Perspectives on Ozlab in the cloud

A literature review of tools supporting Wizard-of-Oz experimentation, including an historical overview of 1971-2013 and notes on methodological issues and supporting generic tools

John Sören Pettersson and Malin Wik

Faculty of Arts and Social Sciences

Information Systems

Perspectives on Ozlab in the cloud

A literature review of tools supporting Wizard-of-Oz experimentation, including an historical overview of 1971-2013 and notes on methodological issues and supporting generic tools

John Sören Pettersson and Malin Wik

WORKING PAPER | August 2014 |

Perspectives on Ozlab in the cloud - A literature review of tools supporting Wizard-of-Oz experimentation, including an historical overview of 1971-2013 and notes on methodological issues and supporting generic tools

John Sören Pettersson and Malin Wik

WORKING PAPER | August 2014 | (2:nd print with minor corrections March 2015)

ISBN 978-91-7063-587-8

urn:nbn:se:kau:diva-33617

© The authors

Distribution:
Karlstad University
Faculty of Arts and Social Sciences
Information Systems
SE-651 88 Karlstad, Sweden
+46 54 700 10 00

Print: Universitetstryckeriet, Karlstad 2015

WWW.KAU.SE

Preface

The Wizard-of-Oz method has been around for decades, allowing researchers and practitioners to conduct prototyping without programming. The extensive literature review in the field that we have conducted revealed, however, that the re-usable tools supporting the method do not seem to last more than a few years. Generic systems start to appear from around the turn of the millennium, but already most have fallen out of use.

Our interest in doing this review was inspired by the ongoing re-development of our own Wizard-of-Oz tool, the Ozlab, into a system based on web technology. In this report we take stock of some key features of Ozlab as well as review and contrast other general Wizard-of-Oz tools. Our ambition has been to list every generic tool even if this entails some problems of defining exactly how generic a system has to be to qualify.

Nevertheless, we think this collection of systems and issues are of interest to people within the field, and we have added an introductory chapter which compares and contrasts prototyping in general with Wizard-of-Oz prototyping. This introductory chapter also provides an historical overview of Wizard of Oz in the development of digital interactive systems spanning the years 1971-2013.

A note on notation: we abbreviate Wizard of Oz into WOz but we have not standardised spelling in citations and titles. We hyphenate compounds as, e.g., 'Wizard-of-Oz prototyping' but leave citations and titles unaffected by this.

We thank Elisabeth Wennö for a language check and all the colleagues who have helped us find some papers or interpret their reports, while acknowledging that any language error, mis-representation, or obscurity remain our responsibility. We would welcome any comments on this report or suggestions for improving the information available on our website www.kau.se/en/ozlab!

August, 2014

John Sören Pettersson
john_soren.pettersson@kau.se

Malin Wik
malin.wik@kau.se

Table of Contents

1 Introduction

The Wizard-of-Oz (WOz) technique is a method used to simulate the inner workings of a system. The simulation is carried out by replacing a system's functionality with a human experimenter (a "wizard") who interprets the user's actions and mimics the functionality, with or without the user's knowledge. The simulation will thus appear as a real and functioning system for the user. These simulations can be employed to probe, discuss, demonstrate and evaluate ideas on how a device should respond to inputs (or actions) from users.

J.F. Kelley (1983) coined the "OZ paradigm" when reporting the development process of a natural-language computer application called CAL (Calendar Access Language), where a human replaced the language processing components in the first steps of the development. The "OZ paradigm" term alludes to the man hiding behind a screen while utilising some (simple) technology to impersonate the wizard in the novel *The Wonderful Wizard of Oz* (Baum 1900).[1] Gould, Conti, and Hovanyecz (1983) reported on a similar experimental arrangement as that used by Kelley. In an experiment meant to gauge users' tolerance of an imperfect listening typewriter, they mimicked automatic speech recognition with a human typist who wrote what the participants in the study dictated, but the system replaced words not in a predefined dictionary with *XXXX*'s. The edited writing was displayed on the user's computer monitor.

In research laboratories nowadays there are several systems supporting WOz experiments. One such system was developed at Karlstad University in the early 2000's. The system, called Ozlab, enabled prototyping, demonstrating, and testing graphical or multimedia interfaces, without any previous programming. However, this system depended on a multimedia

[1] In the book, the humbug wizard hides behind a screen, while in the 1939 film starring Judy Garland, there is a curtain. WOz papers sometimes refer to "the man behind the screen" or "the man behind the curtain".

production tool that no longer is supported by its manufacturer, and since 2012 Ozlab has been redeveloped as a web-based system. Our continuing redevelopment of Ozlab has prompted us to continuously follow the development of WOz systems and in the autumn of 2013 a systematic survey was made of the present state of wizardry around the world. Here we provide some details of Wizard-of-Oz prototyping methodology in general, of Ozlab methodology and functionality in particular as well as issues brought forth by other developers of 'generic' WOz systems, that is, setups where a human being plays an active role and which can be reused between different probings into interaction design problems.

This introductory chapter discusses some general issues for prototyping and a characterisation of WOz. It also provides an historical outline of interaction design work aided by the Wizard-of-Oz technique. Then follows a chapter on the operation of Ozlab, after which we present a literature review of generic WOz tools and then devote some space to problems around the use of such tools including problems for web-based WOz experiments. Although we found that it is beneficial for many 'everyday' employment of Ozlab to have its functions accessible as a cloud service, there are also drawbacks that need to be highlighted and documented. After the account of limitations inherent in a specific *technology* used for the implementation of a generic WOz tool, there is also a chapter discussing limitations in the Wizard-of-Oz *method* itself. Finally, the last chapter presents some concluding remarks beyond the original scope of this work (generic WOz systems).

1.1 Prototypes, prototyping, and the wizard's role

The popularity of the WOz technique in studying language technology and natural language interfaces can be explained by the nature of such systems and technology: "Automatic interpretation of text or speech is difficult and the Wizard-of-Oz technique thus gives systems developers a chance to test systems before it is even possible to make them." (Pettersson and Siponen 2002, p. 293) However, in the course of time, the Wizard-of-Oz technique has shown to be useful in other application areas as well.[2] "Since the system looks real to the test user, one could use Wizard-of-Oz mock-ups to test

[2] Examples of research areas where the Wizard-of-Oz technique has been used: Automatic Speech Recognition (ASR), i.e., Natural Language Processing (NLP), Text-to-speech synthesis (TTS), Multimodal Interaction, Human-Robot Interaction (HRI), Augmented Reality (AR), Artificial Intelligence (AI), Ubiquitous Computing (ubicomp), Mixed Reality (MR), Intelligent Tutoring Systems (ITS).

design ideas when there are reasons to believe that simple tests by sketches and slides […] will not provide the right responses." (Molin and Pettersson 2003, p. 77)

When using the Wizard-of-Oz technique, the user is deceived into believing that he/she is interacting with an automated system. Therefore the user's responses will be more like real computer use than responses in interaction with, for example, a paper prototype. Ozlab was developed at the turn of the millennium to do interaction experiments through the GUI, the Graphical User Interface, and not only through natural language. The GUI was by that time the standard for human-computer interfaces but it was harder to make WOz setups when graphics were involved – it is not sufficient to sit behind the curtain and see what people enter (or listen to what they say in a microphone) and then give the right commands to a system. Instead, user actions on a screen have to be followed and replacements for computer actions produced. There has to be a computer-support for the wizard that goes beyond the mere typing a command to a computer; this supporting tool has to be as graphic as the output itself. And at the same time the tool should not need to be programmed for each new test or demonstration. Otherwise not much is gained in employing a wizard to simulate the program of a system-to-be.

