Dialogue Development and Implementation in the Danish Dialogue Project

Hans Dybkjær, Niels Ole Bernsen and Laila Dybkjær
Centre for Cognitive Science, Roskilde University
PO Box 260, DK-4000 Roskilde, Denmark

Abstract

This chapter presents results on dialogue development and implementation of the first prototype P1 in the Danish Dialogue project. The project as a whole is briefly presented in terms of system components and system architecture. The remainder of the chapter focuses on dialogue. Firstly, it is described how a dialogue model for the first prototype P1 was developed using Wizard of Oz (WOZ) experiments. The WOZ method is described and results from the WOZ experiments presented. Secondly, a description of the implementation of the dialogue model is provided. The conclusion presents a number of open questions to be answered during the test of the prototype.

1. Introduction

The Dialogue project is a Danish national project on spoken language dialogue systems. The project started in 1991 and is carried out with an effort of 30 man/years by the Center for PersonKommunikation (CPK, earlier the Speech Technology Centre - STC), Aalborg University, the Centre for Language Technology (CST), Copenhagen University, and the Centre for Cognitive Science (CCS), Roskilde University. The aim is to develop two application-oriented dialogue system prototypes called P1 and P2 in the domain of Danish domestic airline ticket reservation and flight information accessed through the telephone. The first prototype, P1, has been built and is currently being tested. The next step will be to develop P2 as a more advanced version of P1 based on the test results on P1.

The plan of the chapter is as follows. Section 2 briefly explains the main system components and the system architecture. Section 3 describes the dialogue design process including initial design specification, methodology and

dialogue modelling results. Section 4 describes the implementation of the dialogue model. Section 5 concludes and discusses future work.

2. System Components and System Architecture

To provide the context for dialogue development and implementation an outline of the P1 prototype system is given in this section. P1 is outlined both in terms of logical system structure and physical system structure.

2.1 System Components

The logical system structure is presented in figure 1 which shows the main components of P1.

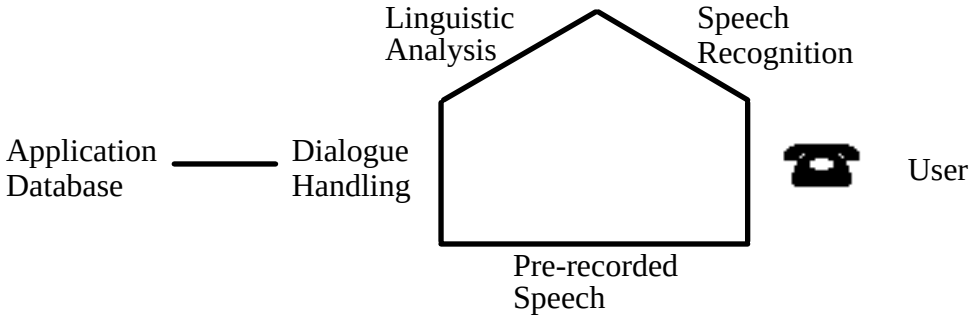

Figure 1: The main components of the P1 spoken language dialogue prototype system.

A user calls the system and provides input to the *Speech Recognition module* which processes the speech signal. The speech recogniser is a further developed version of the recogniser which was developed in the Esprit SUNSTAR project [13]. It is a speaker-independent continuous speech recogniser based on Hidden Markov Models (HMMs). In addition to user input, the speech recogniser needs predictions from the Dialogue Handling module on the sub-grammars to be used at any given point during the dialogue. The sub-grammars used in the Speech Recognition module are word pair grammars represented as finite state transition networks in which the transitions represent HMMs. Viterbi search is used to find a 1-best path through the network. This path represents a string of lexical references which constitutes the output of the Speech Recognition module.

The lexical string is input to the *Linguistic Analysis module*. The Dialogue Handling module indicates to the parser which sub-grammars to use and which semantic objects to fill in on the basis of the input string from the recogniser. The semantic objects are frame-like structures containing a number of slots for domain relevant information. The sub-grammars used for linguistic analysis are unification-based Augmented Phrase Structure Grammars (APSGs)

implemented in a formalism which is a subset of the one used in the Eurotra project [6]. The Linguistic Analysis module analyses the input based on the active sub-grammars using a chart data structure and an object-oriented implementation of the Earley parsing algorithm. The parser uses semantic mapping rules for assigning semantic interpretations [14] which in turn are used for filling in the active semantic objects.

