler-Benz parser [Mecklenburg et al. 1995], the Danish Dialogue System for flight ticket reservation [Bernsen et al. 1998], the Vocalis Operetta automated call routing system [Fraser et al. 1996], the Vocalis Voice Activated Dialling system [http://www.vocalis.com/products/speechtel/ infoframe.html], the Verbmobil spoken language dialogue translation system [http://www.dfki.de/verbmobil/], and the multimodal Waxholm tourist boat information system [http://www.speech.kth.se/ waxholm/waxholm.html].

To ensure soundness of methodology, each aspect was analysed by at least two different DISC partners. For each aspect, at least three significantly different exemplars were investigated. No aspect of a system or component was analysed by a partner who had been involved in its development and evaluation. Every analysis of an aspect of an SLDS or component was verified by the developers of that particular SLDS or component.

The actual analysis of an aspect of a particular system or component consisted in applying the 'grid' and life-cycle models to the description of that particular exemplar. Typically, first versions of the grid and the life-cycle were completed on the basis of available papers and reports. This first iteration always generated a - sometimes quite large - number of questions which could not be answered with sufficient certainty, if at all, based on the initially collected information. Answers were then sought through, i.a., email or telephone interaction with colleagues who had been involved in the development and evaluation of that particular system/component aspect, access to additional data, such as transcriptions and recordings of user-system interactions, and site visits, interviews and demonstrations. In fact, site visits proved necessary to the satisfactory analysis of most DISC exemplars. The final step in the analysis of an aspect of a system or component was to invite verification from that system or component's developers in order to remove any misconceptions from the grid and life-cycle representations.

After several iterations, the grid and life-cycle models proved reasonably stable for carrying out the 50 exemplar analyses made. A basis had been created for compatible and comparable description of systems and components at each level and across levels.

3. Best Practice Overview

Once a current practice overview had been established for SLDSs and components development and evaluation, the next step was to draft a best practice proposal. The DISC best practice draft was developed as shown in Figure 4 and explained below.

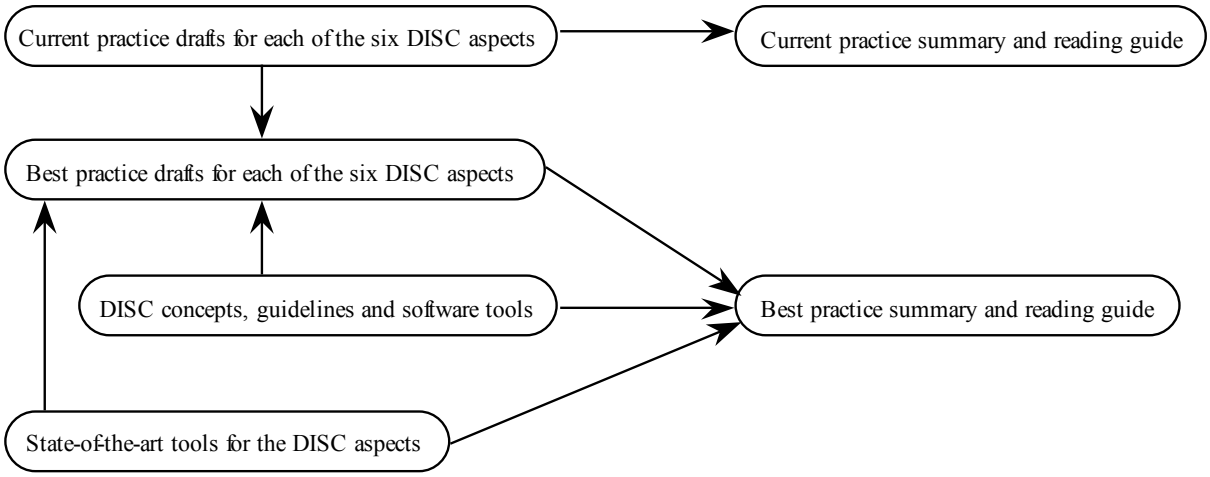

Figure 4. The DISC development of a draft best practice dialogue engineering methodology.

1. For each aspect except systems integration, existing state-of-the-art tools were reviewed and an overview created of which parts of SLDSs and components development and evaluation could be supported by tools.

2. Three hypertext tools accessible over the web have been built in DISC. One is a tool for the evaluation of speech synthesis components of SLDSs. Another is a tool in support of deciding whether or not to use speech and whether or not to use speech in combination with other modalities for the representation and exchange of information. A third tool supports development and evaluation of cooperative spoken dialogue design.

3. Based on the work described in (1) and (2), and the current practice drafts for the six DISC aspects (grids and life-cycles), a best practice proposal for each aspect was established.

4. Finally, a *reading guide and summary* document for the best practice papers was created (this report).

For each DISC aspect, the draft DISC dialogue engineering best practice methodology essentially consists of a grid, a life-cycle, and a set of evaluation criteria which represents an expansion of the life-cycle. The methodology incorporates several scientific and methodological innovations (see below).

