WIZARD OF OZ PROTOTYPING: WHEN AND HOW?

Niels Ole Bernsen, Hans Dybkjær and Laila Dybkjær

Centre for Cognitive Science
P.O.Box 260, Roskilde University
4000 Roskilde, Denmark
+45 46 75 77 11, nob@cog.ruc.dk, dybkjaer@ruc.dk, laila@cog.ruc.dk

Abstract: Progress in speech and language processing and multimodal systems technologies has led to the fact that prototyping with the Wizard of Oz (WOZ) system simulation technique is increasingly being used in systems design. When prototyping with WOZ, one or more 'wizards' simulate part or whole of the performance of the system being designed, while interacting with users who preferably believe themselves to be using a real system. A series of WOZ iterations has the potential to deliver a more or less complete specification of the system's input/output behaviour which can then be safely implemented. This paper addresses the need for a systematic presentation, conceptualisation and discussion of the WOZ technique as a systems design method: when should WOZ be preferred to other prototyping techniques which are probably less demanding in resources? What is needed to set up WOZ experiments? When should a series of WOZ iterations start and stop? And what are the main problems in designing with WOZ? The paper is based on WOZ experience in unimodal systems design but it is hoped that its open-ended generalisations may be of use to designers of multimodal systems as well.

Keywords: Wizard of Oz, prototyping, systems design methods, usability engineering.

1. Introduction

Many usability engineering design methods have found their way into early systems design practice [17]. Common to these methods, whether they involve field studies, thinking-aloud or rapid prototyping, is that they elicit *partial* information on aspects of user-system interaction, which designers use to improve the functionality and usability of the artifact being designed. By contrast, the Wizard of Oz simulation technique (WOZ) promises to deliver complete information on user-system interaction. 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 [5]. This means that, ideally, the end result of the WOZ specify-and-simulate test cycle will be a simulated system which can safely 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. The term 'cognitively demanding' characterises tasks which are relatively easy for humans to perform but generally difficult for current machines. As such tasks increasingly form part of the input/output capabilities of multimodal systems, WOZ has a claim to becoming an important prototyping methodology in the near future.

Until recently, WOZ has been used more in psychological and linguistic research than in systems development [1, 14]. As a systems design method WOZ has been applied almost exclusively in the design of spoken or typed natural language systems (reviewed in [11]). Use of WOZ for multimodal systems design has recently been reported in [6, 16, 18]. We propose to complement these developments by presenting and examining WOZ as a practical systems development technique based on experience with WOZ in the design of a spoken language dialogue system in the domain of air travel reservation and information [2, 7, 8]. A system prototype has been implemented and is now running. Design of spoken language dialogue systems requires the use of WOZ in both input understanding and output generation and may be considered representative of the problems involved in using WOZ in the design of a broad but not yet precisely specifiable class of unimodal or multimodal systems. Thus, spoken dialogue is more difficult to simulate than typed dialogue, and dialogue is, ceteris paribus, more difficult to simulate than either input understanding alone or output generation alone. We present a walkthrough through general aspects and problems involved in using WOZ in systems development, organised around types of system for which use of the method should be considered; the WOZ machinery; the iterative WOZ process; and discussion of the limitations of WOZ and its potential for supporting multimodal systems development.

2. Wizard of Oz in the design process

WOZ is not equally suited to support all design processes. We propose to delimit the design process types for which WOZ should be considered, as follows. *Firstly*, the interactive system behaviour to be simulated should be behaviour which humans are good at performing. This class of behaviour includes cognitively demanding skills which humans learn to master from early on. However, there does not seem to be any reason in principle why later acquired expertise might not also be considered for WOZ simulation. *Secondly,* as systems with such interactive skills are still difficult to build, it is necessary to focus on the design of systems having relatively narrow and well-defined application domains as far as their cognitively demanding task aspects are concerned. Note that multimodal systems may include cognitively demanding tasks as part of their input/output processing capabilities while, e.g., standard graphical user interfaces including keyboard and mouse serve the rest of the interaction with users. *Thirdly,* as

