

WebWOZ: A Wizard of Oz Prototyping Framework

Stephan Schlögl
Trinity College Dublin
Ireland
schlogls@scss.tcd.ie

Gavin Doherty
Trinity College Dublin
Ireland
Gavin.Doherty@scss.tcd.ie

Nikiforos Karamanis
Trinity College Dublin
Ireland
Nikiforos.Karamanis@scss.tcd.ie

Saturnino Luz
Trinity College Dublin
Ireland
Saturnino.Luz@scss.tcd.ie

ABSTRACT

Language Technology (LT) based applications become more popular as technology improves. Prototyping early in the design process is critical for the development of high quality applications. It is difficult, however, to do low-fidelity prototyping (e.g. paper prototyping) of applications based on LT. One technique that has been used for this kind of prototyping is Wizard of Oz (WOZ). However, this generally involves the development of one-off user and wizard interfaces. A tool that facilitates the flexible integration of LT components into WOZ experiments is desirable. In this paper we explore the requirements for such a tool, drawing from the literature and a first WOZ experiment in which different wizards were observed and their behaviour was analysed.

Author Keywords

Wizard of Oz; Prototyping; Language Technology.

ACM Classification Keywords

H.5.2 User Interfaces: Prototyping; H.5.2 User Interfaces: Natural Language; D.2.2 Design Tools and Techniques: Modules and interfaces.

General Terms

Design, Human Factors.

INTRODUCTION

We report on work in progress that aims to build a novel prototyping framework for Wizard of Oz (WOZ) experiments. The focus lies in supporting the flexible integration and usage of language technology components (LTC). Such components are seeing increasing deployment in a variety of consumer applications. Examples include automatic speech recognition (ASR) used in cars to keep a drivers' attention on the road [15], interactive voice response systems using combined ASR and text-to-speech synthesis (TTS) technologies

to automate processes such as buying tickets [13], and the increased usage of web-based machine translation (MT) tools such as Google Translate. Developing and tuning applications incorporating language technology to the point where they can be used with end users can be both time and cost intensive, which makes it difficult to explore design options early in the development lifecycle. Hence effective ways of testing early-stage prototypes are needed.

Existing tools and evaluation techniques have generally either focussed on the technology, i.e. aiming to improve the quality of single LTCs (e.g. using benchmarks in the development of speech recognisers), or explore only the users perspective through completely replacing the LTC by a human (e.g. pure WOZ tools). In both cases test results tend to inform only one side of the interaction. An alternative would be to provide tools that support human augmentation of LTC results, in a WOZ environment, and hence allow for exploration of the interaction with a proposed technology over several iterations, and potentially with increasingly effective language technology components as the system is tuned for the application context.

Dialogue management tools like the CSLU toolkit [27] or low-fidelity WOZ tools like SUEDE [16] are less suitable for this kind of prototyping. Dialogue management tools typically allow a relatively limited range of application types to be prototyped, and include few features to support WOZ prototyping. Similarly WOZ tools offer very limited coverage of the application space and do not support mixed-fidelity prototyping in which language technology components are used to process input and output, as well as the human wizard. Our goal is therefore to design a WOZ prototyping environment that facilitates an easy integration of LTCs (i.e. ASR, TTS, MT, etc.) into WOZ experiments. The idea of WOZ experiments is that a "human wizard" mimics a computer system. The wizard requires a special-purpose user interface to communicate with test participants. The interface serves two purposes. First it allows the wizard to select responses for the participant, allowing the dialogue to proceed, and preferably providing for a consistent (and in some cases mechanistic) behaviour which is needed to make test participants feel that they are interacting with a real system [6]. Second, it is meant to support the cognitively highly demanding task of the wizard [24] by allowing them to keep

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

EICS'10, June 19–23, 2010, Berlin, Germany.

Copyright 2010 ACM 978-1-4503-0083-4/10/06...\$10.00.

track of the dialogue history, dialogue state, and available options.

In this paper we discuss the requirements for our framework. Identifying these requirements started with an analysis of the literature and existing WOZ and dialogue management systems, we then gathered and analysed a number of experimental scenarios, with associated sketching and prototyping activities regarding possible tool support, and conducted a WOZ experiment in which several wizards were observed and their behaviour analyzed, with subsequent retrospective interviews. A web-based architecture for tool support is proposed based on this analysis.