Appendix 1 presents the draft best practice grids for each aspect. Grid questions are aspect-specific and tend to be hierarchically structured. This has to do with the fact that SLDSs and components design is typically hierarchically structured in the sense that decision to include some property may exclude other properties and at the same time make yet other properties candidates for inclusion. To achieve a common presentation across aspects, and to support the readability of the best practice grids proposals, a new piece of DISC methodology was developed. This is the structuring of grids in terms of *issues, options and pros and cons* per option [van Kuppevelt and Heid 1999]. SLDSs developers can look up a particular issue of

interest, inspect the technical options available for that issue, and bring the option-related pros and cons to bear on their own problem.

The life-cycle has a flat structure and would appear to apply across aspects, in most cases irrespective of whether the object of analysis was an SLDS or an SLDS component. A full life-cycle is presented in Appendix 2. All its entries are relevant to the component-oriented aspects whereas several entries relate to human factors issues only indirectly. This is because human factors are not a component but, rather, a set of cross-component perspectives. Focus is on tasks and users rather than on technology and system.

Evaluation is an important part of the life-cycle. In DISC, considerable effort has been invested in identifying and make explicit the particular evaluation criteria which should, or could, be used to evaluate SLDSs and components. As the individual evaluation criteria are highly aspect-dependent, it was decided to separate aspect-dependent evaluation from the aspect-independent life-cycle. The generation of evaluation criteria for SLDSs and components incorporates another innovative DISC approach. For each aspect of SLDSs, evaluation criteria are being systematically generated from the best practice grid issues, thereby ensuring that evaluation can be achieved for all relevant properties of SLDSs and their components, including human factors. To support proper application of the evaluation criteria, a generic evaluation template was created which provides a common structure for describing each individual evaluation criterion. This approach to evaluation has so far been completed for dialogue management and human factors, and is currently being extended to all aspects. The evaluation template and the evaluation criteria per aspect are shown in Appendix 3.

A final element in the DISC best practice methodology currently is the subject of experimentation. It is the development of *checklists* for each aspect. Checklists provide brief summarising overviews of the issues involved per aspect.

4. Surveys and Tools

Originally, WP2 was planned to focus on concepts and their underlying theories, and on software tools development in support of best practice development and evaluation of SLDSs and components. The idea was to further develop concepts and software tools which were already under consideration at the partners' sites. Early in the project is was realised that it would be useful to SLDS developers as well to include overviews of existing software support tools per aspect. Moreover, the surveys would provide a good background against which to measure the tools created by DISC and their importance to best practice. Thus it was decided to include among the WP2 deliverables state-of-the-art surveys of existing tools for each aspect except systems integration which was not included in WP2. The surveys are [Chase, Lamel and Paroubek 1998, Karlsson 1999a, Heid and Markidou 1999, Luz 1999, Dybkjær and Failenschmid 1999].

In addition to those five state-of-the-art reports, WP2 has delivered three software tools and several reports.

For the speech generation aspect, a software tool for evaluation of speech synthesis components in SLDSs has been produced. This tool is a web-based test protocol for the selection of speech synthesis components. It contains a number of questions that are relevant for choosing between different speech synthesis systems for an SLDS and pin-points important issues to consider.

For the human factors aspect, two software tools have been implemented. One tool, SMALTO, is a web-based design support tool which provides advice to system designers on

whether or not to use speech input and/or output, possibly in combination with other modalities. This is a non-trivial decision due to the large amount of domain variables involved. Because of the complexity empirical experimentation cannot solve the problem at a general level. Instead, modality properties are taken as the point of departure. A large number of claims about speech functionality are collected in a database. Users can extract information about these claims from the database obtaining examples on, e.g., why speech input is well-suited for a given purpose and which modality properties justify that the claim is correct. The tool is intended for use during early design.

The second human factors tool, CODIAL, is also web-based. It supports SLDS developers in designing cooperative dialogue. The central part of the tool is a set of cooperativity guidelines which should be followed in order to ensure smooth and cooperative system dialogue design. A tutorial is offered on how to use the guidelines. Moreover, it is possible to extract information about violations of the guidelines from three annotated corpora. This allows the user to draw on rich example data during the process of learning how to use the guidelines. The tool is primarily intended for use during early design but may also be used for evaluation purposes later on.

The tools are available at the DISC web site at http://www.disc2.dk. Each tool is accompanied by one or two reports. The report describing the speech generation tool [Karlsson 1999b] corresponds to the textual contents of the tool. SMALTO is accompanied by two reports. One is a specification of the tool [Bernsen and Luz 1999], the other is a description of the collection and analysis of the data which forms the basis of SMALTO, i.e. the claims on speech functionality [Bernsen and Dybkjær 1999a]. The report on CODIAL [Dybkjær 1999] provides a specification of the tool, including architecture, use and functionality.

Finally, two WP2 reports describe underlying concepts and guidelines rather than an actual software tool. One of these is a report on guidelines and testing protocols for the development of speech recognition components for SLDSs [Chase, Lamel and Paroubek 1999a]. It reviews fundamental issues in using a corpus to evaluate a speech recogniser. The second report [Heid 1999] is on guidelines for the acquisition of lexical data for SLDSs. This report includes recommendations for lexical acquisition both with respect to functions of acquisition tools and to relevant properties of the SLDS for which the acquisition is supposed to take place.