The *Dialogue Handling module* interprets the contents of the semantic objects received from the Linguistic Analysis module and decides on the next action to take which may be to send a query to the *Database* or send relevant output to the user. In the latter case, the Dialogue Handling module also sends predictions to the speech recogniser and the parser on the next sub-grammars to use, i.e. on which input now to expect from the user. The Dialogue Handling module, in particular the dialogue description, is discussed in detail in section 4 below.

The output module is based on *Pre-recorded Speech*. A number of words and (parts of) sentences have been recorded in advance and are selected, put together and replayed according to instructions from the Dialogue Handling module.

2.2 System Architecture

The system architecture of P1 [11] is based on the SUNSTAR DDL/ICM architecture [5] developed in the Esprit SUNSTAR project. Figure 2 presents the physical architecture of P1. The *Dialogue Communication Manager* is a bus carrying messages between the other components. These may be other programs or hardware and communicate with the bus through drivers.

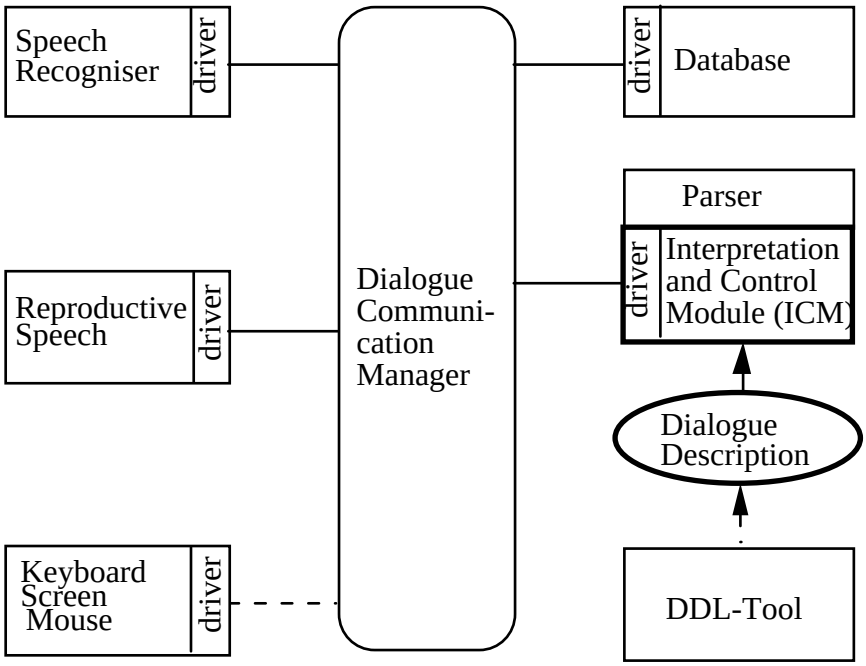

Figure 2: Overall system architecture of P1.

The core module is the *Interpretation and Control Module* (ICM). ICM interprets a *Dialogue Description* which is a program written in DDL (Dialogue Description Language). DDL is an experimental language originally intended for primitive dialogues not involving natural language. DDL has been extended in the Dialogue Project to meet the particular needs of the P1 system. DDL has three layers: a graphical layer which specifies how the dialogue will be controlled in terms of event-driven recursive flow charts; a frame layer which declares data structures; and a textual layer which declares data structures and specifies actions. The DDL Dialogue Description has been created by using the *DDL-Tool* which is a graphical editor and debugger. The Dialogue Description and the ICM jointly form the Dialogue Handling module. The *Parser* has been implemented as a module external to the ICM.

When input from the *Speech Recogniser* is expected by the Dialogue Description, the ICM looks if there is a message. The message is passed through the Parser before the ICM continues its interpretation of the Dialogue Description. Queries to and answers from the *Database* are also exchanged as messages. Output information is sent as a message to the *Reproductive Speech module* and predictions are sent as messages from the ICM to the Speech Recogniser.

The *Keyboard, Screen, and Mouse modules* are not part of the running P1 system but support testing of the system. For instance, the speech recogniser may be simulated via keyboard input. The presence of the Keyboard, Screen and Mouse modules also enables later extension of the system into a multimodal system.

The P1 system's Speech Recogniser runs partly on a Digital Signal Processor (DSP) board and partly on a PC whereas the rest of the system runs on a Sun/Sparc station.