PREVIOUS WORK

The first use of Wizard of Oz (WOZ) as a design method for early-stage prototypes occurred almost 40 years ago when Erdman & Neal [7] tested the concept of a self-service airline ticket kiosk. In general, it has been applied to inform the design of systems incorporating technology components which do not yet exist, or for which the significant engineering effort required for implementation was inappropriate at the prototyping stage. As such it for instance facilitates the development of rich dialogue models or supports the exploration of the naturalness of an interaction. Several researchers have highlighted the importance of early user studies informing the design and development of computer systems (e.g. [10] [3]). In the case of natural language interfaces, examples show that WOZ studies are seen as an effective way of achieving this. That is, as opposed to assuming a certain dialogue flow, WOZ experiments help to explore the dialogue space in more detail [6].

WOZ and Language Technology

In recent years WOZ has been used in various projects dealing with natural language in order to gain early feedback on user behaviour. [22] used it to test their multi-modal access to a mediadata base and [13] as an analysis instrument for their spoken dialogue system. Other published examples include simulating a virtual doorman [20] and testing a web based companion [2]. Aspects that were explored comprise dialogue design, multimodal information retrieval, human agent interaction and emotion. In doing so two different categories of tools for conducting experiments can be distinguished. On the one hand there are Dialogue Management (DM) tools, which are mainly focusing on the evaluation of LTCs. On the other hand one finds specialized WOZ tools that make less use of already existing technology but rather test novel ideas by relying on a human simulation.

Dialogue Management Tools

Examples of DM tools are the CSLU toolkit [27], CMU's Olympus dialogue-framework [1] and the Jaspis dialogue management system [28]. The main goal of DM tools is to test the language-based interaction between a human and a machine and design the dialogue flow. They integrate various LTCs like ASR, TTS and for instance Talking Heads, under some kind of Rapid Application Development framework. As such their focus lies on testing and improving

technology components. A disadvantage of this is, however, that when dialogues are designed the DM tools do not allow for external intervention during runtime. Therefore, the dialogue success solely depends on the quality of the used LTC, which restricts their ability to explore experiences users could have but are not yet supported by the technology. Exploring the user experience before investing in technology is, however, an important part of what WOZ studies aim to do. Hence, the strong focus on existing technology limits the suitability of DM tools for WOZ experiments.

Pure WOZ Tools

The second category of tools used focuses mainly on the design and conduct of pure WOZ experiments. While these also start with the design of a human-machine dialogue, no LTCs are typically used. Instead a human wizard acts as a machine and chooses from different predefined response utterances. The main goal of WOZ tools such as SUEDE [16] is to inform the dialogue design and investigate user experience. They allow for more flexible interactions, and support the exploration of user experiences that might not yet be supported by available LTCs. The outcomes of those studies are not technology dependent but rather rely on the skills of the wizard. Hence, the quality aspect of tests is suddenly shifted from the reliability of a technology component to the consistency of a human, which, however, makes the wizard's task cognitively highly demanding [24].

LANGUAGE TECHNOLOGY COMPONENTS

LTCs as for instance ASR and TTS engines have been successfully integrated into several applications, with interactive voice response systems receiving widespread commercial deployment. Systems like Let's Go [23] and DARPA communicator [31] were used to provide customers with schedule information over the telephone for flights [13], trains [17] and buses [29]. Speech technologies have been integrated with MT in prototypes which support multilingual communication during meetings [30], doctor-patient consultations [25] and travelling [21]. MT for more general purposes is available online, e.g. by Google Translate which currently supports translation from/to 52 languages and integrates text analysis (by automatically recognizing the input language) and TTS (currently only for English output).

In addition to these scenarios, LTCs are also used in settings in which traditional input and output modalities are less appropriate. Hands-busy eyes-busy situations, for instance require special interaction techniques. In-car navigation, route planning and other services such as electronic car manuals and interactive hotel reservation systems are now increasingly accessible via speech input and combined speech/map output [9]. Moreover, in the areas of tutoring systems and edutainment as well as in the health care sector [14] language technologies are used in several multi-modal settings. Here mainly prototypical implementations of Embodied Conversational Agents show interesting application possibilities, including areas such as computer-based mental health care [5] and helping people learning about computer literacy and physics [11].