5. Best Practice Drafts

Based on the WP1 current practice reports and the skeleton DISC dialogue engineering model [Bernsen et al. 1998, Heid et al 1998], the step towards a DISC best practice methodology was briefly addressed in a report by the end of the first year of DISC [Karlsson 1998]. Building on current practice and input from WP2, a series of reports were then produced during the second year of DISC addressing best practice for each of the six DISC aspects [Chase, Lamel and Paroubek 1999b, Karlsson 1999c, Bernsen and Dybkjær 1999b, Failenschmid, Williams, Dybkjær and Bernsen 1999, Failenschmid and Chase 1999]. It should be noted that D3.4 [Heid 1999, to appear] has not yet appeared by the time of writing.

In principle, we now believe, a best practice methodology for an aspect should contain all of the following: a grid, a life-cycle, a set of evaluation criteria, a checklist, a glossary, and a list of references. However, as this view has only recently emerged, none of the best practice drafts currently include all these elements.

The five aspect reports which have appeared so far all include a grid. Moreover, they use a common format for the description of the best practice grid structured around a series of design issues, different options for their implementation, and the possible pros and cons per option as proposed and described in [van Kuppevelt and Heid 1999]. This working paper is a companion to the present report. An overview of the five available grids is given in Appendix 1.

[Karlsson 1999c, Bernsen and Dybkjær 1999b, Failenschmid, Williams, Dybkjær and Bernsen 1999] all contain a life-cycle description which is a revised and extended version of the current practice descriptions and which basically follows the general structure proposed in [Heid et al 1998], see also Appendix 2. [Chase, Lamel and Paroubek 1999b] has a somewhat differently structured life-cycle description which roughly builds on a division of the development process into a specification phase, a development phase and a deployment phase. Moreover, development team staffing is discussed. [Failenschmid and Chase 1999] does not include a life-cycle description. At a recent meeting it was decided to use the life-cycle presented in Appendix 2 as a general frame of reference for future work on life-cycle descriptions.

By the end of year one it was decided, partly based on input from the DISC Advisory Panel (AP), to emphasise evaluation much more than originally planned. It was felt that evaluation needs considerably more attention in a best practice approach than it receives in today's current practice. This increased focus on evaluation was handled in different ways for the individual aspects. [Chase, Lamel and Paroubek 1999b] on speech recognition best practice simply included a reference to [Chase, Lamel and Paroubek 1999a] on guidelines and testing protocols for the development of speech recognition components for SLDSs. [Karlsson 1999c] on speech generation best practice and [Failenschmid, Williams, Dybkjær and Bernsen 1999] on human factors best practice both added extra sections on evaluation to the best practice report. For dialogue management a separate report was made on evaluation [Bernsen 1999]. Finally, the report on systems integration best practice [Failenschmid and Chase 1999] does not include much on evaluation. However, a set of evaluation criteria has later been established and are presented in Appendix 3 together with evaluation criteria for the other DISC aspects. In [Bernsen 1999], a draft generic template for the description of evaluation criteria was established and used for the description of dialogue manager evaluation criteria. This template was later revised on the basis of input from [Failenschmid, Williams, Dybkjær and Bernsen 1999] in which it was applied to human factors evaluation criteria. The revised template is presented in Appendix 3, section 3.7. It contains ten entries to be described for each evaluation criterion, including, e.g., a description of what is being evaluated, which part(s) of the SLDS is being evaluated, and which method is being used.

Two best practice reports [Failenschmid, Williams, Dybkjær and Bernsen 1999, Failenschmid and Chase 1999], contain checklists, i.e. brief list of points to check with respect to a particular aspect when developing SLDSs or components. The checklist in [Failenschmid and Chase 1999] is rather sketchy. However, the DISC consortium and its AP members found the idea of including checklists a very good one. A checklist is brief (at most a couple of pages) and provides an overview of important points to remember. Thus it was decided that future best practice descriptions should include a checklist for each aspect.

All best practice reports include a list of references but none of them has a glossary. The creation of a common DISC glossary was only discussed recently. It was decided to include a common DISC glossary in future work, covering and explaining terminology relating to the six DISC aspects. This addition is expected to be useful to users of the DISC best practice engineering methodology.

In addition to aspect-related draft best practice proposals, WP3 has produced the following two reports.

[van Kuppevelt and Heid 1999] presents IDEM, i.e. the integrated DISC evaluation model. The model takes a particular exemplar grid description as its point of departure and generates from it a set of evaluation questions, as described above. The idea is to facilitate easy creation of a uniform and state-of-the-art-complete set of evaluation questions tailored to a specific application.

Finally, an assessment report has been written by the DISC Advisory Panel members on the DISC best practice methodology and toolbox, i.e. the DISC WP2 and WP3 deliverables [DISC Advisory Panel 1999]. Unfortunately, due to the delayed completion of some working papers, not all WP2 and WP3 deliverables were available to the Advisory Panel at the time of writing, and some were only available in an unfinished draft version. Thus, these deliverables are either not evaluated at all or only partially evaluated in the assessment report. Seventeen of 46 Advisory Panel members kindly answered a series of questions concerning DISC, its purpose, deliverables, tools and packaging, including comments on the individual deliverables. The DISC Consortium is grateful for the advice offered and will take it into account during DISC-2.