From the outset it was clear that making a graphical WOz system would not enable us to make simulations of action games – the wizard should otherwise have an infinite capacity to outperform the user. Thus, Ozlab was more for drag-and-drop interaction, as well as text input and (of course) speech input, clicking on objects and other 'pedestrian' interaction that a user might engage in, which constitute the interaction of most applications.

Thus, 'manuality' was (and is) an enabling and constraining characteristic of a WOz demonstration and experimentation; *enabling* as it releases a designer from the need to program, *constraining* as the demonstration will rely on the performance of the human wizard. The WOz technique is used when it is more important to demonstrate the user interface of some functionality than to demonstrate how to program the same functionality.

However, some automaticity may help a wizard to conduct the demo, simply because if there are some automatic parts, the wizard does not have to react all the time on what the other person is doing. Clicking on links would not need a wizard to shift pages if simple means for prototyping are used such as PowerPoint or HTML editors. Such interaction – navigation – is the typical interaction paradigm for much web use, and is also what paper prototyping is often used for: the test leader has to provide a sheet of paper each time a test user 'clicks' on a link by tapping his finger on the most recent sheet. Paper prototyping has certain drawbacks when test subjects do not feel free to act as they usually do because of the system's very obvious dependence on the test leader.[3] Prototyping by PowerPoint can on the other hand entail other problems because without a human intervention in the display of different pages, many pages have to be made in several variants for notifications, drop-down lists, etc. In one reported study, the experimenters started with five slides in the first version but finally ended up with a PowerPoint file consisting of 103 slides. "The more slides one is working with, the more a function is missed which would allow to act out changes on a defined number of certain slides." (Bönisch, Held & Krueger 2003, p. 1070).

In general, the production of a prototype to demonstrate or test a design idea should not cost much in proportion to the total cost of the development. The reason for this is simply that prototypes are made in order to revise or even discard the ideas they are built to demonstrate. Therefore interaction designers have proposed different kinds of simple prototyping methods such as those mentioned above. Often, this practice has been presented as an exploration of prototyping techniques along a fidelity axis: high-fidelity to low-fidelity. Before discussing various WOz studies and generic WOz tools it can be worthwhile to have a look at such dimensions of prototypes and prototyping. The reason is that it is not obvious how to categorize or classify a Wizard-of-Oz prototype.

Former Ozlab researchers Nilsson and Siponen (2006) presented a 3-dimensional characterisation of prototypes to better account for what is essential in WOz prototyping. Obviously, the WOz prototype differs from paper prototyping as it behaves like a programmed prototype or even as real system implementation. At the same time manual labour is needed. Someone has to do the processing, the 'thinking', when the user has acted in some way. Thus, the crux is the automaticity, or rather how it is perceived. When producing the prototype, it matters a great deal if the prototype is autonomous or not. For the user, this is of no concern as long as it appears to be automatic. Therefore, Nilsson and Siponen define two independent axes: *implemented* and *perceived automaticity*.

[3] Uceta, Dixon & Resnick 1998; Sefelin, Tscheligi & Giller 2003; Hundhausen et al. 2008; see also Lim et al. 2008 for other problems. Prototyping with children, esp. children with severe learning difficulties, is also a situation where paper prototyping has its limits; Molin and Pettersson 2003, p.77.

As for the concept of 'fidelity', Nilsson and Siponen note that "it is easy to judge a representation's fidelity if the representation has a counterpart in the physical world" (p. 9) but that this is precisely not the case when doing a design work. Instead they prefer the term 'precision' as do Beaudoin-Lafon and Mackay in their *HCI Handbook* article on prototyping tools and techniques: "*Precision* describes the level of detail at which the prototype is to be evaluated" (2003, p. 1007; update 2012). The latter authors use a four-dimensional framework for analysing prototypes:

- *Representation* [paper, computer simulated, etc.].
- *Precision* [e.g., informal or highly polished]
- *Interactivity* ["the extent to which the user can actually interact with the prototype": video clips are "fixed prototypes", while simple slide shows are "fixed-path prototypes"; programmed prototypes are "open prototypes" and for the present purpose it is worth noting that also WOz setups are "open prototypes"]
- *Evolution* [throw away, iterative, evolutionary; the latter prototypes eventually become part of the final system]

In contrast, Nilsson and Siponen after defining the two dimensions for automaticity, i.e. implemented automaticity and perceived automaticity, are satisfied with only one more dimension, i.e. precision. In WOz prototyping, the distinction between offline or online (as in the dimension Representation) is not the essential factor: it refers too much to the materiality of the prototype rather than to its appearance when actually used. This is not to neglect the discussion about the "Anatomy of Prototypes" by Lim, Stolterman, and Tenenberg (2008). As these authors point out, building prototypes in the HCI field has a peculiar characteristic compared to other design fields:

> "the material used in the field – digital material – is of a different kind, a 'material without qualities' (Löwgren and Stolterman 2004). As they can take almost any shape or form, digital materials have very few intrinsic 'material' limitations. Physical materials – such as wood, concrete, or steel – all have limitations and distinct properties that limit us in the choice of the desired form and function of a design. Working with the design of a digital artefact means that the material qualities determine form and function to a lesser degree, and that the design space therefore is larger and less restricted."
>
> (Lim et al. 2008, p. 9)

This fact is aptly demonstrated by prototyping with the Wizard-of-Oz method. A WOz prototype is not existent until it is used because the essential parts of the projected interactivity are not coded in the prototype. In Ozlab methodology we have always called the basic file/files for "interaction shell" rather than "prototype" – shells are empty – and added that an "interaction scheme" or "response scheme" is needed for the wizard. Pettersson (2003) points out that the script guiding the wizard's responses may be quite incomplete in the first design cycle. The scheme may then be elaborated. This explorative use of WOz is as important as the employment of WOz in evaluation. "People are generally good at interactivity, but not in programming it in advance." (Pettersson 2003, p. 163) Thus, in the *situated* context of a GUI dialogue, the wizard will not only notice when people have problems but also understand what might help them. This is to really use the human wizard as a *human*, not as a machines substitute.

Two major types of dimensions are identified by Lim and co-workers when they try to capture the anatomy of prototypes, namely filters and manifestations, because they see it as a fundamental principle that "*Prototyping is an activity with the purpose of creating a manifestation that* […] *filters the qualities in which designers are interested* […]" (ibid., Table I). For the manifestation aspect, they count three dimensions: *material* (medium used to form a prototype), *resolution* (level of detail of sophistication, incl. response time), and *scope* (range of what is covered to be manifested) (ibid., Table III). Regarding WOz, we can see that each dimension will contain several different variables, and thus the filtered quality dimensions, which are five in number, will at times be hard to judge even in one and the same interaction session. While the anatomy concept may be interesting to dwell on when analysing various setups, both before a round of interaction sessions as well as after, it will not be used in the present work. For generic WOz tools, production possibilities for wizards and what input a wizard can get from the system are more important (in addition, there is a general WOz methodological question of how much 'extra' information the wizard gets which a final system would not be able to capture – this question will recur here in various instances when development cycles are discussed).

To continue, the line of thought emphasized above is to use the human wizard as a *human*, not as a machines substitute. This line of thought – and actual experience – led Ozlab methodology early to recognize others than designers as wizards, and, vice versa, sometimes a programmer as nominal test subject in order to demonstrate for a non-professional designer wizard that users might act in various unforeseen ways, and there has to be specifications of system responses also for such (mis)use. This became evident in the very first development of Ozlab where special educators acted as designers and wizards for various interactive training materials (Pettersson 2002). The possible combinations of wizard types and user types have been further extended when it was recognized that hiding the wizard was often not needed; good GUI dialogues were obtained when content experts acted as test participants in order to validate content-correctness of interaction shells (Pettersson 2003, pp. 180f). Thus, the reliability and validity of WOz testing hinges as much on the purpose of each test as on any purported qualities of the prototype 'itself'. (Also Lim et al. take a critical stance towards the discussion on the validity of prototyping. Buxton's 2007 book on sketching demonstrates how prototyping can be a generative process.)