While the use of language technologies is undoubtedly increasing, several barriers remain to widespread its application. Language technology components are unavoidably imperfect, particularly in the case of ASR. Although state of the art recognition algorithms promise word error rates (WER) as low as 20% [12] and less than 10% within specified domains [26], different accents and dialects can increase this value significantly [19]. To improve this examples have shown that domain data can be used to train speech recognizers. [8] for instance uses previously used material like transcribed speech from meetings and presentations to decrease the WER for public speeches and lectures. A similar approach can be applied to increase the quality of MT. Thus a major challenge in the design and development of language technology applications is the need for domain-specific collections of text (corpora). Application designers and engineers need to invest significant time and effort in building up these resources in order to tune the components to an acceptable quality level.

On the other hand, some studies show that even if the used technology is flawed, people may be able to successfully accomplish given tasks. It has been demonstrated that by adapting the dialogue in cases where the ASR is poor, the overall user satisfaction can be increased [18]. Thus, we see three separate issues which might be supported through an effective prototyping tool: facilitating interactions with a (WOZ) prototype, so that language resources can be gathered; exploring interaction strategies to allow the users to overcome the limitations of the technology; and investigating what level of quality is required from the LTC's in order to deliver a satisfactory user experience, and allow the users to successfully complete their tasks. These correspond to the major "use cases" for the tool - gathering language resources, exploring interaction design, and evaluating language technology components.

Discussion

In summary, previous work has shown that WOZ can be used to inform the design of applications that incorporate language technologies. Tools that enable testing various LTCs in different settings, however, are missing. In addition it was identified that the task of the wizard is cognitively highly demanding and the use of contextual information and domain-specific knowledge may be used to leverage error prone technology components. It is therefore important to provide methods and tools that help the wizard, allow for a flexible integration of LTCs, and support the prototypical exploration of different application scenarios. Existing prototyping tools do not meet these requirements. To date two different types of tools are used, both of which have their advantages and drawbacks. In order to combine their qualities, i.e. the technology support of DM and the flexibility of WOZ tools, we propose to create an environment that supports the easy integration of LTCs into a WOZ framework. This framework should on the one hand allow for a flexible usage of different LTCs. On the other hand, it should provide ways of capturing and analysing contextual information and domain knowledge. As such, we aim for a platform that supports the conduct and analysis of experiments, and incorporates

modules like ASR and MT, as well as human interventions. Unlike other tools, however, we treat the DM as simply another component that can be parameterised or replaced according to the needs of the design to be tested. In doing so the LTCs are not only evaluated, but also used to support the wizard's task and make the experiment outcome more consistent. Finally we also aim for an easy integration of additional modalities.

A PROPOSED FRAMEWORK

Following a review of existing tools, a range of scenarios for WOZ and mixed-fidelity experiments involving LTC's was gathered (see Figure 1). These were used to produce a number of prototype designs and sketches, with a view towards producing a framework which is adaptable to different dialogue models and scenarios.

Scenario 1: A multilingual conversation
A Smartphone application used to translate between a Japanese taxi driver and an english-speaking business traveller.
Scenario 2: Location based storytelling
A travel device that tells stories to tourists triggered by their current location.
Scenario 3: The multilingual instructor
A multilingual 'Avatar' that talks a user through the setup process of a video game console.
Scenario 4: The adaptive helpdesk
An InfoPoint terminal that advises customers about the best internet connection that fulfils their specific requirements.

Figure 1. Scenarios that were used to produce prototypes and sketches.

A main issue that was identified during this process is that a future framework would need to allow for an easy integration of components. One click should be enough to incorporate them into a wizard interface. In doing so it seems important to incorporate technology that helps the wizard but keeps her in control of the dialogue. For instance, a built-in ASR module could recommend possible response utterances so that the wizard only needs to search through a list of responses in cases where the recognizer fails. Other examples would include a MT module supporting the wizard in a multilingual test setting or a Natural Language Understanding (NLU) module recommending the next utterance. Also it seems that there is a need for supporting the analysis of conducted experiments. Therefore the proposed framework should offer the possibility to export logging data with enough annotation to allow correlation with audio and video recordings. A way of bringing results back into the design stage could then close the test cycle and inform the start of another round of experiments.