6. Concluding Discussion

As indicated in Section 5, there is still some best practice work to be done for the individual aspects, and the DISC best practice drafts need to be tested. These challenges are currently being addressed in the successor project to DISC, DISC-2. The missing best practice descriptions in terms of life-cycles, evaluation criteria and their thorough description following the proposed evaluation template, the checklists, glossary items, and possibly additional references are now being prepared in a version suitable for being presented at the DISC web site [http://www.disc2.dk]. This is the first step towards a packaged version of the DISC best practice dialogue engineering model. The web-based draft version will be tested by DISC partners and, hopefully, by the AP members as well in order to provide input to an improved, final, and well-presented version of the DISC best practice dialogue engineering model.

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. We would also like to thank all DISC partners for their contributions to the present overview paper.

References

Bernsen, N. O.: Working Paper on Dialogue Management Evaluation. Deliverable D3.10, 1999.

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

Bernsen, N. O. and Dybkjær, L.: Working Paper on Speech Functionality. Deliverable D2.10, 1999a.

Bernsen, N. O. and Dybkjær, L.: Draft Proposal on Best Practice Methods and Procedures in Dialogue Management. Deliverable D3.5, 1999b.

Bernsen, N. O. and Luz, S.: SMALTO: Speech Functionality Advisory Tool. Deliverable D2.9, 1999.

Chase, L., Lamel, L. and Paroubek, P.: Survey of Platforms and Methods for Speech Recognition. Deliverable D2.1, 1998.

Chase, L., Lamel, L. and Paroubek, P.: Guidelines and Testing Protocols for the Development of Speech Recognition Components for Spoken Language Dialogue Systems. Deliverable D2.2, 1999a.

Chase, L., Lamel, L. and Paroubek, P.: Draft Proposal on Best Practice Methods and Procedures in Speech Recognition for SLDSs. Deliverable D3.2, 1999b.

den Os, E., Boves, L., Lamel. L. and Baggia, P.: Overview of the ARISE project. Proceedings of Eurospeech, Budapest, 1999, 1527-1530.

DISC Advisory Panel: Assessment Report on the DISC Best Practice Methodology and Toolbox. Deliverable D3.9, 1999.

Dybkjær, L.: CODIAL, a Tool in Support of Cooperative Dialogue Design. Deliverable D2.8, 1999.

Dybkjær, L. and Failenschmid, K.: State-of-the-Art Survey of Existing Human Factors Tools. Deliverable D2.7b, 1999.

Failenschmid, K. and Chase, L.: Draft Proposal on Best Practice Methods and Procedures in Systems Integration. Deliverable D3.7, 1999.

Failenschmid, K., Williams, D., Dybkjær, L. and Bernsen, N. O.: Draft Proposal on Best Practice Methods and Procedures in Human Factors. Deliverable D3.6, 1999.

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

Heid, U.: Towards Guidelines for the Acquisition of Lexical Data in Spoken Language Dialogue Systems. Deliverable D2.6, 1999.

Heid, U.: Draft Proposal on Best Practice Methods and Procedures in Natural Language Understanding and Generation. Deliverable D3.4, 1999. To appear.

Heid, U., Bernsen, N. O. and Dybkjær, L.: Current Practice in the Development and Evaluation of Spoken Language Dialogue Systems. Deliverable D1.8, 1998.

Heid, U. and Markidou, E.: Survey of Concepts, Methods and Tools for the Acquisition of Lexical Data in Spoken Language Dialogue Systems. Deliverable D2.5, 1999.

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

Karlsson, I.: Working Paper on the DISC Dialogue Engineering Best Practise Methodology. Deliverable D3.1, 1998.

Karlsson, I.: A Survey of Existing Methods and Tools for Development and Evaluation of Speech Synthesis and Speech Synthesis Quality in SLDSs. Deliverable D2.3, 1999a.

Karlsson, I.: Software Tool for Evaluation of Speech Synthesis Components in SLDSs. Deliverable D2.4, 1999b.

Karlsson, I.: Draft Proposal on Best Practice Methods and Procedures in Speech Generation. Deliverable D3.3, 1999c.

Luz, S.: State-of-the-Art Survey of Dialogue Management Tools. Deliverable D2.7a, 1999.

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

van Kuppevelt, J. and Heid, U.: From a Description of Spoken Language Dialogue Systems to their Evaluation and Best Practice. DISC Deliverable D3.8b, 1999. Based on Jan van Kuppevelt: A Proposal for an Integrated DISC Evaluation Model. November 1998.

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

DISC: http://www.disc2.dk/

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

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

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

Appendix 1: Draft Best Practice Grids

1.1 Draft Best Practice Grid for Speech Recognition

Overall issue	Issue	Option
Telephony Interface	Telephony network board	Required for telephony applications to ensure synchronisation between the switch and the SLDS components. Can be analogue or digital.
	Archiving process	Necessary for system tuning, development and evaluation
	Barge-in	Necessary for naturalness and to support "skip" commands so that users don't get frustrated by long playback messages.
	Network Integration	Through custom work, PBX/PABX or Intelligent Network, depending on what is required for a given application.
Signal Processing	Sampling rate	Must be at least twice the rate of the highest frequency modelled accurately by the system. Two main options are 8000 and 16000 samples per second.
	Frame rate	10 msec intervals with up to 50% overlap between frames is standard.
	Cepstral parametrisation of frames	Number of coefficients used: 13 or 14 plus a power term is typical. Method of calculation: direct or via LPC coefficients are common. Inclusion of up- and down-stream information such as differences and double differences or multi-frame averages. Use of one or multiple parallel streams of information.
	Calculation of Pitch/Tones	Choice of whether or not to include the information.
	Principle Components Analysis	Choice of whether or not to perform this rotation after cepstral calculation.
Phone Set	Phone set size	The base phone set generally contains between 35 and 50 units.