Finally, to comment on the fourth dimension in Beaudoin-Lafon's and Mackay's framework, i.e. *evolution*, it might seem that a WOz prototype always ends up at zero along this dimension, even if evolution in general is a relevant dimension of prototyping. However, it is important to put the WOz prototyping into the perspective of a longer design cycle (as will be discussed in particular in 3.3). Thus, it is again not the 'stored' version of a WOz prototype that should define its characterisation (other than for technical purposes, of course), but its employment. Furthermore, several WOz setups have been made to include functioning parts in order to gradually replace WOz simulations and thus arriving at working systems (possibly merely working prototypes, but anyhow, an evolution perspective can be relevant in WOz prototyping). Even if there is no fixed evolutionary goal, there is sometimes a need to mix functional and manual parts: Serrano and Nigay (2010, p. 218) point to the problem of evaluating multimodal[4] systems where different components have different levels of perfection. Such misalignments will probably make test participants prefer the modalities that work smoothly, thus making it impossible to evaluate full multimodal interaction. Therefore, wizardry is needed to simulate some of the components while other components are concurrently working automatically.

This brings us to another use of a WOz setup, namely one where the wizard is closely following the interaction of a test subject and a working system,

[4] See also their paper on multi-modality prototyping tools entitled "A three-dimensional characterization space of software components for rapidly developing multimodal interfaces." (Serrano, Juras & Nigay 2008)

and has the ability to intervene. At one extreme, this may be the sole purpose of letting a test leader into the working system:

> "The trainer control unit is implemented as a touch-based interface serving as a teaching tool that allows instructors to animate game characters in response to trainees' actions.
>
> All the animations and actions that the game can perform could be initiated through this trainer control interface. Moreover, game actions that are initiated by trainees through spoken commands can be overridden by instructors (e.g. when the speech recognition system did not accurately process the command) or reversed by instructors to create "on-the-fly" training situations that test the responsiveness and judgment of the trainees, as, for example, introducing non-compliant behavior for the virtual game characters."
>
> (Fournier et al. 2012, p. 6)

In this example the purpose of the wizardry is not to redesign the system but to act as a teacher (or game master) and this falls somewhat outside the "OZ paradigm" as it were. But it illustrates the width of application along an evolution dimension. In fact, some years ago the Ozlab group in Karlstad discussed with a rehabilitation centre that suggested using Ozlab for scaffolding when people with acquired brain injury was training to come back to computer literacy including mastery of widgets such as dropdown lists.

After this introduction to WOz as one prototyping method among others, and to the various aspects of WOz tools and wizardry, the following section gives an historical exposé over how Wizard-of-Oz tests have been conducted.

Many studies have used the "OZ paradigm" since Kelley coined the term in the beginning of the 80s. The aim of this literature review is mainly to give examples from the various application areas rather than listing all papers etc. reporting Wizard-of-Oz studies. Moreover, in the domain of Natural Language Processing (NLP), many systems are commercial and therefore many studies and methodological refinements are not reported in scholarly or other publicly available publications, which makes it hard to provide a detailed account.

In addition to the exposé in section 1.2 of WOz studies in different application areas, Chapter 3 aims at providing a complete list of WOz setups that are made for re-use as general experimental tools rather than for a specific test or series of tests. They are here called "generic WOz tools".

Naturally, any definitive boundary between generic and non-generic systems is not possible to establish, especially regarding the many and divergent application areas as explicated in section 1.2. What one team finds 'generic' might well be regarded as specialized or limited by other teams. But because quite many systems have been developed with reusability in mind the intention of the present work has been to gather as many such examples as possible in order to see general trends and problems (to be discussed in Chapter 3 and onwards).

1.2 Examples of WOz experimentation 1971-2013

In this section, WOz studies with non-generic tools are presented. The order of presentation is chronological rather than by application area.

Wizard-of-Oz-like experiments were not common before the '80s. There is of course the fantastic story of von Kempelen's Mechanical Turk from the 18th century (see, e.g., Standage 2002), but the present historical account is confined to digital interaction automata. A rare pre-80s example in the IT field is the evaluation of a self-service airline ticket vending machine, where part of the internal functionality – communication with the airline booking system – was performed by a human operator instead of the Automatic Ticket Vendor machine itself (Erdmann & Neal 1971). Another example is found in Malhotra's research on how to make management systems more accessible by letting managers use English rather than formal query languages. Malhotra let test participants "solve a realistic problem using a simulated 'perfect' English language management-support system" (1975, p. 56) – perfect in the sense that it was not reliant on any immature English language processing capability of a computer. Participants typed in their request on a hard copy console but the requests were interpreted by the experimenter, who also composed answers with the help of a database and his own knowledge. Malhotra reports:

> "In fact, surprising as it may seem, few subjects realized that the experimenter was creating the responses until they were told so after the experiment. Until this secret was revealed, many subjects were extremely impressed by the range of capabilities displayed by the system. Thus, the Perfect System could be said to be a success as the subjects behaved as if it were an ideal English language question-answering system."
>
> (Malhotra 1975, p. 57)

Thus, this was a success as a method of generating typical systems request made by managers if they could use English without having to learn a formal language. "Their English was informal, much closer to the spoken language than prose" (p. 61). On the other hand, would it not be more interesting to see how human-machine dialogue would *differ* from human-human interaction since no Perfect English Language System is likely to be built? A caveat is given by Tennant: "Data can be gathered from simulations that cannot be gathered from interaction with actual systems simply because people can so readily adjust their habits." (1981a, p. 37; not to be confused with his 1981b book on NLP systems). Tennant develops his own evaluation methodology where the human intermediary is used to generate a standard or background against which to judge a specific system:

> "Completeness—simulate the system with a human intermediary and real world problems to study user expectations toward a linguistically capable system; compare results with the capabilities of the system under test."
>
> (Tennant 1981a, p. 54)

However, by this time researchers had been accustomed to functioning systems for typed and spoken natural language processing, albeit limited in capacity, and interests were geared towards making usable systems rather than wholly natural systems (for the continued debate on this, cf. e.g. Edlund et al. 2008). Thus, in the '80s the deceitful Wizard-of-Oz method started to flourish as a method to test systems not yet implemented.

As mentioned at the beginning of this chapter, Gould, Conti, and Hovanyecz (1983) used WOz when developing a listening typewriter. The experiment was meant to gauge users' tolerance of an *imperfect* listening typewriter. During the experiments the subjects talked into a microphone in front of a computer screen. In an adjacent room, a typist acted wizard and took down what the test subject said if the words were in a predefined dictionary. The wizard's text output was displayed on the monitor in front of the test subject.

Kelley (1983) coined the "OZ paradigm" when employing Gould's and co-workers' methodology for the development of natural-language computer applications. The "OZ paradigm" was used to simulate the language processing components of CAL, the Calendar Access Language, in two ways, as described by Kelley (1984, p. 28; cf. 1983, p. 193):

> "*First run of OZ* (*simulation*). Here, *no* language processing components were in place. The experimenter simulated the system *in toto*."
>
> "*Second run of OZ* (*intervention*). This was the iterative design phase of program development. Fifteen participants used the program, and the experimenter intervened as necessary to keep

> the dialog flowing. As this step progressed, and as the dictionaries and functions were augmented, the experimenter was phased out of the communications loop." [In this run, participants used keyboards.]