System architecture

The proposed flexible use of LTCs puts certain requirements on the system architecture. Existing DM tools, for instance, use some sort of pipeline architecture in which the output of one component serves as input for the following component. A strict sequence of those inputs and outputs leaves little flexibility for human intervention. Specific WOZ tools on the other hand tend to collect all input in one place and leave it to the human wizard to deal with it. A structured process that makes use of LTCs is missing. Hence, in order to

combine the flexibility of WOZ and the technology support of DM a different architecture is needed.

In order to freely combine different LTCs we investigated the relationship between them as well as the role the human wizard plays upon them. Looking at different application scenarios for WOZ experiments we were able to abstract four different states a LTC can be in. That is, a specific LTC can be relevant for a given scenario (e.g. ASR in a hands-busy-eyes-busy situation) or irrelevant (e.g. MT in a monolingual setting). Consequently a LTC can be ON or OFF. In cases where a component is ON we can distinguish between three different states (see Figure 2). In situations where a used LTC satisfies the demands of an experimental setup the wizard does not need to alter the output (ON). In settings where one wants to simulate a better or different technology behaviour, the wizard’s task is to correct or change the output (COR). Finally, in cases where a specific LTC is relevant but not available it is up to the wizard to simulate the whole component (SIM). Looking at those four states we were able to deduct several rules that apply for consecutive components. That is, first a component that is ON can be followed by a component in any other state. Second, a simulated (SIM) component must be followed by a fully functional component (ON). In cases where two or more consecutive components are simulated (SIM) the simulation of all of them merges into one. The same applies for consecutive correction (COR) components. Finally, a component can only be in correction (COR) state, if the previous component is fully functional (ON). Using these rules it seems possible to define a system architecture that flexibly combines different LTCs.

	ASR	MT	TA	DM	NLG	MT	SS
Scenario 1	ON	SIM				OFF	ON
Scenario 2	ON	OFF	SIM		ON	OFF	ON
Scenario 3	COR		ON	ON	ON	ON	ON
Scenario 4	ON	COR	ON	ON	ON	SIM	

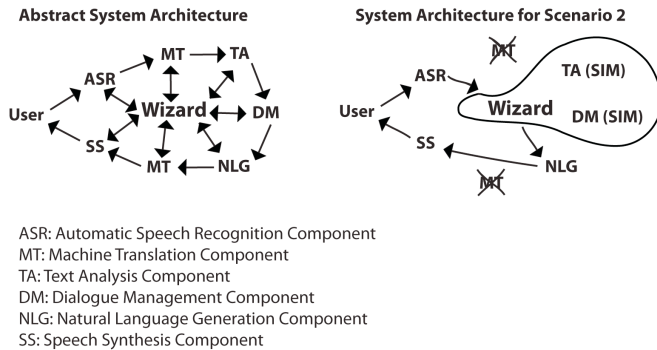


Figure 2. The relationship between LTCs: An LTC can be OFF (not relevant to a scenario) or ON (LTC output is used). Depending on the wizard’s task, an ON component can additionally be CORR (the wizard corrects its output) or SIM (the component is fully simulated by the wizard). Consecutive components are merged (grayed out).

Interface Layout

As a next step we wanted to gain a better understanding of the role of the wizard in order to identify specific requirements of a WOZ interface. A first prototype of a web-based

tool was built, using an experimental scenario that focused on measuring the impact of the quality of machine translation on a task-based dialogue. The tool was mainly supporting a pure WOZ methodology simulating ASR and NLU, but made use of a text based MT component. Lead by the task a preliminary dialogue model was designed and a wizard interface supporting this dialogue created. Three different members of our research team used the tool in an experimental setting with 12 different test participants.