WOZ is no 'quick and dirty' prototyping method but somewhat demanding in resources, the system to be built should be high-risk in the sense that the cost-times-risk of having to re-build the artifact more or less from scratch after prototype failure is sufficiently high to warrant the investment in a more costly but strongly risk-minimizing prototyping technique. We hypothesize that systems performing cognitively demanding tasks are generally also high-risk ones. Conversely, for interactive tasks which are not cognitively demanding, there are likely to exist rapid prototyping methods which are preferable to WOZ in cost-benefit terms. *Finally*, cognitively demanding interaction often, if not always, relies on natural and spontaneous user input behaviour such as gesture or spoken or written discourse. The technology will normally enforce restrictions on the system's capacity for understanding spontaneous user input. In such domains, realistic artifact development should only be undertaken if there is some way of ensuring that the user input which the system can understand is not restricted in unnatural or unprincipled ways. If such restrictions obtain, input production will be practically impossible for users [8]. For instance, whereas users may quickly learn to practice short input sentences, unnaturally restricted grammar, on the other hand, can make a system practically useless. WOZ offers mechanisms which support the detection of unnatural or unprincipled restrictions on user input.

3. Setting up the Wizard of Oz

Figure 1 shows the general setup of a WOZ simulation. In practice, details may vary considerably. Examples are found in [8, 10,]. All simulations involve a subject acting as user, at least one person (the wizard) simulating part or whole of the interactive behaviour of the system, and a subject-wizard interface which hides the fact that the subject is interacting with a human rather than a real system. This section presents a walkthrough through Figure 1.

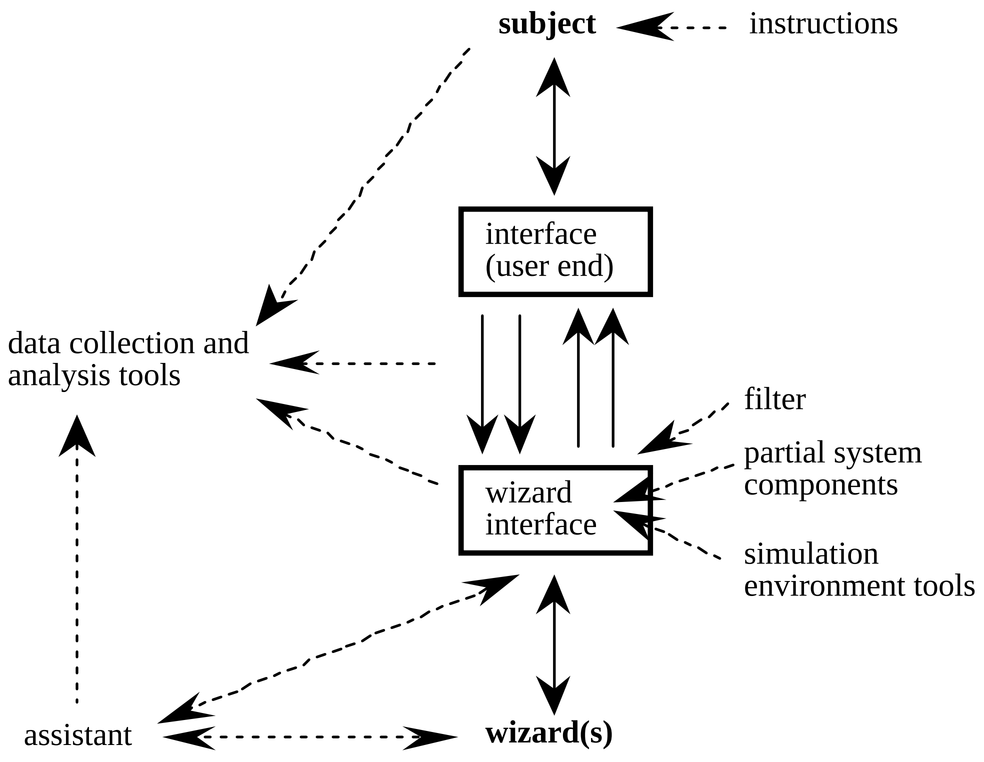
subject
instructions
interface (user end)
data collection and analysis tools
filter
partial system components
wizard interface
simulation environment tools
assistant
wizard(s)

Figure 1: General setup of a WOZ simulation. The main communication line is along the solid arrows.