3. Dialogue Model Development

The goal of dialogue model development in the case of the spoken language dialogue system P1 was to enable the machine to conduct a dialogue with users which allowed them to solve their tasks in a way which was as natural as possible given the heavy technological and other constraints on the design process, many of which were imposed by the speech recogniser. This section describes, firstly, the initial design phase where knowledge is elicited for a first dialogue model and other design decisions are made which influence the rest of the design process. Secondly, the Wizard of Oz (WOZ) prototyping method used for iterative dialogue model design is described followed by a review of the main results obtained in attempting to meet the design process constraints.

3.1 The Initial Design Phase

A number of different information sources contributed to the design of the first dialogue model for P1. The research literature provided an update on the state of the art in spoken dialogue systems [12]. Field interviews provided information on the tasks done by human travel agents and how to define the domain of P1. Details on departures, fares, travel conditions, etc. were obtained from standard timetables. Due to practical difficulties, recordings of human-human dialogues in the selected domain of application were made too late to be used in defining the first dialogue model. The main issue which was identified was a set of conflicting constraints which had to be traded off against one another in order to build a usable and technologically feasible system.

On the system side, the dialogue model for P1 had to satisfy the following technological constraints which were mainly imposed by the speech recogniser:

- an average user utterance length of 3-4 words;
- a maximum user utterance length of 10 words;
- at most 100 words can be active in memory at a time for real time performance to be possible. Real time performance has high priority in usable systems in the chosen domain of application;
- project resources limit the vocabulary to about 500 words.

On the user side, the aim is to allow use of natural forms of dialogue and language. This will contribute to making the system easy to use by both novices and experts but obviously conflicts with the technological constraints just mentioned. Naturalness therefore has to be traded for system feasibility as naturalness is the only aspect of usability which reasonably may be thus traded. Other aspects of usability must be satisfied for the system to be at all usable. Basic system usability requires close-to-real-time performance, sufficient domain and task coverage, sufficiency of task-related vocabulary, natural grammar, robust handling of error, and that limitations on the naturalness of dialogue and language be principled and practicable by users [1]. So the trade-off process is further limited by these basic usability constraints.

No current theory is able to resolve this conflict. The best approach is to use an experimental and iterative design technique, such as WOZ.

3.2 The Wizard of Oz Method

WOZ [10] is a powerful empirical technique which is well suited to the iterative development and evaluation of intelligent interactive systems whether these be uni-modal, as in the current case where speech is being used for both input and output, or multi-modal. WOZ makes possible the testing of design ideas and the acquisition of detailed knowledge of the system, its users and user/system interaction prior to system implementation. Design goals and constraints may be simulated and adjusted until an acceptable trade-off has eventually been found.

WOZ involves one or more ‘wizards’, i.e. humans who simulate the performance of non-implemented or partially implemented computer systems in front of users who are preferably ignorant of the fact that they are interacting with a simulated system rather than a real one. Interactions are logged and recorded in various ways, often transcribed and indexed, and analysed for a variety of purposes. WOZ differs from other prototyping techniques, firstly in that it does not rely on reductions of the artifact and/or the task domain into presumed ‘essential’ or ‘representative’ features whose identification remains problematic. This means that, ideally, the end result of the WOZ specify-and-simulate test cycle will be a simulated system which can be implemented more or less directly on the assumption that the cycle has helped the designers to identify nearly all potential problems with the future system. Secondly, the presence of a human wizard allows simulation of a broad class of cognitively demanding tasks which humans are naturally good at, such as natural language understanding and generation, gesture recognition or visual scene understanding.

In developing a dialogue model for P1 seven generations of WOZ experiments were performed [9]. The simulation set-up is shown in figure 3.

The *graph structure* used by the wizard describes the dialogue structure including who has the initiative while the *predefined phrases* show the language to be used by the system. The graph structure and the phrases jointly constitute the dialogue model and are the crucial variables involved in finding an appropriate trade-off between technological constraints and naturalness. A *timetable* and a *calendar* acted as database.