Observations and retrospective analysis of those sessions generated insights with respect to the composition of a wizard interface, as well as illustrating many practical difficulties associated with being the wizard (interpretation of pauses by the user, deciding when to initiate a repair dialogue, etc.). Three factors influencing the interface layout were identified. The first factor is the scenario that needs to be supported. That is, in situations where there is a clearly defined dialogue structure the interface can act as a step-by-step tool. However, in settings where the interaction between a person and a system is rather loose and dynamic, a wizard simulating the system needs more flexibility. The second aspect we found influential for the interface layout is the role a wizard is playing. The wizard interface will appear differently when a wizard is simulating the behaviour of a LTC, compared to a situation where an actual LTC is in place and the wizard is only correcting its output. In the second case the interface needs to offer some kind of edit element whereas in the first case no interface element is needed at all. The third factor identified is the modality through which a test participant interacts with the system. For instance, a situation in which a participant would interact with a system by speech only is treated differently to a situation in which speech is coupled with one or more other input modalities. Thus, input modalities can be active, like text input or gestures, or passive as for instance location parameters. Both cases have a different effect on the wizard interface.

Common concepts across various scenarios

Guided by several scenarios we were able to define a common screen layout that is divided into different areas and consists of certain interface elements. It was found that there are consistent concepts that could be applied to various WOZ scenarios and influence the interface layout (see Figure 3). Firstly, a wizard interface needs to have a dialogue representation area that serves as the main interaction channel between the wizard and a test participant. Second, based on our experiment as well as the literature there is a need for a representation of the dialogue history. In other words, a wizard should be able to see what has happened previously in the dialogue. A third commonality between different WOZ experiments is that they all have a built-in way of dealing with errors. An interface area that is dedicated to dialogue repair strategies is commonly used. Finally, in cases where the dialogue progress is based some kind of slot filling mechanism, the status of those slots needs to be visible to the wizard. Using these concepts we were able to define an abstract generic interface structure that can be used in different WOZ scenarios.

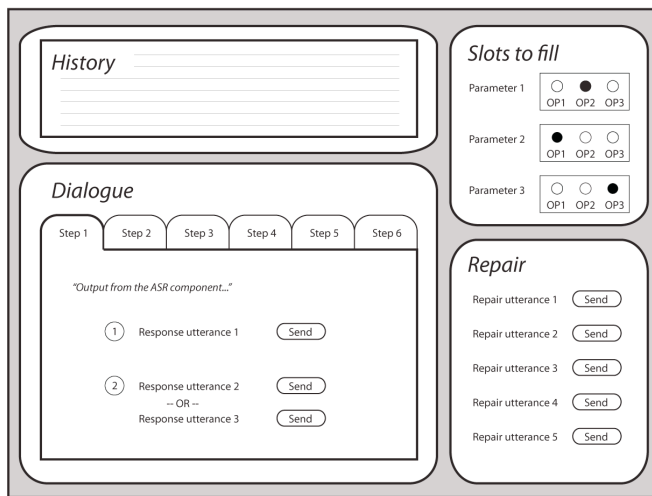


Figure 3. An abstract wizard interface with areas corresponding to consistent concepts that could be applied to various WOZ scenarios. These concepts are the dialogue representation, the dialogue history, repair strategies and status of slots to fill.

The role of the Wizard

After defining the abstract layout of an interface, it is then important to look at the LTCs that are actually used in a given scenario and what role the wizard plays upon them. In situations, for instance where ASR is used the wizard's task could consist of simulating (SIM) the recognition, correcting (COR) the output of a faulty recognition engine or simply acting upon the output of a working recognition engine (ON). In all cases the abstract layout of a wizard interface can be the same, the particular elements enclosed, however, might differ. That is, where in the case of simulation an 'output element' is necessary, in the case of correction some kind of 'edit element' is required and in the case where the wizard acts upon a recognition result a 'display element' is sufficient. Hence, based on the LTCs that apply in a given scenario and the role the wizard plays upon them it is possible to derive certain requirements for interface elements.

The input modalities

By looking at the different input modalities for a scenario, additional elements can be identified that might influence the interface layout. For example, if there should be a possibility for a test participant to interact with the system not only through speech but also via gestures, some way of displaying these gestures to the wizard must be found. Another example would be a scenario that requires providing the wizard with location parameters in cases where location based information is simulated.