3.2 The system (wizard) side

The system simulation side consists of wizard(s) and wizard support. In the representative case we are considering, system simulation covers input understanding and output generation including appropriate response times which are an important factor in the evaluation of the usability of computer artifacts. System simulation involves two closely related tasks: to *determine* the input/output behaviour of the final system and to *simulate* this behaviour as closely as possible. There is evidence that people's communication with computers differs from their communication with humans (discussed and reviewed in [1]). Thus, as long as the computer demonstrates appropriate functionality, people are prepared to simplify their input behaviour and to accept simplified system output. The simulation should capitalize on these simplifications by maintaining subjects in the belief that they communicate with a real computer system.

Wizard

The wizard's task is hard primarily because of the high demands on working memory which result from the number and difficulty of the tasks the wizard has to simultaneously perform during interaction. Countermeasures include careful training of the wizard and good support tools. Response time measurements are useful when judging whether the wizard needs more training or support. Wizard training starts before the simulations begin, continues during simulation and pertains to application domain knowledge, the intended system's skills and how to use the support tools.

We found that the wizard's main problem was that of having superior knowledge and skills compared to the intended system. Such superiority tends to make the wizard understand input that the final system cannot understand as well as generate responses beyond the capacity of the final system. However, as long as the superior knowledge of the wizard is declarative and can be explicitly represented more or less easily, it seems that the problem can be solved through practise and external support without generating unrealistically long response times. Our wizard, for instance, was supplied with a list of standard response phrases to be used consistently in similar circumstances. The real problem seems to be the wizard's unavoidable possession of superior skills due to the fact that many aspects of cognitively demanding tasks are skill-based. The wizard must, for instance, consistently simulate limited language comprehension in terms of vocabulary, semantics, grammatical complexity or flawed or non-standard user input, or limited language generation in terms of rhythm and intonation. Reduced skills are much harder to simulate in close-to-real-time than is reduced declarative knowledge, partly because skills are automated and partly because efficient external support is more difficult or even impossible to provide.

Wizard Support

The system end of the interface (the *wizard interface*) normally includes an artificial interface medium such as a telephone or computer screen combined with support tools such as *filters, partial system components and simulation environments* (see Figure 1). The wizard must be able to operate this interface quickly and reliably.

Filters are hardware/software tools inserted on the communication channel for manipulating input or output during simulation. Filters on system output serve to support subjects' belief in communicating with a real system and both output and input filters may help the wizard perform at the system’s expected level of skills. Examples of filters are vocoders for distorting spoken output or input, speech synthesizers, filtering of typed input according to whether it belongs to the system's lexicon or not and, in the case of typed output, response facilities which hide the wizard’s typing rate and correct misspelled words. We used an equalizer/harmonizer combination to distort the wizard’s voice and no input filters. In contrast to much of the literature, e.g. [12], it turned out that voice output filtering had no significant effect on user performance nor on subjects' beliefs about the system [7]. Our hypothesis is that the potential effects of output filtering had in this particular case already been achieved by a strongly system-directed dialogue in combination with the wizard's (mostly successful) use of monotonous voice and controlled intonation. When the dialogue becomes less system-directed, voice output filters may still have useful roles to play, including that of bringing speech output quality closer to that of the final system in the case of synthesized speech systems.

Input quality affects recognition and may vary widely. A telephone, for instance, affects the quality of speech and so does the speaker's voice. To support simulation of the final system's expected input misrecognition rates, input may be distorted [13] or, better still, a real input recogniser inserted as a partial system component (see below). To support the wizard at the skill-based level, principled decisions should be made as far as possible on how to handle, e.g., non-standard accents, dialects, indistinct voices, pauses, input/output overlaps and interruptions, turntaking cues, etc., and simulation of the corresponding error-recovery mechanisms should be trained.