	Silence model	A silence model is absolutely necessary. Options include: whether or not to specialise begin and end silence from the between-word model, and whether to combine the silence model with "garbage" or noise models.
	"Garbage" model	Whether or not to include a "not speech" model.
	Noise models	Whether or not to train special "not speech but not garbage" models for sounds such as lip smacks, coughs and laughing.
	Context-triggered phones	Whether or not to include context-triggered phones in the base phone set.
Dictionaries	Multiple pronunciations	Can allow words to have more than one alternate pronunciation
	Phrases as entries	Can allow short phrases to play the logical role of a single word in the system
	Automatic pronunciation generation	Can generate pronunciations automatically rather than with the help of online dictionaries or hand editors.
	Multiple dictionaries	Can use multiple dictionaries such as a general base dictionary plus a task specific one.
Acoustic Models	HMM topology	3-state, 5-state or variable topologies possible
	Output distribution modelling	Discrete vector quantization *vs*. either continuous or semi-continuous mixture Gaussian modelling
	Number and distribution of Gaussians	Even, fixed number distribution of Gaussians: Typically use 16 or 32 Gaussians per mixture model for simplicity *vs*. More "intelligent" distribution of free parameters: Semi-automatic or automatic assignment of Gaussians to mixture model
	Explicit duration modelling	Inclusion of "add-on" duration model for phones, typically based on a single-parameter gamma function.
	Covariance matrix modelling	Diagonalisation of covariance matrices in Gaussian models.
	Viterbi vs. "alpha" training	Use of best-scoring path vs. all possible paths in training output distributions and transition probabilities.

	Context-dependent phones	Whether or not to include triphones and/or quinphones in building acoustic models.
	Clustering of triphone/quinphone states	Whether to "cover" unseen triphones and/or quinphones by clustering with other similar models or by using backoff to diphones and monophones.
	Vocabulary-specific phones	Whether or not to train special versions of phones for specific vocabulary words such as digits.
	Amount of training data	Minimum amount for task is absolutely necessary. Must come from a wide variety of speakers, no more than a hundred or so utterances from each. For small, simple tasks such as digits, connected digits, yes/no, command word lists of under 50 in length: 5000 utterance for each task, 3000 for training, 1000 each for development and independent test For large vocabulary tasks: minimum of 100 hours of training material for tasks such as broadcast news material. With this level of training material, there should be at least 10 hours each of development and independent test material.
	Source of training data	Purchase *vs.* collect-it-yourself
	Specificity of training data	Task-specific *vs.* general data
	Level of transcription	Word, character or phone-level transcription?
	Detail of transcription	Transcribe hesitations, restarts, noises, partial words? Include time stamps? Or just make rough word-level effort?
	Control of transcription	Double check transcriptions?
Language Models	Type of language model	Stochastic *vs.* rule-based
	N-gram distance	2, 3, 4, or variable
	N-gram smoothing	Katz, Kneser/Ney, or other
	Amount of training data	Domain with much data available? Some applications, such as broadcast transcription, offer almost unlimited on-line sources for language model training data. Other applications do not have such an advantage.

Search Organisation and Control	Search algorithm	Viterbi (or "alpha") decoding *vs.* A* (or some kind of breadth-first) strategy
	Multiple passes	Use more than Viterbi?
	Combination of knowledge sources	Use both acoustic and language model information in all passes?
	Phonological rules	Allow phonological rules to adjust word-level pronunciations in context in paths generated during search.
	Pruning	Allow tuning of pruning thresholds, either empirically during development or automatically as a function of available CPU and memory.
	Confidence Annotation	Produce either rejection/OOV or correct/incorrect scores at the word or utterance level.

1.2 Draft Best Practice Grid for Speech Generation

Issue	**Option**
Device(s) Output	Describe output device(s). Is multimodal output possible? Which API is/are needed? Is barge-in allowed? Is there any other way to stop the speech output? Is multimedia output possible?
Input	What is the form of the input? Is any standard input format or mark-up language used? Are any additional notations, for example prosody allowed?
Platform	System requirements? How much memory is needed? Is any special hardware needed?
Language(s)	Is the system monolingual or multilingual ? Which language(s) does the system speak? Is it easy to switch between languages?
Lexicon	Possible entries: What are the types of words found in the lexicon (e.g. lemmas or inflected forms)? How are they transcribed? Are for example abbreviations or idiomatic expressions included in the lexicon? Size: How big is the lexicon? Tagging: Are entries tagged in any way? Are there any special lexicons: name pronunciation, date

	pronunciation etc. Can the user add an extra lexicon? Can the speech generation share a lexicon with another module?
Sound generation technique	Characterise generation approach: canned and/or concatenation, analysis-synthesis, synthesis by rule. Synthesis technique: formant synthesis or waveform (concatenative) synthesis? If waveform synthesis, is PSOLA, MBROLA or some other technique used? If formant synthesis, is articulatory models used? If canned speech or concatenative synthesis: What size are the basic units, words, morphemes, etc.? How many units? Are all units of the same length? How are the units chosen and combined? How much recorded data was needed? Synthesis method: rule-based versus lexicon look-ups? Are there fall-backs, for example rules for words not in lexicon? Disambiguation of homographs? Are prosody rules included? What prosodic control is used (prosodic phrasing, stress, accent)? Is it possible to add a new voice chosen by the user?
Flexibility	Are different voice characters (e.g. male/female) allowed for? Different speaking styles? Different speaking rates? Emotion? Can the customer change the voice character or quality? Can changes be performed during use?
Evaluation	Has any evaluation of the speech been performed? What has been evaluated if so and how has the evaluation been performed? Describe the test that have been used and the results.