Software Platform

When creating a new software framework, one also needs to think about the support of different platforms. Existing WOZ and DM tools mostly require a certain platform dependent configuration of the host system in order to run smoothly. Also they typically need an installation routine and a very specific experiment setup (i.e. several computers acting as clients and servers, multiple screens, cameras, microphones,

etc.). Recent hardware and software developments, however, already allow for a less intrusive way of testing interactions. Laptops with integrated cameras and microphones and network technologies that on the one hand support the reliable transmission of audio and video and on the other hand provide access to several LTCs, offer completely new possibilities. We propose to make use of those technological advances and create a mainly web-based framework. As such a generated wizard interface should run on an off-the-shelf computer without the need for special settings. The same should apply for the interface with which a test participant interacts. Also, since web application frameworks such as Ajax offer almost the same functionality as software that is locally installed [4], we can overcome lengthy installation routines and platform dependencies. Finally, in terms of language components the usage of 3rd party APIs and web services would offer great flexibility. In summary, by having a fully web-based implementation of a WOZ framework we would be able to offer new possibilities when it comes to running WOZ based user studies. That is, in theory it would not matter anymore whether a wizard is hidden next door or actually works from a different country since the framework providing the interfaces for the different parties and collecting the data would live online.

CONCLUSIONS AND FUTURE WORK

We have presented work in progress aiming for a WOZ framework supporting the use of language technology components. A novel system architecture allowing for a flexible integration of LTCs into WOZ experiments was proposed. Drawing from the literature and results of a first WOZ experiment, several requirements influencing the layout of a wizard interface were identified. Finally, we highlight the advantages of an internet based WOZ framework from a technological as well as a flexibility point of view. A prototype is being developed informed by the results of this first analysis. A system architecture that supports the flexible use of LTCs is being adopted and the layout for a generic wizard interface refined. Future work will also explore the implications of the use of web technologies for WOZ and evaluate the effectiveness of this approach.

ACKNOWLEDGEMENTS

This research is supported by the Science Foundation Ireland (Grant 07/CE/I1142) as part of the Centre for Next Generation Localisation (www.cngl.ie) at Trinity College Dublin.

REFERENCES

1. D. Bohus, A. Raux, T. K. Harris, M. Eskenazi, and A. I. Rudnicky. Olympus: an open-source framework for conversational spoken language interface research. In *NAACL-HLT: Proceedings of the Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technology*, pages 32–39, 2007.
2. J. Bradley, O. Mival, and D. Benyon. Wizard of oz experiments for companions. In *Proceedings of the British Computer Society Conference on HCI*, pages 313–317, 2009.