Partial system components are completed modules of the system which is being developed. An increasing number of such components, such as databases, speech recognisers or speech synthesizers, may be incorporated over time. The components act as support tools in that they allow the wizard to concentrate on other aspects of the simulations and may help reducing response times. However, they also require the wizard to act as intermediary and sometimes as operator. Several cooperating wizards may be needed if there are several system components [18].

Simulation environment tools may be manual or automatic and are defined negatively as wizard support which is non-human and does not fall into the categories of filters and partial system components. Examples are typed or pre-recorded phrases to support reduced language generation skills, an explicit dialogue model to support consistent performance or paper notes on user input during dialogue. When automatic simulation environment tools are used, the wizard usually has a screen with a set of windows, one for each functionality such as one window showing information provided by the user and another showing the wizard’s dialogue decisions so far. To support dialogue interpretation and output selection we used a graph-structured simulation environment for keeping track of the dialogue, with system actions in the nodes and expected subject reactions on the edges. The wizard’s task was to decide which edge to follow on a given input.

An *assistant* is a person assisting the wizard on the system simulation side without communicating with the subjects. It is strongly recommended to unload the wizard in this way. The assistant may share part of the simulation environment tools with the wizard, e.g., by taking notes during the wizard’s interaction with subject. The assistant may act as the wizard’s interface to partial system components and help in operating other parts of the equipment including the data collection and analysis tools (see below). To minimize reponse times and reduce error, the assistant needs training.

3.2 The user side

The user side consists of subjects who are instructed on their roles. To support subjects' belief in interacting with a real system, the user end interface medium (or media, such as telephone, screen, keyboard, and camera) should be the same as in the final system and the interface itself should iteratively approximate that of the final system.

Subjects

Even in early WOZ iterations, subjects should be selected such that their backgrounds and skills correspond to those of the expected end-users. This is not only a question of being novice or expert in the domain. Subjects' educational background seems to influence the way in which they communicate with the system [8]. It is therefore not sufficient to just ask students or colleagues to behave as if they were a certain type of person. Preferably also, subjects should act in their habitual environments in order to make the setting as realistic as possible. As for numbers, we used only a couple of subjects in the first five iterations which mainly served the purpose of training the wizard. In each of the last two WOZ iterations 12 subjects were used. A total of 16 of subjects were external, new to the simulations and representative of the expected end-users. The rest were colleagues some of which had participated in one of the earlier iterations.

Instructions

Since it is desirable that subjects believe that they are interacting with a real system they should not be told the truth about it in advance. Neither should they be told a lie for ethical reasons. Rather, they should be given vague information which most obviously may be interpreted as if the system were real. Unless the experiments are of a kind where no instructions are provided and subjects do not know that they are acting as such [10], care should be taken to ensure that subjects know exactly what they are expected to do and are able to perform their tasks in as natural a way as possible.

In most cases, subjects are given scenarios to perform. We are referring here to scenarios for system *development* [4] which are intended to more or less systematically cover the intended system functionality. Scenarios are normally designed by the system designers. User-designed scenarios will typically not be appropriate for the purpose of system development. However, the risk in using only designer-designed scenarios is that designers may ignore important task aspects and other constraints, ending up with an implemented system which works well only in a fictitious world. It is therefore recommended that scenarios for *evaluation and testing* of the system be developed jointly by designers and end-users. Another important point is that scenarios should not provide a too detailed task description. Rather, subjects should fill in the details by themselves. We found that subjects tended to model the language used in the scenarios [15], which implies the risk that the language understood by the final system would be that of the designers rather than that of the end-users.