The second run yielded fewer and fewer new words for each new participant, and it was succeeded by a *validation* step where a further six participants tested the resulting program to see how it performed.

Notably, Kelley used the word "iterative design" for *one* single phase where improvements to program code were made for each participant trying to complete various task with the calendar application. The word "iterative design" is often used for each major round rather than for each test session by many other authors. Kelley's use of the word reveals how the Wizard-of-Oz technique can be applied, that is, to *develop* an interaction design rather than *validating*, *evaluating*, or in general *testing* it. On the other hand, in a simulation setup it is cheap to compare different designs because they are not even partly implemented yet.

After the works of Gould, Kelley, and other researchers at IBM Watson Research Center, several other Wizard-of-Oz setups were employed by researchers and developers within the area of NLP, natural-language processing. By the 90s it was employed in several commercial developments; suffices it here to note one paper which at some length discussed the method of using hidden wizards including the criticism which by this time had been voiced against this way of conducting experiments.

The simulation environment ARNE-3 is seen here as non-generic even though it was used in different studies by Dahlbäck, Jönsson, and Ahrenberg (1993). The authors state that customizing the environment is time-consuming, and needs to be done prior to a new study. When working with ARNE-3, the wizard interprets the participant's commands and chooses what to display from a database connected to a graphical interface. The system contained a menu-driven sentence generator, preventing the wizards from making spelling mistakes and grammatical errors.

Interestingly, and typical of the time when they conducted their experiments, Dahlbäck and co-workers addressed two questions outside the immediate concern of their experiments, namely "Does the method work?" and "For and against chosen method". These were big questions by that time. Are test participants really fooled by the setup so that data can be regarded as valid (i.e. being data from a human-computer interaction and not between two humans)? The authors answer this question in the affirmative. The ethical issues of fooling people are countered with the explanation that people are informed afterwards and that in the reported experiments, no test subject became angry when hearing the truth.

Some more factors were considered in the weighing of arguments for and against the Wizard of Oz. Some critics had advocated the study of real systems instead of simulated ones – a proposal obviously not from the design disciplines. Even so, Dahlbäck and his co-authors, referring to other NLP researchers, conclude “that people can often adapt to the limitations of an existing system, and such an experiment does not therefore tell you what they ideally would need.” (p. 265). Here they could also have referred to Leiser (1989), who demonstrated how participants adapted their queries to paraphrases made by the system before presenting search results (as NLP was immature when Leiser conducted the experiment, the study relied on the Wizard-of-Oz technique).

Dahlbäck, Jönsson and Ahrenberg, finally addressed the criticism often directed against laboratory-based studies, namely that the setups entail rather artificial situations and role playing by the participants (that is, not only the more obvious role-playing by the wizard): “However, if the focus is on aspects not under voluntary conscious control”, as the case is with linguistic elements, data are likely to be valid, the authors conclude. Some of these issues will be discussed in more detail in Chapter 5.[5]

Maulsby, Greenberg and Mander (1993) used the Wizard-of-Oz technique in order to explore how users would teach an “intelligent agent” Turvy to do often repeated tasks in the UI. Relying on previous studies such as Gould and co-workers, Leiser referred to above, Hauptmann (1989) on manipulations by gestures and speech, and the ARNE study, as well as on their own user study on a programming-by-demonstration system, they had come to the conclusion that:

> “Agents must be designed around our understanding of what people require and expect of them. However, the traditional approach of system building is an expensive and unlikely way to gain this understanding. The underlying discourse models and algorithms for agents are usually so complex and entrenched with assumptions that changes—even minor

[5] With all the variability potentially introduced by a human being, could a WOz experiment deliver reliable data as compared to other experiments in psychology? This is not raised by the ARNE group – to the contrary, they claim that their use of several wizards ensure that participants’ sentence constructions are “not the reflection of the idiosyncrasies of one single person’s behaviour” (p.265).

> ones—may require radical redesign. Moreover, because agents act as intermediaries between people and their applications, the designer must craft and debug the agent/application interface as well. A viable alternative to system building is Wizard of Oz."
>
> (Maulsby, Greenberg & Mander 1993, p. 277)

By using WOz, the experimenters could see what object selections, demonstrations, and verbal input users would use, and also how users structured Turvy's learning. As for the method itself, the experimenters drew five conclusions (ibid., Abstract and p. 283):

a) "Design of the simulation benefits greatly from prior implementation experience." The essence of this argument is that it is important to be close to what systems could realistically perform. Otherwise, the WOz data generated will be invalid.

b) "The agent's behavior and dialog capabilities must be based on formal models." The reason for this is the same as a) and also because it "ensures consistent behavior and experimental repeatability." This is thus in sharp contrast to the "iterative" method employed by Kelley for quickly working towards a usable as well as machine-recognisable set of words for CAL. One might say that the latter experiment was more complicated as it involved example-giving by the users and not only spoken commands ("Turvy is the most complex Wizard of Oz simulation done to date.") and this fact would make it necessary to limit the degrees of freedom along all the involved dimensions.

c) "Studies of verbal discourse lead directly to an implementable system." Maulsby and his co-authors refer to other systems implanted by the group. One could also compare with other systems developments where some freer linguistic interactions have preceded more restricted interaction (again Kelley can serve as an example, but one can also compare with the Iterative Communication Capacity Tapering, ICCT, by Mavrikis and Gutierrez-Santos mentioned below).

d) "The designer benefits greatly by becoming the Wizard." An interesting remark, of course, and its implied scope can be extended. When non-professional designers are involved, experienced programmers and/or designers can act as test participants (beside the target group) to demonstrate specific 'aberrant' user behaviours that one has to design computer responses to, as noted above in section 1.1.

e) "Qualitative results are the most valuable. By acting as Wizard, facilitator, and interviewer, the experimenters become immersed in the experiment and many important results become obvious. The most interesting experimental questions cannot be answered by statistics, at least in small, cheap studies. Still, measurements are useful: they validate the opinions of experimenters and users, and allows a detailed (if myopic) exploration of particular activities." (p. 283)

It could be added that the test participants in this study were aware of the 'man behind the curtain'. "To reinforce the fantasy, the Wizard spoke in clipped sentences, with rather mechanical intonation. While we did not deceive users, they quickly bought into the illusion. They spoke more curtly to Turvy than to the facilitator, and referred to Turvy and the Wizard as two separate entities." (p. 281) In addition, the Turvy wizard not only heard the user's voice but could also see his or her gestures and respond consistently.

The book *Humans, Computers, and Wizards. Analysing human (simulated) computer interaction* from 1997 by Wooffitt, Fraser, Gilbert, and McGlashan argues that an HCI approach based on cognitive psychology does not encompass what is going on in an interaction: "Dialogue emerges through the interplay of the participants. Each participant only has the ability to influence the directions of the dialogue on a turn-by-turn basis. Neither can plan a course for the whole dialogue *a priori* and impose it on the other." (p. 13) The authors refer to sociology, conversation analysis, and Suchman (1987), but already when it was published this book must have been a bit antiquated, not only because of its focus on conversation as telephone dialogues, but also because design science was discussed in HCI and the scope was seldom narrow linguistics and cognitive psychology. (Some reviews of this book will be discussed in 6.1 below.) A year later, the volume *Designing Interactive Speech Systems. From First Ideas to User Testing*, by Bernsen, Dybkjaer, and Dybkjaer (1998) appeared, in which the focus is once again on speech-based systems. "Wizard of Oz Simulation" is given a 34-page chapter even though the authors embrace the method with caution: "WOZ is a relatively costly development method because: (1) the wizard needs a significant amount of training and support; (2) involving experimental subjects, WOZ experiments require careful planning and preparation and take time to run; (3) experimental results have to be transcribed and analysed, which take time and requires skill to benefit further system development." (p. 127) The first point is especially pertinent in NLP simulations: "it is difficult or impossible for the wizard to precisely simulate the limited speech recognition of the future system" (p. 130).