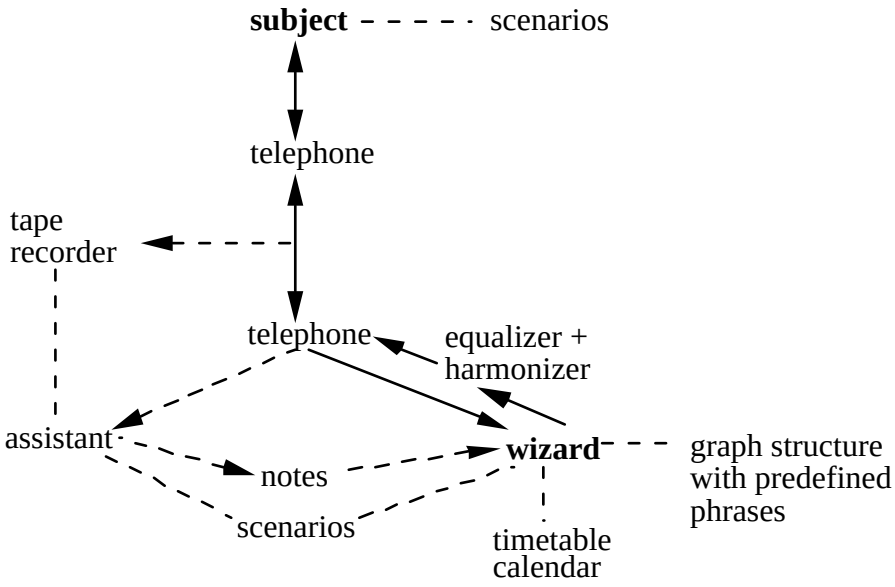

Figure 3: The set-up of the WOZ experiments.

In the later generations an *assistant* helped offload the wizard. The assistant operated the *tape recorder*, took *notes* on the information provided by subjects and gave other practical support.

To induce subjects into believing that they were speaking to a computer, an *equalizer* and a *harmonizer* were used to distort the wizard's voice during the last set of experiments [7].

The first five generations served training of the wizard and adjusting major shortcomings in the dialogue model. *Subjects* were exclusively system designers and colleagues. In each of the two last generations 12 subjects were used. The majority were external subjects and the rest were colleagues. External subjects were selected so that half of them had a background as secretaries (the expected end-user group) and the other half were computer scientists. The results obtained confirm that subjects' professional backgrounds influence the way they interact with the system [7]. Each subject received a letter which briefly introduced the system and informed on the subject's role. The letter also contained four *scenarios*, i.e. domain-relevant tasks which the subject was asked to perform, as well as a questionnaire to be filled in and returned after the experiments.

3.3 WOZ Results on the Dialogue Model for P1

Each WOZ generation produces large amounts of quantitative data which are used for measuring the extent to which quantitatively stated constraints are being met. This section describes the dialogue model development process focusing on the extent to which the mentioned (section 3.1) technological constraints were satisfied. A second, equally important, use of (quantitative, qualitative or structural) WOZ data during design is the use of data as evidence of user problems with the simulated system. The user problem types which were identified and addressed during P1 dialogue design have been described elsewhere [2,4].

Initially the dialogue structure was a loosely ordered set of predefined phrases. There were no constraints on which phrases could be used in which circumstances. The choice was fully left to the wizard who had great problems being consistent as a result. Subjects had as much of the dialogue initiative as they wanted to but the technological constraints were not met. A more powerful tool was needed to obtain a consistent and incremental dialogue model which might eventually satisfy the technological constraints. A graph structure having predefined phrases in the nodes and predicted contents of user input along the edges was chosen for this purpose. The graph represented a more structured dialogue in which it was well-defined which ordered pieces of information the system needed from the user in order to make, e.g., a reservation. Domain coverage was adjusted to make its limits increasingly well-defined and the coverage itself more complete.

As P1 requires limited user utterance length, at most 100 active words at a time and limited vocabulary, user dialogue initiative causes problems because of

the length and unpredictability of users' utterances. To satisfy those constraints, the dialogue had to be made increasingly system-directed. This was done by converting user questions into system questions. Asking the questions allows the system to have well-defined expectations concerning user utterances (answers) in context.

As can be seen from figures 4 and 5, users' average utterance length and the average number of utterances exceeding ten tokens (words) decrease while more and more of the dialogue initiative is left to the system which asks nearly all the questions in the 7th generation (figure 6). Two other factors instrumental in reducing user utterance length were: (a) an introductory admonition to users to be brief when answering questions posed by the system, and (b) the fact that the system addressed users tersely rather than politely [15].