3. B. Buxton. *Sketching user experiences*. Morgan Kaufman, 2007.
4. C. L. Chen and T. V. Raman. AxsJAX: a talking translation bot using google im: bringing web-2.0 applications to life. In *Proceedings of W4A*, pages 54–56, 2008.
5. D. Coyle, G. Doherty, M. Matthews, and J. Sharry. Computers in talk-based mental health interventions. *Interacting with Computers*, 19:545–562, 2007.
6. N. Dahlbäck, A. Jönsson, and L. Ahrenberg. Wizard of Oz studies: why and how. In *Proceedings of IUI*, pages 193–200, 1993.
7. R. L. Erdmann and A. S. Nea. Laboratory vs. field experimentation in human factors an evaluation of an experimental self-service airline ticket vendor. *Human Factors*, 13:521–531, 1971.
8. C. Fügen, M. Kolss, D. Bernreuther, M. Paulik, S. Stücker, S. Vogel, and A. Waibel. Open domain speech recognition and translation: lecture notes & speeches. In *Proceedings of ICASSP*, 2006.
9. P. Geutner, F. Steffens, and D. Manstetten. Design of the VICO spoken dialogue system: Evaluation of user expectations by wizard-of-oz experiments. In *Proceedings of LREC*, 2002.
10. J. D. Gould and C. Lewis. Designing for usability: key principles and what designers think. *Communications of ACM*, 28(3):300–311, 1985.
11. A. C. Graesser, K. VanLehn, C. P. Rose, P. W. Jordan, and D. Harter. Intelligent tutoring systems with conversational dialogue. *AI Magazine*, 22(4):39–51, 2001.
12. D. Jurafsky and J. H. Martin. *Speech and Language Processing*. Prentice Hall, second edition, 2008.
13. A. Karpov, A. Ronzhin, and A. Leontyeva. A semi-automatic wizard of oz technique for Let's Fly spoken dialogue system. In *Lecture Notes in Computer Science: Text, Speech and Dialogue*, pages 585–592. Springer, 2008.
14. C. Keskin, K. Balci, O. Aran, B. Sankur, and L. Akarun. A multimodal 3D healthcare communication system. In *3D Conference*, 2007.
15. D. G. Kidd, D. M. Cades, D. J. Horvath, S. M. Jones, M. J. Pitone, and C. A. Monk. Listen up! Do voice recognition systems help drivers focus on the road? *User Experience*, 7(4):10–12, 2008.
16. S. R. Klemmer, A. K. Sinha, J. Chen, J. A. Landay, N. Aboobaker, and A. Wang. SUEDE: a wizard of oz prototyping tool for speech user interfaces. In *Proceedings of UIST*, pages 1–10, 2000.
17. L. Lamel, S. Rosset, J. L. Gauvain, and S. Bennacef. The LIMSI arise system for train travel information. In *Proceedings of ICASSP*, pages 501–504, 1999.
18. D. J. Litman and S. Pan. Predicting and adapting to poor speech recognition in a spoken dialogue system. In *Proceedings of AAAI*, pages 100–111, 2000.
19. Y. Liu and P. Fung. Multi-accent chinese speech recognition. In *Proceedings of INTERSPEECH*, pages 133–136, 2006.
20. K. Mäkelä, E.-P. Salonen, M. Turunen, J. Hakulinen, and R. Raisamo. Conducting a Wizard of Oz experiment on a ubiquitous computing system doorman. In *Proceedings of the International Workshop on Information Presentation and Natural Multimodal Dialogue*, pages 115–119, 2001.
21. M. Paul. Overview of the IWSLT 2009 Evaluation Campaign. In *Proceedings of IWSLT*, 2009.
22. M. Rajman, M. Ailomaa, A. Lisowska, M. Melchiar, and S. Armstrong. Extending the wizard of oz methodology for language-enabled multimodal systems. In *Proceedings of LREC*, pages 2531–2536, 2006.
23. A. Raux, D. Bohus, B. Langner, A. W. Black, and M. Eskenazi. Doing research on a deployed spoken dialogue system: One year of Lets Go! experience. In *Proceedings of INTERSPEECH*, pages 65–68, 2006.
24. D. Salber and J. Coutaz. A wizard of oz platform for the study of multimodal systems. In *Proceedings of CHI*, pages 95–96, 1993.
25. H. Somers. Language Engineering and the Pathway to Healthcare: A user-oriented view. In *NAACL Workshop on Medical Speech Translation*, pages 32–39, 2006.
26. S. Stücker, C. Fügen, F. Kraft, and M. Wölfel. The ISL 2007 english speech transcription system for european parliament speeches. In *Proceedings of INTERSPEECH*, 2007.
27. S. Sutton, R. Cole, J. de Vielliers, J. Schalkwyk, P. Vermeulen, M. Macon, Y. Yan, E. Kaiser, B. Rundle, K. Shobaki, P. Hosom, A. Kain, J. Wouters, D. Massaro, and M. Cohen. Universal speech tools: the CSLU toolkit, 1998.
28. M. Turunen and J. Hakulinen. Jaspis - a framework for multilingual adaptive speech applications. In *Proceedings of ICSLP*, pages 719–722, 2000.
29. M. Turunen, J. Hakulinen, E.-P. Salonen, A. Kainulainen, and L. Helin. Spoken and multimodal bus timetable systems: design, development and evaluation. In *Proceedings of SPECOM*, pages 389–392, 2005.
30. W. Wahlster. *Verbmobil: foundations of speech-to-speech translation*. Springer, 2000.
31. M. A. Walker, A. Rudnický, R. Prasad, J. Aberdeen, E. O. Bratt, J. Garofolo, H. Hastie, A. Le, B. Pellom, A. Potamianos, R. Passoneau, S. Roukos, G. Sanders, S. Seneff, and D. Stallard. DARPA communicator: cross-system results for the 2001 evaluation. In *Proceedings of ICSLP*, pages 269–272, 2002.