Other interaction researchers have taken more advantage of the fact that the Wizard-of-Oz method could be used for experimenting with other natural expressions than language as input to a system. Hauptmann (1989), referred to by Maulsby as noted above, used a person for the recognition of gestures and speech when test participants tried to manipulate 3D-graphic images. Haputmann did not use the term "wizard" for this person and there is nothing in his paper that indicates that participants were deceived. At any rate, these were gestures directed at the computer but one can also imagine systems recognising other actions taken by humans. NEIMO, which is listed among the generic systems in Chapter 3, was built to collect data from several observers/wizards to capture not only keystroke level events but also "high level tasks such as sending a fax" (Coutaz, Salber, Carraux & Portolan 1996, p. 402). Thus, the idea of an ambient computing environment could be explored. The system was designed to analyse multimodal inputs such as combined use of several modalities as well as redundancy "i.e., simultaneous use of multiple modalities with identical semantic content as in uttering 'Call Jo Smith' while clicking on Jo's direct phone number" (ibid.; for further developments by the groups in Grenoble, see MultiCom and OpenWizard in Chapter 3 including the footnote about EmOz).

After NEIMO, several WOz studies have been made in the area of Ubiquitous Computing. For instance, Mäkelä, Salonen, Turunen, Hakulinen, and Raisamo (2001), using a tool similar to the one used by Dahlbäck and co-workers referred above, conducted a WOz experiment on a computerized Doorman system in its intended environment at a university in Finland. The system's speech recognition was simulated during the experiment. This study also exemplifies the growing interest in including gesture *output* from the system: the wizard controlled speech synthesis and direction pointing of a robot inside the building. After the turn of the millennium, there have been an ever-growing number of HRI studies, i.e. studies in Human-Robot Interaction. While the Ozlab-based study on an orthopaedic robot was quite advanced (section 2.1.1; Larsson & Molin 2006), the most challenging question is how people interact with movable robots (see e.g. works on service robots by Green et al. 2004, 2006, and Hüttenrauch et al. 2006). Such studies have expanded the Wizard-of-Oz methodology as will be further highlighted in section 1.2.1 on Riek's (2012) review of 54 WOz-based HRI studies.

Hudson, Fogarty, Atkeson, Avrahami, Forlizzi, Kiesler, Lee, and Yang (2003, p. 257) explored "[…] whether and how, robust sensor-based predictions of interruptibility might be constructed, which sensors might be most useful to such predictions, and how simple such sensors might be". Hudson et al. claim that the Wizard-of-Oz technique was used to simulate "a range of possible sensors through human coding of audio and video recordings." (p. 257) One could argue that such sensors were simulated by a human. But even so, this study does not seem to incorporate what is normally described as the Wizard-of-Oz technique, as no human wizard is acting/giving output to the user in real-time: the audio prompts sent to the participants in the study, asking them to rate their current "interruptibility" were sent at "random but controlled intervals" (p. 259), indicating no human/wizard involvement in the audio prompts.

Probably the most extreme 'manual' intervention by wizards is reported by White and Lutters (2003). They used the Wizard-of-Oz technique to assess concepts, identify design requirements, and understand organizational forces in a field study of "cross-organizational expertise recommendation and organizational memory systems (ER-OMS)" (p. 129). They argue that it is hard to make a proof-of-concept requirements gathering with other prototyping techniques because content and (other) users are missing. The users' questions to the simulated inter-organizational system were physically carried between the three partaking sites, where the two other sites printed solutions from their databases if available. Solutions were then physically carried back to the person posing the question. Email or fax was not used, as that would have "limited our ability to observe processes of sensemaking over the returned solutions" (p. 132, right column).

Another area that was gaining ground in the 1990s was avatars or artificial faces embodying information in conjunction with text or speech output. The EU project Humaine exemplified this with its focus on "Embodied Conversational Agents (ECAs)". Wizard of Oz was used to drive conversations with test participants while there was a set of selectable embodiments for the wizard to make use of. See especially the project workshop document by Cavalluzzi, Clarizio, De Carolis, and de Rosis 2005 (for interaction data and conclusions, see Berry, Butler & de Rosis 2005; de Rosis, Cavalluzzi, Mazzotta & Novielli 2005; Cavalluzzi, de Rosis, Mazzotta & Novielli 2005).

Höysniemi, Hämäläinen, and Turkki (2004) used WOz to simulate a body movement controlled computer game for children by letting a wizard interpret the children's body movements when playing the game and controlling the avatar via a computer.

Akers (2006, p. 454) used the Wizard-of-Oz technique in a "participatory design process in which users invent and test their own gestural selection interfaces" helping in the development of an interface for 3D selection of "neural pathways estimated from MRI imaging of human brains". The users (neuroscientists and one radiologist) explained the intentions with their gestures and the wizard implemented (simulated) them.

Another attempt to provide for end-user design development but within the area of sensor-based application is iCAP. “End-users with little technical expertise should be able to exercise control over context-aware systems and rapidly prototype applications. They have more intimate knowledge about their activities and environments than a hired programmer and they need the ability to create and modify applications as those activities and environments change.” (Dey, Sohn, Streng & Kodama 2006, p. 255) The system iCAP tried to capture this by providing simple and graphically displayed rule settings (Sohn and Dey 2003. iCAP used SATIN by Hong and Landay 2000; see also other works by Landay). WOz was used to confirm test users ability to set rules: “Finally, users were able to successfully test their rules using the Wizard-of-Oz prototyping interface. For each rule, they verified that the correct action was executed when they manually set the appropriate contextual conditions.” (Dey et al. 2006, p. 266)

Consolvo, Harrisson, Smith, Chen, Everitt, Froehlich, and Landay (2007) report on a study where the collection of in-situ data, especially for ubicomp products, was evaluated. Three techniques were used, among them Wizard of Oz. The WOz prototype was used to simulate sensors picking up on the elder’s activities and doings, “deployed in home settings for supporting eldercare” (p. 104). The prototype was that of “CareNet Display”. Phoning the elder and/or the caregiver several times per day simulated the intended sensors. The output of the simulated sensors was then manually updated to be visible on the “CareNet Display” via “web connection”, without needing to involve family members of the elders.

Also NLP researchers were interested of “in situ” data in spite of Dahlbäcks et al.’s optimistic view that in NLP research “the focus is on aspects not under voluntary contrary conscious control” (loc. cit.). At the Finish-Swedish telephone operator Telia Sonera research had been ongoing for more than a decade when it finally aborted all speech research in 2009. By integrating research into customer companies call center services, natural data could be collected, even if the speech output was controlled by wizards. Eklund (2010) makes the case for the benefits when doing research on so-called disfluencies, especially “filled pauses”, i.e. when people say “er”, “ah”, etc., which of course can interfere with automatic speech recognition. In fact, the Telia researcher found filled pauses to be twice as common in their data as in lab-based studies. Eklund hold this against laboratory studies: “As have been shown, disfluency is within speaker control (e.g. [Siegel et al. 1969]) and it could be the case that awareness of the recording devices actually have an effect on disfluency production. For example, it has been shown that speaker disfluency is decreased simply by directing a TV camera on the speaker [Broen & Siegel 1972]." (ibid., p. 25f) But he acknowledges that studies based on uninformed subjects are unlikely to pass ethical committees.