1.3 Draft Best Practice Grid for Natural Language Understanding and Generation

Not available at the time of writing.

1.4 Draft Best Practice Grid for Dialogue Management

Overall issue	**Issue**	**Option**
Goal of the Dialogue Manager	Efficient task performance	

System Varieties	Multimodal systems including speech	Connecting to a human being and other output actions Using other modalities than speech
	Multilingual systems	Translating from one language into another Offering the user to choose among several languages
	One or several tasks and users?	
Input Speech and Language	Are the speech and language layers OK?	
	Do the speech and language layers need support from the dialogue manager?	Input prediction support Input language processing support Output generation support
	Real-time requirements	
Getting the User's Meaning	Task complexity	
	Sub-issue: Volume of information	
	Sub-issue: Ill-structured tasks	Using domain structure Electronic yellow pages
	Sub-issue: Well-structured tasks	
	Sub-issue: Negotiation tasks	
	Controlling user input	Information on system capabilities Instructions on how to address the system Feedback on what the system understood Processing feedback Output control Focused output and system initiative Textual material Barge-in
	Who should have the initiative?	System-directed dialogue Mixed initiative dialogue User directed dialogue
	Input prediction/prior focus	Knowledge-based input prediction Statistical input prediction

	Sub-task identification	Local focus Global focus Identify one sub-task only Identify each single sub-task
	Advanced linguistic processing	Co-reference and ellipsis processing Discontinuous user input Processing of indirect dialogue acts
Communication	Domain communication	More information needed Database look-up Producing an answer Making an inference More constraints needed Inconsistent input Language translation
	Meta-communication	System-initiated repair meta-communication System-initiated clarification meta-communication User-initiated repair meta-communication User-initiated clarification meta-communication
	Other forms of communication	Handling out-of-domain terms
	Expression of meaning	Pre-recorded utterances Concatenation of pre-recorded words and phrases Filling in a template used by a synthesiser Producing meaning
	Error loops and graceful degradation	Focused questions Asking for re-phrasing Asking for a complete sentence Yes/no questions Spelling
	Feedback	Information feedback Process feedback
	Closing the dialogue	Closing (only) when finished Closing when failing

History, Users, Implementation	Histories	Task history Topic history Linguistic history Performance history
	Novice and expert users, user groups	Domain and system experts System novices Domain and system novices Other user groups
	Other relevant user properties	Standard goals User beliefs User preferences Cognitive loads Response packages
	Implementation issues	
	Sub-issue: Architecture and modularity	
	Sub-issue: Main task of the dialogue manager	
	Sub-issue: Order of output to the user	
	Sub-issue: Task and domain independence	

1.5 Draft Best Practice Grid for Human Factors

Overall issue	**Issue**	**Option**
Error detection and correction addresses only a small fraction of the problem		Several strategies exist
Dialogue initiative		System directed User directed Mixed initiative
	Interaction mode	Transactional mode Interactional mode

	Menu design	Skip and scan A linear main menu or help prompt listing all of the functions (ordered sensibly) A menu-hierarchy Pseudo sub-menus (PSMs) Form filling
	System output design	
	Type of system output	Complete speech recordings Concatenated speech recordings Synthesised speech
	Cues for turn-taking	
	Help design	Implicit part of dialogue Alternative dialogue on request Automatically enabled
	Dealing with naive, novice and expert users	
Dealing with user expectations	User experience	
	User adaptation to system output	
	User's understanding of the system functionality (Introductions)	Mailouts Quick reference cards
	User's understanding of the structure of the interaction	
User modelling and user-dependent dialogues	Anonymity vs. identity	
	Sub-issue: Keeping a user profile	Interactional information Asking users their preferences Tracking user interaction
	Uses of user-specific information	Prompt adaptation Dialogue flow adaptation More complex recognition Suggestive dialogues
Addressing users	Asking questions	
	Sub-issue: Navigational questions and task selection questions	All task choices via closed questions Closed questions when the user knows the system functionality

	Sub-issue: Task detail questions	Instructions as part of question
	Sub-issue: Confirmation questions	Repairing collected information which is wrong
	Answering questions	
	Sub-issue: Clear and unambiguous answers	Use language the user is familiar with Provide sufficient information Do not provide too much information Be correct and relevant Be short
	Providing information	
	Sub-issue: Uniformity in information provision	Same order Same vocabulary for similar types of information
	Sub-issue: Playing out number strings	Follow normal conventions
	International dialogue interfaces	
	Politeness	Expert users do not need polite interfaces

1.6 Draft Best Practice Grid for Systems Integration

Overall issue	Issue	Option
Logical (Functional) Integration	Functional Modules	Speech Recogniser Syntactic Parser Semantic Parser Dialogue Manager Output Generator
	Application Resources	Acoustic Models Grammar and Lexicon Dialogue Data Speech Output Resources User Models
	Module and Resources Communication	Strategies APIs, Platforms and Protocols