Interestingly, system-directed dialogue seems quite acceptable and natural in some tasks. Recordings of dialogues from a travel agency showed that once the customer has expressed a goal and a few constraints, the travel agent typically takes over and asks questions. This is particularly clear in the case of reservation tasks whereas customers typically ask more questions when performing information tasks. The difference between reservation and information tasks is that reservation tasks require the exchange, in some sequential order, of well-defined sets of information whereas information tasks have no such structure.

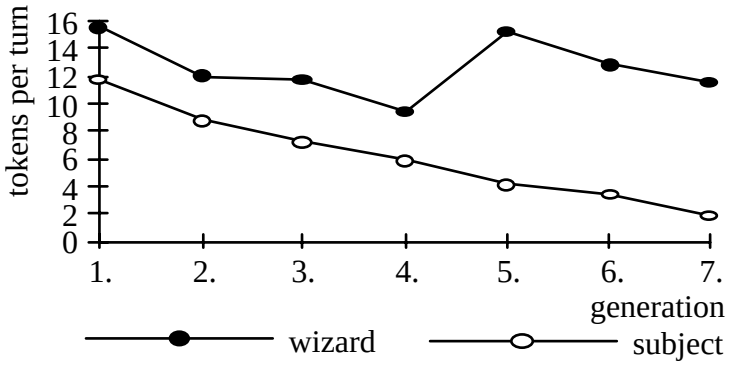

Figure 4: Average length of wizard and subject utterances in terms of tokens per turn. In the 5th generation, more information was included in the wizard's utterances, sparing users from having to ask for it.

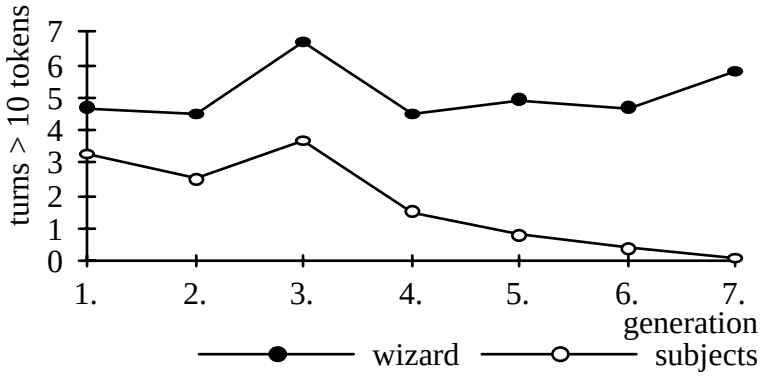

Figure 5: Average number of turns per dialogue exceeding 10 tokens.

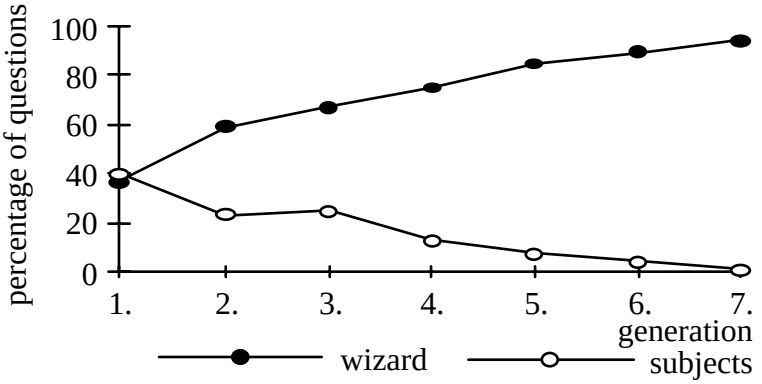

Figure 6: Number of questions in per cent of total number of turns.

Field recordings also showed that the average number of words per system/user exchange as well as per task were largely at the same level in the 7th WOZ generation as in similar human-human dialogues. This may be taken to indicate that a natural level of information exchange had been reached.

A sub-language vocabulary of 500 words has been defined on the basis of the 6th and 7th generations of WOZ experiments. However, it is not clear whether 500 words are sufficient for enabling recognition of the vocabulary that is natural to users in the task domain. The WOZ vocabularies did not clearly converge, as indicated by the vocabulary from the 7th generation in figure 7. Note that the figure only represents types other than numbers, days of the week, months and destinations. Numbers, etc. are irrelevant to the issue of convergence as a complete set of them has to be represented in the system anyway.