Another natural language phenomenon, not often prominent in dialogue systems development, is the fact that two speakers often predict when the interlocutor will go silent and therefore start speaking almost at the instance when the interlocutor ends an utterance. "Contrary to this, most spoken dialogue systems use a silence threshold to determine when the user has stopped speaking. The user utterance is then processed by one module at a time, after which a complete system utterance is produced and realised by a speech synthesizer" according to Skantze and Hjalmarsson (2010). Instead they used an "incremental" speech analyser to produce utterance quicker. As the automatic speech recognition module was not yet complete, they had to use the Wizard-of-Oz method to test the impact of repair sequences and pause fillers for the overall user experience. This setup made quite demanding requirements on the wizard to quickly write down what test users said. At the same time the setup included just the methods humans would use (fillers and repairs). A comparable setup without incremental analysis was also used and test users found the incremental method to be significantly more polite and efficient, and it was easier to understand when to speak. Interestingly, the different system behaviours did not result in any statistically significant differences in user behaviour (user utterance length and user response time were analysed).

Some years before, Robins, Dautenhahn, te Boekhorst, and Nehaniv (2008) reported on a wizard facilitated experiment on child-robot interaction: Would delays in robot response affect children's turn taking and pace of performing gesture input? When children were beating a rhythm on a tambourine, which the robot repeated, "the effect of delay was especially strong when the robot did not show a facial/gestural expression" (p. 21). "Delay by the robot also increased the average drumming duration in the children, but in this case the effect was significant only when the robot did exhibit facial/gestural expression" (p. 22). On the other hand, when the children were to elicit some gestures/postures for the robot to imitate, delayed response had the opposite effect on some children: "To the experimenter, it appeared almost as if they 'couldn't wait for their turn'" (p. 22). Despite variations, a "statistical analysis of the whole sample" showed a prolongation in the children's performance of the gestures/postures that the robot should imitate (p. 23).

Studies such as those by Skantze and Hjalmarsson, and Robins and his co-workers make it harder to generalize the results by Leiser (1989), who demonstrated how participants adapted their queries to paraphrases made by the system as mentioned above. Moreover, for an experimental technique that relies on an extra loop before responses can be provided to test subjects, as the Wizard-of-Oz method does, the effects of delays must be taken in account. This will be discussed more thoroughly in Chapter 6.

Lee, Mott, and Lester (2010) simulated "natural language dialogue functionalities" by using the WOz technique, when experimenting with interactive narrative decision-making in a learning environment (which were narrative-centered) for middle school students. The wizard provided the user with narrative planning functionalities, guided the user through the game via hints and decided what the user should/could do next (through spoken natural language dialogue, but also through game specific hints/guide/plot changes). Less narrative focused but with an interesting pedagogical approach where the system asks the science student was elaborated in part by WOz experimentations by Ward et al. (2011).

Webb, Benyon, Bradley, Hansen, and Mival (2010) used the Wizard-of-Oz technique to collect dialogue data fit for *companions*: "Companions are advanced spoken dialogue systems, that attempt to go beyond the limited functionality of current task-oriented systems, to be cooperative, collaborative dialogue partners, that form long term relationships with the user." (Webb et al. 2010, p. 875) In their experiments text-based user utterances were used to classify the user's mood. The wizard had "a strict series of guidelines to control the interaction to identify and/or react to certain user driven situations". (ibid. p. 875) The classification was done by the wizard who calculated the cumulative score of the user's utterance, and then answered according to what system strategy was decided for that specific session. In their experiments, Webb and co-workers used two interaction strategies that they called "empathy" and "positivity". (For more on the Companion project 2007-2010, see http://www.companions-project.org.) Another attempt to base analysis on more than linguistic expressions is presented by Rösner, Frommer, Friesen, Haase, Lange, and Otto (2012) when building up the corpus LAST MINUTE: skin reductance, heartbeat, and respiration as well as stereo camera data are available to researcher (see instructions in Rösner et al. 2012, p. 2566). Also extending beyond the NLP domain, Li (2012), after a cursory literature review, tries to make the case that more stringent WOz experimentation is needed for the proper analysis of proposed interaction design for so-called intelligent systems, i.e. pro-active systems. Through a series of WOz experiments Li demonstrates the impact on the consistency of system operations of four factors: interaction schema, wizard user interface, wizard's interpretation of participant's actions, and change of wizards.

A totally different approach to multi-modality in WOz studies, but also with a very strong methodological focus, is taken by Mavrikis and Gutierrez-Santos (2010). They present a study on the development of pedagogical software where the concept of Iterative Communication Capacity Tapering (ICCT) was used. The word *tapering* means that communication between learner and teacher is gradually narrowed down in a series of iteration when developing a tutoring system (by WOz, of course, otherwise there would not be any interaction involving the teacher). ICCT is a combination of tapering along two dimensions, namely what the authors call "interaction bandwidth" and "feedback improvisation":

The *interaction bandwidth* captures "the available modalities of communication between the student and the facilitator" (p. 643) the authors explain and provide an example: "In face-to-face communication, there are many different ways to communicate with the student. The facilitator can speak orally, but can also point to entities on the screen, take control of the actions (e.g. moving the mouse for the student) to prove an argument, and draw inferences based on facial expressions and gaze direction (e.g. focus of attention, emotions like boredom or excitement, etc). This rich communication is far from what can be achieved by most computer-based system." (p. 644)

Feedback improvisation captures "the freedom provided to the facilitator to improvise during a session with the student. This becomes important in the case of exploratory environments, in which the student holds a greater freedom to act than in other systems, while the computer-based system holds the usual limitations in the kind of feedback it can provide. [¶] This issue is especially important in the case of textual communication; the most common approach to provide support. Despite the great advances in the NLP field, natural language generation is still far from being a mature technology that can be used easily for the provision of intelligent support. Therefore, most systems rely on a template of pre-generated messages for their interventions. The design of these templates is crucial for the correct deployment of effective intelligent support and plays a central role in our methodology" (p. 644)

In conclusion, Mavrikis and Gutierrez-Santos suggest that the amount of side-channel information between the test participant (a learner) and the wizard (teacher), and the wizard's freedom to improvise how guidance is given to the test subject should be gradually narrowed as the iterations of the development process replace one another. "After each cycle, the communication capacity of the situation is reduced. This spiral process brings obvious similarities to the spiral model of software design (Boehm, 1986). However, it is important to note that our spiral moves inwards, not outwards. In the traditional spiral software design, a bigger radius after each iteration represents more or better functionalities. In contrast, the shrinking radius of our spiral represents a reduction of the communication capacity." (2010, p. 643)

In this survey of WOz experimentation it is worth noting some recent reports which makes the increased efforts on movable support clear. Spindler, Weber, Prescher, Miao, Weber and Ioannidis (2012) conducted a WOz evaluation of an indoor exploration and way-finding smartphone application for blind and low-vision people. The pilot test took place in Frankfurt Airport, where six blind test subjects were asked to walk from an adjacent railway station to a terminal in the airport. WOz was used to manually trigger spatial descriptions and directions through Text To Speech (TTS), transmitted to the test subjects' Bluetooth headset.

Poschmann, Donner, Bahrmann, Rudolph, Fonfara, Hellbach and Böhme (2012) simulated the speech recognition in a tour-guide robot in its intended environment by using the Wizard-of-Oz technique. The wizard interacted with the visitors of the museum through the tour-guide robot by choosing answers from a set of predefined phrases. "In order to allow the operator to [...] react to the visitors, the images from the omnidirectional camera as well as an audio stream were transferred to the operator's laptop." (p. 703)

On-going experiments in Grenoble use small robot movements and non-lexical sounds as "socio-affective glue" to make elderly people accept to control a "smart home" by speech directed to a little service robot (Aubergé et al. 2014).