Architectural (Environmental) Integration	External links	Telephony Interface LAN/WAN Databases Peripheral Devices Other Applications
	External Communication Protocols	Telephony Protocols Network Protocols Database Protocols
Service and Maintenance	Monitoring	Collecting Real-Life Data
	Changing Systems	Portability Extensibility Backward Compatibility

Appendix 2: Draft Best Practice Life-cycle

Overall design goal(s): *What is the general purpose(s) of the design process?*

E.g. to build a product, a research prototype, to achieve excellence in a certain area of research, to explore a particular approach, other; to particularly study certain aspects of the system or component. Comment, if needed, on whether the design goal is worthwhile, e.g. from the point of view of innovative research.

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

Constraints could be economical, derived from performance demands on the system (e.g. real-time), other. Describe the effects, if any, of the constraints (e.g. on vocabulary size, recognition quality).

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

E.g. use of in-house or off-the-shelf speech recognisers, synthesisers, other. Describe the effects, if any, of the constraints.

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

E.g. hardware constraints, adequacy evaluation criteria, other. Note that research prototypes may be built to hypothetical customer constraints. The basic advantage of assuming hypothetical customers is that the developers force themselves to face realistic problems and hence to be accountable for any deviations from a realistic development life-cycle. Such deviations may be justifiable from many different points of view but they are not likely to be recognised as such unless the project has (simulated) realistic boundary conditions. Describe the effects, if any, of the constraints.

Other constraints: *Were there any other constraints on the design process?*

E.g. on cost, person power, purchase price, development time, development phases, standards conformation, knowledge in the developer team. Describe the effects, if any, of the constraints.

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

E.g. innovative product features, innovative experimental features, other? Describe the effects, if any, of the ideas.

Designer preferences: *Did the designers impose any constraints on the design which were not dictated from elsewhere?*

E.g. programming language preferences, development methodology. Describe the effects, if any, of the preferences.

Design process type: *What is the nature of the design process?*

E.g. exploratory research, product development, redesign, other.

Development process type: *How was the system/component developed?*

E.g. through Wizard of Oz, based on human-human or human-computer dialogues, using development methodology X (describe it) using no particular methodology except "to see if things work", other.

Requirements and design specification documentation: *Is one or both of these specifications documented?*

Describe the specifications. In the absence of requirement and design specifications, the developers have no guidance wrt. when or to what extent they will have achieved their development objectives.

Development process representation: *Has the development process itself been explicitly represented in some way? How?*

E.g. bits and pieces can be found in scientific papers, the entire process was carefully documented in semi-formal notation, most of the process has been systematically represented in reports or meeting protocols, other. The advantages of explicit development process representations are that these can be re-used, possibly in revised form, in new projects and with new developers coming on the team, and can support re-design and maintenance. The main disadvantage is that this represents an additional project cost.

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

Most interactive speech systems have something to do with real user needs. However, to appropriately address real user needs, the development process often needs to include extended end-user contact, extensive work on domain delimitation, clear up-front performance criteria, final adequacy criteria, extended quantitative and qualitative evaluation throughout the development process, an explicit development methodology etc.

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

E.g. "allow users to do tasks X and Y", "include barge-in", "respond in real-time". Note that this entry is more general than, but may partially overlap with, the "grid" properties.

Usability criteria: *What are the aims in terms of usability?*

E.g. spontaneous unconstrained dialogue, usable with no training, usable with training in .. (explain), naturalness, high user acceptance, intuitively well-defined task domain, other.

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

E.g. partially replace the switchboard operator, require backup for difficult or incomprehensible queries.

Customer(s): *Who is the customer for the system/component (if any)?*

E.g. the system/component is custom-built, addresses a specific market segment, has not customers but produces spin-off products, has "simulated" customers, other.

Users: *Who are the intended users of the system/component?*

E.g. users speaking High German, or Swedish, walk-up-and-use users, specialised user group X. Is walk-up-and-use an appropriate paradigm for the application?

Developers: *How many people took significant part in the development? Did that cause any significant problems, such as time delays, loss of information, other (explain)? Characterise*

each person who took part in terms of novice/intermediate/expert wrt. developing the system/component in question and in terms of relevant background (e.g., novice phonetician, skilled human factors specialist, intermediate electrical engineer).

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

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

However difficult this may be to do in any formal way, it is essential to good development practice to make explicit and systematically evaluate the requirements or design specifications.

Evaluation criteria: *Which quantitative and qualitative performance measures should the system/component satisfy?*

The definition, from early on in the development process, of clear, relevant and appropriate evaluation criteria, and the continuous evaluation of progress using those criteria, are main characteristics of best practice in development and evaluation of spoken language dialogue systems.

Appendix 3 contains a list of best practice evaluation criteria per aspect. For the relevant criteria, state their definitions, describe the performance targets and state whether these were achieved.

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

Describe, one-by-one, the aspects that were evaluated, when, the set-up and the methodologies used, e.g. Wizard of Oz scenario-based, glassbox, blackbox, progress (comparing successive measurements), diagnostic, performance, adequacy, acceptance, field, objective, subjective. Number of subjects/users involved in each test.