3.3 Data collection and analysis

Interaction via the user interface and activity at the wizard interface should be logged and recorded for later analysis. Results of the analyses are used to improve the simulated system as a basis for subsequent iterations. The input/output modalities involved constrain the choice of hardware for data-logging. If the only modality involved is spoken language, a tape recorder will be sufficient. If the modality is typed and/or hand-written language, a computer log can be made of input and output. Video is needed for recording gesture, facial expression, visual scenes and the like. In multi-modal interaction, a combination of data recording hardware will be necessary. In general, as WOZ simulations tend to generate large amounts of data, there is a strong need for improved facilities for data-filtering, indexing, transcription and analysis [18]. We found that data-analysis had to be strictly focused to be feasible within the time constraints on the design process. It became focused on the parameters crucial to technological feasibility, such as sub-language vocabulary, mean utterance length and dialogue structure coherence and consistency, and the majority of usability problems were identified through iterative task structure analysis motivated by the simulated interactions with users rather than from in-depth analyses of the recorded and transcribed interactions [3]. In addition to the data types on user performance already mentioned, questionnaires given to subjects who had interacted with the system proved very valuable.

4. Iterative design and evaluation

Each WOZ iteration is costly to prepare, run and analyse. This is particularly true of the key iterations which involve external subjects. Having started our WOZ work without sufficient operational guidance from the literature we now believe that, given careful planning of the series of iterations and awareness of the problems, the results we obtained from 7 iterations could have been achieved in 3 to 4 iterations. However, before involving 'real' external subjects it makes sense to run the simulation setup with the designers themselves and perhaps a few colleagues. WOZ iteration raises the three questions on how to begin, how to iterate and when to stop, which will be discussed in that order.

4.1 How to begin?

WOZ is not a stand-alone usability engineering design method but is based on a number of initial design decisions concerning overall design goals, technological and other feasibility constraints on the design process, various criteria to do with the realism, functionality and usability of the artifact, choice and delimitation of application domain, target user types and so on [2]. Furthermore, and based on such decisions, the WOZ iterations must start from a preliminary task model, interface model and system model. Unless fed with such information, WOZ risks producing only iterations over a designers' fictitious task domain. In other words, standard requirements capture and usability engineering methods (cf. introduction) are needed to determine the *whats?* and *whys?* and part of the *how?* of interaction. Only then may the *how?* of user-system interaction be developed in detail with WOZ. To establish the initial task model, we built the interactive structure around a number of *basic user tasks* which the system was intended to support [9].

The wizard needs training. The best way of initially training the wizard is to let two system designers act as subject and wizard, respectively. This will give the wizard experience in acting at the system's level of skills, provide domain knowledge training and familiarity with the equipment used. The wizard interface should be adjusted if there are problems. While the resulting data will hardly be reliable enough to serve as a basis for implementation, this first simulation will provide rough estimates of system and user performance and allow new constraints to be added and unforeseen problems solved. Also the data collection and analysis tools are tested in this initial phase. For instance, it turned out to cause unexpected problems to connect our voice distorting hardware to the telephone line. A list of potential subjects should be prepared together with a set of instructions. In addition to general information on the simulations and their purpose, the instructions include a carefully prepared set of scenarios which should aim at covering the entire interactive task domain. The material is mailed to subjects who agree to participate.

4.2 How to Iterate?

Over a series of iterations, WOZ delivers a detailed specification of user-system interaction. Each iteration consists in a test of interactive system design (the simulations) followed by an evaluation based on analysis of the collected data. It is recommended to mail each subject a questionnaire together with the instruction set. Subjects are asked to fill in and return the questionnaire immediately after their participation. The questionnaires should be analysed to identify specific problems, general complaints, subjects' overall impression of the system, etc. In addition, it is recommended to phone subjects shortly after their participation to ask about their impression of the system. At this point they should be told that the system was being simulated.

Each iteration produces large amounts of *quantitative data*, for instance: data on subjects' sublanguage vocabulary in the task domain (full word types, word stem types, non-words), utterance-length (average-per-turn, maximum), type/token ratio, number of turns per task scenario (average, maximal), percentages of questions and statements, grammatical complexity, ungrammatical phenomena, hesitations and false starts, and number and types of discourse phenomena (anaphora, ellipsis, etc.). Data on wizard performance can be just as important, for instance when measuring against training target levels (number of deliberate recognition errors, number and types of errors due to 'over-skilled' performance) or when measuring the effects of intended changes in communication style (e.g., when the wizard is required to talk less per turn). As such data are obtained by transcribing the simulations and counting the relevant phenomena, there is ample need for time-saving, special-purpose automatic analysis tools. Secondly, in addition to quantitative information there is often a need for analysing *structural information* such as variations in the expression of identical messages, users' task or sub-task ordering preferences and stereotypes, their problem-solving strategies, etc.