Thus, HRI continues to develop with more mobile robots being tested in-situ outside university or industry laboratories, and the WOz technique often plays a role in such tests; section 1.2.1 rounds off the HRI discussion for the present work by referring an HRI WOz literature review. In general, with the dramatic increase of interactive handsets and ubicomp arrangements, the mobility of the interaction is notable. As will be seen in Chapter 3 where generic WOz tools (more or less prototypical, but used in real research or development projects) are reviewed, the number of WOz tools for mobile applications are increasing and was in 2013 the obvious goal for many such tools. There are also several projects where the WOz part is really just a substitute – as in Mavrikis and Gutierrez-Santos model above – which is weeded out to be replaced with working software as the development projects go on. This is interesting as it revives the original "OZ paradigm", which was not used to simply generate scientific facts about man-machine dialogue under varying conditions, but to produce a specific system, in Kelly's case a calendar system for office use. However, Chapter 3 notes some problems for the longevity of WOz setups when these are intended to be more generally applicable than the single one motivating the setup.

1.2.1 Riek's proposal for reporting HRI WOz studies

After reviewing 54 papers on human-robot interaction where WOz techniques were applied, Riek proposes as set of "reporting guidelines for HRI studies that use WOz" (Table 2 in Riek 2012, p. 130). These guidelines are provided as a set of questions for each "Experimental Component":

Robot

How many robots were used?

What kind(s) of robot(s)? (e.g., humanoid, zoomorphic, mechanical, android?)

What level(s) of autonomy? (i.e., which components of the robot(s) were autonomous and which were controlled by the wizard?)

What were the robot's capabilities?

What hypotheses did the researcher have for the robot?

User

How many users participated in total, and per experimental trial?

What were the user demographics, sampling procedure, etc.?

What instructions were provided to the user?

What behavioural hypothesis does the researcher have about the user?

Was the simulation convincing to the user?

What expectations did the user have about the robot, before and after the experiment?

Wizard

How many wizards were used?

What were the wizard demographics? (e.g., the researcher, lab mates, naïve?)

Did the wizard know the behavioral hypothesis of the experiment?

What were the wizard production variables and how were they controlled for?

What were the wizard recognition variables and how were they controlled for?

How did the experimenter control for wizard error (deliberate and accidental)?

How much and what sort of training did wizards receive prior to starting the experiment?

General

Where did the experiment take place?

What were the environmental constants and how were they controlled?

What scenarios did the researchers employ?

Was this experiment part of an iterative design process?

Does the paper discuss the limitations of WOz?

Obviously, not all are applicable to all WOz studies outside the robotic field. Noteworthy for development work directed to the WOz system itself is Riek's question about how many wizards participated – when developing a re-usable WOz tool the persons acting as wizards are the test subjects (compare Li's, 2012, analyses of how using several wizards can impact consistency of system output).

For our own discussion we find the following questions interesting even if we do not explicitly address these HRI aspects in the present work:

- In WOz HRI studies, what are the key characteristics? Test Participant input (e.g. speech or remote control by mobile phone), input to wizard (location, video, direct supervision), wizard output (navigation, facial expr., speech)
    - What are the wizard production variables and how were they controlled for?
    - How is multi-wizard control concerted in HRI?
    - What are the wizard recognition variables and how were they controlled for?
- Are there HRI tools like Ozlab: what could 'generic' features of a WOz tool mean for HRI developers?
- Are there any limitations to the Wizard-of-Oz methodology in HRI?

2 The Ozlab system

Ozlab is a WOz supporting system developed at Karlstad University since 2001. Ozlab may be used as a tool for designing, testing, evaluating, experimenting and discussing graphical interfaces and interaction design, before effort is put into development in any programming language. Conceptually, the wizard functions in the present web-based reincarnation of the Ozlab system originates from the 2001 Ozlab system, which was based on Macromedia Director.

2.1 Director-based Ozlab (2001-2012): System overview

In Ozlab there are no automatically functioning prototypes. The prototypes, in Ozlab terms called the *interaction shells*, are manually controlled by a wizard. Pettersson argues that Ozlab "[…] supports explorative experiments in interactivity design by letting experimenters manipulate directly the output on the user's screen. The focus is specifically on simple graphical human-computer interaction." (2002, p. 144) By using the Director based Ozlab system the outcome is not program code. Instead, the user of Ozlab can design and test a concept with the intended end-users, before any programming is conducted. Doing so, Molin and Pettersson (2003) argue that Ozlab "can aid the process of formulating the requirements specification for multimedia systems" (p. 78). Multimedia systems in this case refer to systems that "are characterized by the important role the system's extrovert parts have. […] Such systems are, to a large extent, defined through their user interfaces." (p. 70) The authors furthermore argue that "most information systems nowadays have their 'multimedia' parts" (p. 70).

The system was based on Macromedia Director 8.5 (or MX). To prepare and run a Wizard-of-Oz test several entities werre used: Ozlab Testrunner, Ozlab Setup, Ozlab FileUpdater and a template file (.dir) with pre-programmed Ozlab-specific functions. (Siponen, Pettersson & Alsbjer 2002)

To build and design a prototype the template file was opened in Macromedia Director. To design the interface of the prototype, the designer added graphics, text, videos and pre-recorded sound to the library (called "Cast" in Director) in the template file. As the Ozlab prototypes were designed in Macromedia Director, the built-in tools for e.g. drawing and writing could also be used to create objects. To make the prototype come alive, that is, to function on another level than just communicating the interface via plain pictures, the pre-programmed Ozlab-specific functions, called *behaviours*, were used to add certain functionality to the dummy objects. Such behaviour was, e.g. "objectMoveableByTP", allowing the test participant (TP) to drag and drop objects; or "textFieldEditableByTP" allowing TP to write text in input fields. By using the timeline in Macromedia Director (called "Score"), the designer could create different pages or as called in Ozlab, *scenes,* in the prototype.

To run an interaction design test in Ozlab Testrunner, the prototype file(s) needed to be copied from the wizard's computer to the test subject's computer. Further, the communication between the computers, handled by Macromedia Multiuser Server 3.0, needed to be established. These settings were configured in Ozlab Setup. Ozlab FileUpdater, using the settings from Ozlab Setup, was used to copy the file(s) from the wizard's to the test subject's computer, and after a redesign of an interaction shell only the changed files were updated to quicken time-to-test if changes were made while a test subject were waiting. When fully configured, Ozlab Testrunner was started on each computer, allowing an interaction design test or demonstration to start. (Siponen, Pettersson & Alsbjer 2002)

During the test or demonstration the wizard's and the participant's interface was mirrored. In order to control and simulate the "system's" responses the wizard had wizard-specific controls, such as navigating to different scenes, opening new interaction shells, hiding/showing objects, pausing the test, etc. The test participant's mouse cursor was duplicated as an enlarged cursor in the wizard's interface, letting the wizard easily follow what objects the participant interacted with (and therefore allowing the wizard to produce appropriate responses). Input from the participant was collected in a log, which could be consulted during or after the test session.