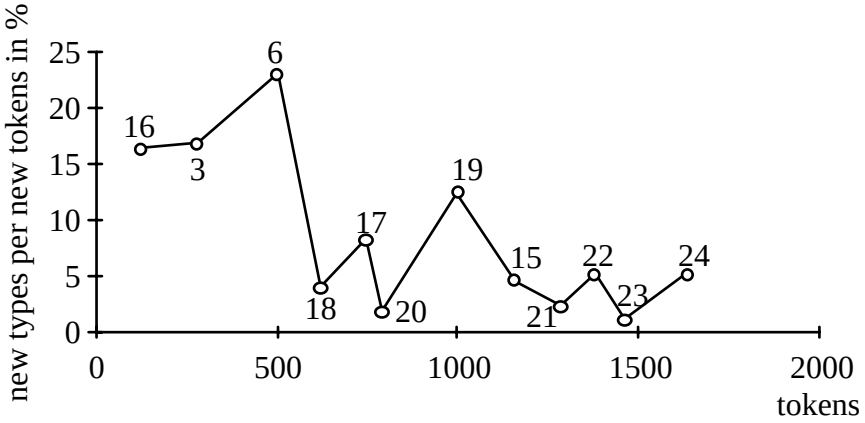

Figure 7: Cumulative type/token ratio for the subjects in the seventh generation. The types counted do not include numbers, week-days, months and destinations. Subjects' numbers are indicated in the data points.

4. Dialogue Implementation

The dialogue model developed during the WOZ experiments has been implemented as a dialogue description in DDL [8]. This section provides an outline of the dialogue program structure.

The implementation of the dialogue description has the following two main aspects:

1. Domain. The system is task oriented. Each task comprises a number of pieces of information each of which must be established and checked for bindings. For example, the task of determining a travel route requires in P1 two pieces of information, namely the departure airport and the arrival airport. For two such airports to define a route they must exist in the timetable.

2. Dialogue. This primarily concerns how the order of establishing information (i.e., the order of individual tasks) is defined, who has the initiative, and the built-in facilities for supporting task-independent dialogue with the user such as the user commands *Repeat* and *Correct*.

The main flow of the implemented dialogue description is expressed by the DDL procedures (graphical level) shown in the figures below.

a

b

Figure 8: The graphical DDL representation of *Graph->?* (a) *Graph->new* (b).

Graph->? (figure 8a) determines which node to proceed to, i.e. which piece of information to establish, and *Info-node* (figure 9) finds a value for that node. In broad terms, the entire dialogue is carried out by repeatedly performing these two main actions.

All information collected from the user as well as from the database during the dialogue is represented in the *dialogue state*. The dialogue state is defined by the following four elements to be explained below:

- a number of task objects each of which has slots indicating
  - if the node is already being checked;
  - the node status as regards user and system;
  - a value;

• the current item (piece of information);
• the previous item; and
• the current graph;

The first three elements are established in *Info-node* and the last one in *Graph->?*. When a user calls the system the dialogue state contains an empty task object, and the current and previous items are set to *ZERO* (cf. figure 8.b) which is the root node.

The previous item acts as a degenerate dialogue history. A real dialogue history would contain information on the dialogue from its beginning to the present state, but in P1 only the previous item is being stored at any given time. The current graph is initially set to *Graph->new* (figure 8.b). After initialisation, the program repeatedly performs the two main actions: shift to a new node and find a value for it.

Graph->? checks which graph is the current one. For instance, *Graph->new* (figure 8.b) is the current graph in the initial part of any dialogue, and *Graph->reserve* becomes the current graph if and when the user decides to make a reservation and until the reservation has actually been made. On the basis of the current graph, *Graph->?* determines the next node to proceed to and calls *Info-node* instantiated to this node.

Info-node controls the acquisition of information. It first checks if the current item is already being checked and, if so, control is returned to *Graph->?*. The purpose of this check is to avoid circularity in the dependency graph not shown in figure 9 but underlying *Info premises* and *Info depends*. If, e.g., the current item is the arrival airport and the previous item is the departure airport then the information on the departure airport has caused the system to ask for the arrival airport in order to check whether the route is a valid one. However, in this case the information on the arrival airport should not cause the system to ask once more for the departure airport.