Detection of the *problems users have* in interacting with the system constitutes a third important goal of data analysis, as each problem suggests a need to change the design. A practical method for revealing user problems is to make walk-throughs of the transcriptions and match observed user behaviour against that which had been predicted by the designers in advance. Such comparisons often lead to design changes. A systematic study of user problems identified during the WOZ-supported design of user-system interaction in our dialogue system revealed 16 different user problem types [3].

Fourthly, identification of *developmental patterns* in data across a series of iterations may be important for several reasons. One is to measure the extent to which specified technological feasibility constraints on the system have been met, such as the average user utterance length which in our system was set to four units (words). A second is to ascertain the effects of interaction design changes. Systematic changes in data patterns may occur as a result of manipulations of the system's interface, ranging from major changes in task domain coverage to subtle changes in the semantics of system utterances. In the design of systems undertaking cognitively demanding tasks, quantitative data and development patterns such as those exemplified above are particularly important because the system has to be able to *interpret natural and spontaneous user behaviour*. In some cases, such as anaphora resolution, we still lack part of the theoretical understanding that may make this possible in the general case, and in a larger number of cases we lack the tools and algorithms necessary to implementation. This means that the capabilities to manipulate developing patterns in user behaviour and to accurately measure the effects of interface manipulations are essential to successful design. User behaviour must be brought within the boundaries of current scientific and technological constraints while maintaining its naturalness [8]. If and when this has been done through iterative design with WOZ, and when the user problem aspects have been taken care of, WOZ can deliver a close-to-complete specification of user-system interaction for implementation.

4.3 When to stop?

Given the number of unknowns which are normally involved in systems design with WOZ, it cannot be decided in advance how many iterations are needed to obtain a complete specification of user-system interaction. The decision to stop must be based on evaluation of results from the last iteration. Subjects should have no more substantial problems during interaction and all feasibility constraints should be satisfied. For instance, if one of the constraints is that the system can only recognise a limited vocabulary of a certain size, it is important to verify that the vocabulary used by subjects converges at zero and hence is sufficient to the execution of all tasks within the domain. If this continues not to be the case then either the vocabulary constraint must be changed (relaxed) or the dialogue structure should be changed to induce further restrictions on users' sublanguage. In the case of spoken language dialogue systems, there is evidence that subjects tend to use longer utterances when addressed politely by the system than when addressed in a terse manner [20]. Another important point is that the data must be sufficient to permit the relevant conclusions to be drawn. This should be ensured by having a sufficiently large number of subjects each performing several tasks during a number of iterations.

5. Conclusion

The WOZ simulation technique seems mandatory in the design of high-risk, cognitively demanding systems. WOZ simulations must be based on the use of standard methods for requirements capture and usability engineering. In this context, WOZ elicits much more complete information than other rapid prototyping techniques as it may deliver a close-to-complete specification of user-system interaction for implementation. However, both the quality of the specification produced and the resources required for producing it strongly depend on how well the simulations have been planned, trained, executed and iteratively evaluated. Lack of attention, before as well as during a series of WOZ iterations, to the implications of the serious scientific and technological constraints which currently characterize cognitively demanding systems development may easily lead to the wasted effort in the specification of non-implementable or non-usable systems. Given appropriate attention to the specific feasibility/usability trade-offs which characterize the type of artifact to be designed, the main weakness of WOZ simulation is the wizard's difficulty in simulating inferior skill-based behaviour. Countering these difficulties requires careful analysis of the data produced by the simulations. Improved data analysis tools could significantly reduce the cost of performing WOZ simulations.