How was data collection done (logfiles, corpora, questionnaires, interviews, other)? Describe the corpora, etc. Was data annotation done? How? Which information has been extracted from the data?

What were the results? Is test material/data/test suites (and/or a description of the test conditions) available? Can the test be replicated? Can anybody perform the tests?

Is anything stated about comparability of the test(result)s with those of other systems/components of similar scope?

Mastery of the development and evaluation process: *Of which parts of the process did the team have sufficient mastery in advance? Of which parts didn't it have such mastery?*

Note that lack of mastery of parts of the development process is a normal condition in research projects which often serve in part as competence-building exercises.

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

E.g. problems of collaboration in the team, major delays caused by ?, difficulties in satisfying specification requirement X, developer Y left the team, lack of quality of what was delivered by some in the team.

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

Component selection/design: *Describe the system components and their origins.*

E.g. off-the-shelf, based on somebody else's parser (specify), built in-house for the application, other (specify).

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

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

Note that maintenance may include continued development and re-design. Are there guidelines for maintenance (of, e.g., lexicon and grammar)?

Portability: *How easily can the system/component be ported?*

E.g. OS dependencies, machine dependencies.

Modifications: *What is required if the system is to be modified?*

Additions, customisation: *Has a customisation of the system been attempted/carried out (e.g. modification of a part of the vocabulary, new domain/task, etc.)? Has there been an attempt to add another language? How easy is it (how much time/effort) to adapt/customise the system to a new task? Is there a strategy for resource updates (e.g. a predefined sequence of update steps to be performed if a new item is added to the lexicon or if a new grammatical description is added to the grammar)? Is there a tool to enforce that the optimal sequence of update steps is followed (e.g. a menu-driven update interface, etc.)?*

Property rights: *Describe the property rights situation for the system/component.*

Documentation of the design process:

E.g. specification documents or parts thereof, architecture diagram (mandatory), user scenario(s), transcribed dialogue(s), other.

References to additional project/system/component documentation:

Please refer to relevant information.

Appendix 3: Draft Best Practice Evaluation Criteria

This appendix lists the draft best practice evaluation criteria proposed for each aspect (Sections 3.1 to 3.6). A template (Section 3.7) has been proposed for a common description structure of each individual evaluation criterion. For some aspects the criteria have already been described using the template. For other aspects this description will be made as part of the DISC-2 work.

3.1 Criteria for Speech Recognition Evaluation

1. Word Error Rate
2. Processing Time
3. Lexical Coverage
4. System Complexity
5. System Size
6. Modifiability
7. Complementary Transcription Annotations

3.2 Criteria for Speech Generation Evaluation

1. Transcription

Translation from text to phonemes, pronunciation rules

2. System

Real-time

Lexicon content

Special lexicon

Markup codes

3. Speech

General impressions

- Overall Impression
- Overall Quality
- Naturalness
- Pleasantness
- Acceptability

Intelligibility

- Intelligibility of Phonemes, Nonsense Words and Words of One or More Syllables

Intelligibility of Names

Intelligibility of Sentences

Comprehension of Content

Listening Effort

Noise Sensitivity

Intelligibility Over Telephone Networks

Quality

Pronunciation

Articulation – Segmental Quality

Prosody

Speaking Rate

Voice Quality

Distortion, Clicks or Other Extraneous Sounds

Continuity

4. Multimodality

Talking head

Graphics

3.3 Criteria for Natural Language Understanding and Generation Evaluation

Not available at the time of writing.

3.4 Criteria for Dialogue Management Evaluation

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).

3.5 Criteria for Human Factors Evaluation

1. System communication about the communication (meta-communication)
2. Dialogue initiative
3. Dialogue structure
4. Modality appropriateness
5. System communication about the task
6. Adaptivity to user differences
7. Task and domain coverage
8. Coverage of user vocabulary and grammar
9. Information about system capabilities, limitations and operations
10. Number of interaction problems
11. User satisfaction

3.6 Criteria for Systems Integration Evaluation

1. Global task performance
2. Portability
3. Maintainability
4. Extensibility
5. Robustness

3.7 A Template for Evaluating Aspects of SLDSs

The evaluation template includes ten entries A - J. 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 ten 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. System part evaluated

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

C. 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.

Quantitative and qualitative evaluation are both *objective evaluation*.

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

Comparative evaluation consists in comparing quantitative, qualitative or subjective evaluations for different SLDSs and components. Comparative evaluation is often done internally in a development process in order to measure progress (progress evaluation). In most cases, this does not produce independently meaningful scores which can be used in comparisons with other SLDSs or components. 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.

D. 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.

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

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

Post-trial interviews are useful for capturing user observations which might otherwise have been missed and which might have implications for virtually any kind of system deficiency.

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.

E. Symptoms to look for

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

F. 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. 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.

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

G. 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.

H. Difficulty of evaluation

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

- the difficulty of evaluation may depend on various forms of *complexity,* such as task complexity, user input complexity, dialogue manager complexity, or overall system complexity;

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

I. Cost of evaluation

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

- evaluation is more or less *costly* to perform in terms of time, manpower, or skilled labour;

- difficult evaluation may be relatively uncostly, for instance if done by a consultant; easy evaluation can be costly, for instance because of the volume of data involved;

- Wizard of Oz simulations, field studies and their associated data analysis are costly.

J. Tools

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