If no circularity exists, *Info-node* checks the premises of the node and then its status is considered. According to its status an action is performed. The node's user status expresses the dialogue description's record of the user and what information the user has provided or been given as regards the current item. The node's system status expresses the dialogue description's record of what the system has done as regards the current item. User status and system status may each take one of the following values:

bottom	the node has no value yet.
no	the value is marked as incorrect.
check	the value must be checked.
partial	the value is only partially determined.
yes	the value is determined and accepted.

Figure 9: The graphical DDL representation of *Info-node*.

In addition, the user status may take the value:

inform	tell the user the value.

The (user, system) status pair determines the action to be taken by the system in its next turn. Turn actions may be

NOA	no action.
NEW	ask the user for a new value.
CUS	check the value with the user.

PUS	ask the user, given a partially determined value.
ERR	tell the user that there is an error.
IUS	inform the user about a value.
CON	check validity and consistency of a value

Suppose that the next information to be established is the hour of departure. When *Info-node* is called, user status (U) as well as system status (S) have the value *bottom*. In *Info premises* it is checked if the premises for hour (route and date) have both U and S set to *yes*. If so, *Info?* is entered because of the two *bottom* values for hour. The user is asked to indicate the hour of departure. *Info get* awaits the user answer which could be: “In the morning.” In this case U is set to *partial* and S to *check*. This status leads on to *Info consistent* which checks the database and sets S to *partial* (U remains unchanged). In *Info!?* the user is told which departures are possible in the morning in question and asked if s/he wants one of them. *Info get* awaits the user answer which could be: "Seven thirty." In this case U is set to *inform* and S is set to *check*. Since system status has higher priority than user status checking will be done before informing the user. In *Info consistent* S is set to *yes* because the departure existed in the database. Then *Info!* is entered and the system informs the user that 7:30 is the value it has accepted and U is then also set to *yes*. Now a value is obtained which is supposed to be accepted by user as well as system and it is checked whether the value of the node has been changed. This is not the case since both U and S were set to *bottom* from the start which means that there was no value. Control is now returned to *Graph->?*.

In cases where a value has been changed, e.g. if the day of departure has been changed from Monday to Tuesday, it is also checked if the values of all the nodes depending on the current one are still correct.

In addition to the described main flow of the program there are four exceptions: *Repeat, Correct, Not understood* and two versions of *Timeout*.

When the user says *Repeat*, the program will just return to the choice of action in *Info-node* and then execute it again.

A *Correct* event from the user will cause the current item to be set to the previous item and then *Info-node* is executed again.

A *Not understood* event occurs when the system did not understand what the user said. This fact is communicated to the user who is supposed to answer the previous question again.

Timeout events may be prompting or non-prompting. A prompting *Timeout* event occurs when the user does not say anything during a given time interval. Then the user is asked again. If the user has not responded after a certain number of prompting *Timeouts*, a non-prompting *Timeout* will occur and the system will hang up.

5. Conclusion

Despite the unparalleled power of the WOZ prototyping method, WOZ does have a number of theoretical and practical limitations preventing it from

delivering a system specification which is guaranteed to satisfy all design constraints [3]. Ongoing testing of P1 will help answering questions which were not fully resolved during the WOZ experiments. The main points are the following:

- Since no clear convergence of vocabulary could be observed in the WOZ data material it remains uncertain whether the defined and implemented 500 word vocabulary provides sufficient coverage of the task domain;
- sub-grammars and lexica have been defined and implemented for each node in the dialogue in order to cover the WOZ material from the two last generations. However, it remains uncertain whether no more than 100 active words are needed at any time during dialogue;
- user and system misrecognitions and their repair are difficult to simulate to any great quantitative detail with WOZ. In consequence, P1 may prove to be less robust than desirable. Furthermore, it is an open question whether the implemented task-independent support facilities *Correct* and *Repeat* are sufficient to ensure robustness. Additional error-handling mechanisms may be needed which exploit the power of error correction through dialogue with the user;
- the WOZ method does not in itself ensure a dialogue theory which is sufficiently formalised (for implementation) and abstract (for maintenance and portability of the application).