Having only used WOZ in unimodal systems design, we are aware of the limited nature of the generalisations presented above and more work is needed on generalising the WOZ methodology to cover complex interface modality combinations. The literature is still sparse. [6] discusses simulations of graphical direct manipulation combined with written natural language. [16] describes experiments with a combination of mouse and speech. The Neimo system seems to represent the only attempt so far at building a general multimodal WOZ platform [18]. Extending the WOZ methodology to multimodal systems design requires consideration of aspects which have not been discussed above, such as how to make several wizards act consistently together and how to analyse the complex data produced.

Acknowledgement

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.

References

1. Amalberti, R., Carbonell, N. and Falzon, P. User Representations of Computer Systems in Human-Computer Speech Interaction. *International Journal of Man-Machine Studies 38*, 1993, 547-566.

2. 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, 1993, 221-244.

3. Bernsen, N.O. Types of User Problems in Design. A Study of Knowledge Acquisition Using the Wizard of Oz. *Esprit Basic Research project AMODEUS Working Paper* UM/WP 14, 1993.

4. Campbell, R.L. Will the Real Scenario Please Stand Up? *SIGCHI Bulletin* 24, 2, 1992, 6-8.

5. Carroll, J.M., Kellogg, W.A. and Rosson, M.B. Getting Around the Task - Artifact Cycle: How to Make Claims and Design by Scenario. *ACM Transactions on Information Systems*, 10, 2, 1992, 181-212.

6. Dahlbäck, N., Jönsson, A. and Ahrenberg, L. Wizard of Oz-Studies— Why and How. *Proceedings from the Workshop on Empirical Models and Methodology for Natural Language Dialogue Systems*. Trento, Italy, March 1992.

7. Dybkjær, L., Bernsen, N.O. and Dybkjær, H. Knowledge Acquisition for a Constrained Speech System using WOZ. *Proceedings of the Sixth Conference of the EACL*. Utrecht, April 1993, 467.

8. 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 1993. In Press.

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, CCI Roskilde University, CST University of Copenhagen*. February 1993.

10. Francony, J., Kuijpers, E. and Polity, Y. Towards a Methodology for Wizard of Oz Experiments. *Proceedings from the Workshop on Empirical Models and Methodology for Natural Language Dialogue Systems*. Trento, Italy, March 1992.

11. Fraser, N.M. and Gilbert, G.N. Simulating Speech Systems. *Computer Speech and Language* 5, 1991, 81-99.

12. Fraser, N. M. and Gilbert, G.N. Effects of System Voice Quality on the User Utterances in Speech Dialogue Systems. *Proceedings of EUROSPEECH '91*, 57-60.

13. Guyomard, M. and Siroux, J. Experimentation in the Specification of an Oral Dialogue. Niemann, H., Lang, M., and Sagerer, G., Eds. *Recent Advances in Speech Understanding and Dialog Systems.* NATO ASI Series, Vol. F46, 497-501. Berlin, Springer Verlag, 1988.

14. Hauptmann, A.G. and Rudnicky, A.I. Talking to Computers: An Empirical Investigation. *International Journal of Man-Machine Studies* 28, 1988, 583-604.

15. Klausen, T. Talking to a Wizard. Report from the Design of a Natural Speech Understanding System. *Esprit Basic Research project AMODEUS Working Paper* UM/WP 11, 1993.

16. Maulsby, D., Greenberg, S. and Mander, R. Prototyping an Intelligent Agent through Wizard of Oz. *Proceedings of INTERCHI '93*. Amsterdam, April 1993, 277-284.

17. Nielsen, J. *Usability Engineering*. New York, Academic Press, 1993.

18. Salber, D. and Coutaz, J. A Wizard of Oz Platform for the Study of Multimodal Systems. *Adjunct Proceedings of INTERCHI '93*. Amsterdam, April 1993, 95-96.

19. Sanderson, P.M. Designing for Simplicity of Inference in Observational Studies of Process Control: ESDA and MACSHAPA. In Hollnagel, E. and Lind, M., Eds. *Proceedings of the Fourth European Meeting on Cognitive Science Approaches to Process Control: Designing for Simplicity*. Copenhagen, August 1993, 19-47.

20. Zoltan-Ford, E. How to Get People to Say and Type what Computers can Understand. *International Journal on Man-Machine Studies* 34, 1991, 527-547.