The P1 evaluation results will be used for developing the planned second prototype P2 which is intended to have improved naturalness, flexibility and robustness. The dialogue history and user model which in P1 are rather primitive will be augmented in P2. P2 should accept more complex user input including longer utterances. Experiments with a larger perplexity will be necessary and a new series of WOZ experiments will be performed to develop an improved dialogue model and determine to what extent the present sublanguage should be enlarged. DDL and the DDL-Tool will be extended to meet the more advanced requirements of P2. A computer-supported WOZ environment will be built in which revisions to the simulated dialogue structure are concurrently being implemented in DDL. On the output side, increased dialogue complexity will require the use of speech synthesis instead of pre-recorded speech.

References

1. Bernsen, N.O. “The Structure of the Design Space”. In Byerley, P.F., Barnard, P.J. and May, J., (Eds.): Computers, Communication and Usability. Design Issues, Research and Methods for Integrated Services. Amsterdam, North-Holland, 221-244, 1993a.
2. Bernsen, N.O. “Types of User Problems in Design. A Study of Knowledge Acquisition Using the Wizard of Oz”. Esprit Basic Research project

AMODEUS-2 Working Paper UM/WP 14. Included in Deliverable I.2, June 1993b.

3. Bernsen, N.O., Dybkjær, H. and Dybkjær, L. “Wizard of Oz prototyping: How and when?”. Submitted to CHI'94, 1993.
4. Bernsen, N.O., Dybkjær, L. and Dybkjær H. “Task-Oriented Spoken Human-Computer Dialogue”. Report 6a, Spoken Language Dialogue Systems, CPK Aalborg University, CCS Roskilde University, CST University of Copenhagen. To appear early 1994.
5. Bækgaard, A., Roman, A. and Wetzel, P. “Advanced Dialogue Design - DDL Tool and ICM”. Esprit project 2094 SUNSTAR, Deliverable IV.6-2, August, 1992.
6. Copeland, C., Durand, J., Krauwer, S. and Maegaard, B. (Eds.) “The Eurotra Formal Specifications”. Studies in Machine Translation and Natural Language Processing, vol. 2, 1991.
7. Dybkjær, H., Bernsen, N.O. and Dybkjær, L. “Wizard of Oz and the Trade-Off between Naturalness and Recogniser Constraints”. Proceedings of EUROSPEECH ‘93, Berlin, September, 947-950, 1993.
8. Dybkjær, H. and Dybkjær, L. “Representation and Implementation of Spoken Dialogues”. Report 6b, Spoken Language Dialogue Systems, CPK Aalborg University, CCS Roskilde University, CST University of Copenhagen. To appear early 1994.
9. Dybkjær, L. and Dybkjær, H. “Wizard of Oz Experiments in the Development of a Dialogue Model for P1”. Report 3, Spoken Language Dialogue Systems, STC Aalborg University, CCS Roskilde University, CST University of Copenhagen. February 1993.
10. Fraser, N.M. and Gilbert, G.N. “Simulating Speech Systems”. Computer Speech and Language 5, 81-99, 1991.
11. Larsen, L.B., Brøndsted, T., Dybkjær, H., Dybkjær, L. and Music, B. “Overall Specification and Architecture of P1”. Report 2, Spoken Language Dialogue Systems, STC Aalborg University, CCS Roskilde University, CST University of Copenhagen, February 1993.
12. Larsen, L.B., Brøndsted, T., Dybkjær, H., Dybkjær, L., Music, B. and Povlsen, C. “State-of-the-art of Spoken Language Systems - A Survey”. Report 1, Spoken Language Dialogue Systems, STC Aalborg University, CCS Roskilde University, CST University of Copenhagen, September 1992.
13. Lindberg, B., Kristiansen, J. and Andersen, B. “SUNCAR Functional Description”. Esprit Project 2094 SUNSTAR, STC.WPIV.008, March 1992.
14. Povlsen, C. and Music, B. “Definition and Specification of the Sub-language for P1”. Report 4, Spoken Language Dialogue Systems, CPK Aalborg University, CCS Roskilde University, CST University of Copenhagen. To appear early 1994.

15. Zoltan-Ford, E. “How to Get People to Say and Type what Computers can Understand”. International Journal of Man-Machine Studies 34, 527-547 1991.

Acknowledgements

The work described in this paper was carried out under a grant from the Danish Government’s Informatics Research Programme whose support is gratefully acknowledged. Thanks are due to Tom Brøndsted, Anders Bækgaard, Paul Dalsgaard, Lars Bo Larsen, Børge Lindberg, Brad Music and Claus Povlsen for helpful comments.