2.1.1 Director-based Ozlab: Usage and application

The Director based Ozlab system was used during courses given at Karlstad University, and in several research projects. For example: Molin (2004) used Ozlab to design and evaluate a touch screen interface for a hip surgery robot, collaborating with the prospective user groups and designers;

Pettersson (2002) reports on the pilot study of Ozlab made in autumn 2001 with inexperienced multimedia designers as wizards; Nilsson (2005) conducted user tests on a prototype of pedagogical software for children; in collaboration with Swedish Civil Contingencies Agency several iterations of user tests were conducted on different aspects of a software, reported by Nilsson (2006) and Kilbrink (2008); Pettersson and Nilsson (2011, p. 500) assessed "code quality when it was either programmed based on mock-upped and user-tested designs, initially made from perceived needs by real users, or programmed only according to perceived needs by real users"; and Lindström and Nilsson (2009) used Ozlab as a usability testing tool in the PrimeLife project, and Pettersson and others in the initial year of the PRIME project (cf. for example PRIME project, Privacy and Identity Management for Europe, a 6FP EU project; usability work reported in deliverable series D6.1.a-d, www.prime-project.eu). For further examples of previous Ozlab usage, see Pettersson (2003), which also includes a list of student theses, and the webpages about Ozlab.[6]

2.1.2 Plans 2011 for redevelopment of the Ozlab system

After Macromedia was acquired by Adobe in 2005,[7] the Ozlab system suffered from being based upon an increasingly outdated program. Several stakeholders suggested directions for a redevelopment, some conflicting. A student thesis collected and developed some of these thoughts and suggested several bases for a redeveloped Ozlab: Ozlab could be based on Adobe's Photoshop (for quick integration with graphic design work), an HTML5 editor, or XML. The authors noted that building Ozlab based on Photoshop or an HTML5 editor would make the system dependent on software and certain file types, as well as being less accessible to inexperienced users; the authors argued that Ozlab should be based on XML (Lamberg & Brundin 2011, p. 42).

For the ongoing re-development another main mark-up language was chosen: HTML5 combined with JavaScript. (An XML solution would in fact not be a strict XML solution. Javascript and other programming languages would, actually, be needed; Lamberg 2011.) This HTML5 solution is not dependent on a particular HTML5 editor, as we now integrate the

[6] The Ozlab webpages can be found at http://kau.se/en/ozlab (or the old version, accessed through www.is.kau.se/~jsp/ozlab); Ozlab itself was not translated to English until 2007 by Jenny Nilsson in collaboration with Christina Hochleitner, then at CURE which participated in the PrimeLife project.
[7] Wikipedia, s.v. 'Macromedia'. http://en.wikipedia.org/wiki/Macromedia [2014-01-04]

editorial functions in the wizard controls by letting the system be a web service (i.e., like a cloud service). Of course, we are then dependent on a web server. However, running with multiple users as WOz experimentation necessarily implies (at least two users, one TL and one TP), a server solution seems unavoidable. A web solution will be a bit sluggish but brings several benefits such as ease of setting up experiments in various places. Pros and cons will be discussed further on.

2.2 Web-based Ozlab Step 1 – design and implementation

The first version of the web-based Ozlab system, "Step 1", uses the following techniques and frameworks: ASP.NET MVC 4.0, Microsoft's web server IIS 8 with WebSockets[8], the JavaScript library JQuery[9], and Sencha Ext JS 4.2, which provides widgets such as drop-down lists and text fields (but only for traditional mouse input; however, the spring 2014 version Ext JS 5 includes support for touch input; see www.sencha.com). Ozlab can be accessed via any web browser but runs best in Google Chrome.

Ozlab consists of two main entities: Shell Builder and Test Runner. When accessing Ozlab in a web browser, the user ends up at a landing page where one can choose between four roles: As a prototype designer the user can:

- "Build or edit shell" (which will start the Shell Builder);
- Start a test as a **Test Leader** (wizard) by choosing a shell and scene in that shell (this will open the wizard's view of the Test Runner);
- Join a test session as a **Test Participant** (this will start the participants' view of the Test Runner);
- View a test session by starting the **Test Viewer** which is a simplified form of the Test Runner mode which gives a view of the mockup but does not allow for any input/interaction with the shell (and its wizard).

[8] "The WebSocket Protocol is an independent TCP-based protocol. Its only relationship to HTTP is that its handshake is interpreted by HTTP servers as an Upgrade request." RFC 6455, Internet Engineering Task Force, December 2011. http://tools.ietf.org/html/rfc6455#section-1.7

[9] "jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers." jquery.com [2014-07-14]

Ozlab terminology assigns the identifiers **TL**, **TP**, and **TV** to the roles connected to the three user interfaces for a running session. Figure 1 depicts the interrelationship between these roles. Multiple wizards can be assigned subscripts: TL_1, TL_2, etc. and can also work outside the system (e.g., with live voice feedback via microphone and loudspeaker).

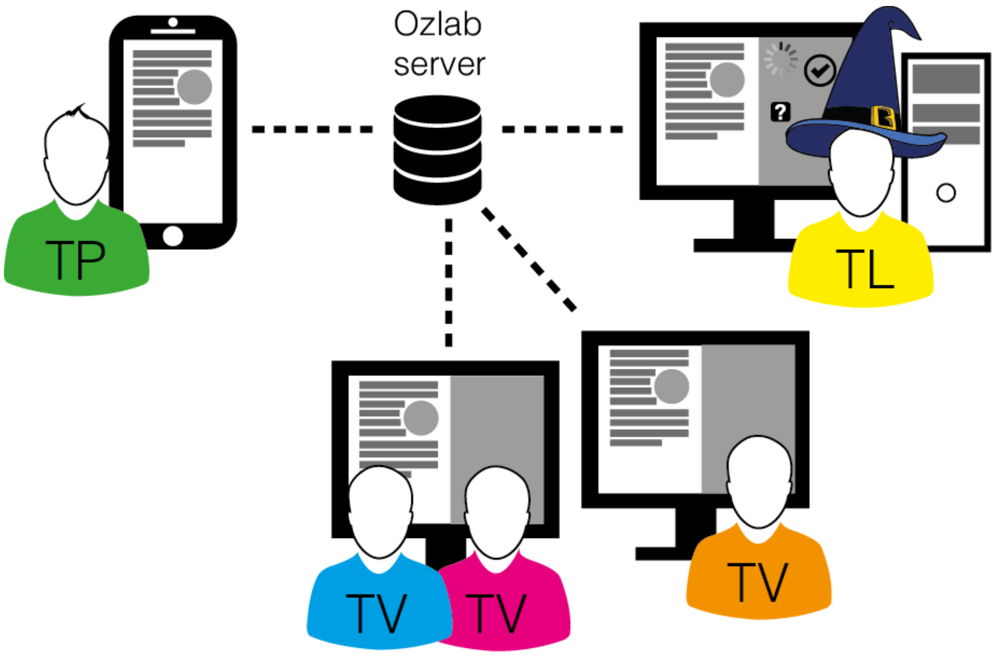


Figure 1. Every browser connects to the Ozlab server. The view of the interaction shell depends on the role. TL has several controls outside the TP scene while TVs only sees TP's view.

When building and editing an interaction shell in the Shell Builder, no "wizardry" is going on: TP cannot see the interaction shell or any changes made in the shell.

The TL, TP, and TV views can be run on the same computer but in different web browser windows, allowing the designer to preview the interaction shell easily. The current implementation allows one wizard, one test participant, and several Test Viewers to be connected to the same session (1 TL, 1 TP, multiple TV for each session). An experiment can possibly include more than one session: TP can have several browser windows open before him or one can let the TP browser alternate between different sessions because these are identified by web addresses. In fact, the web solution makes it possible to run Jack-in-the-box Wizard-of-Oz sessions by including one website (i.e. one interaction shell) within another utilizing so-called embedded iframes. Jack can then be another wizard than the one controlling the main session. Thus, the web-based Ozlab implicitly allows multiple TLs to act in the same session. Remains still to evaluate to which extent transmission speed problems are multiplied when Ozlab sites (i.e. interaction shells) with heavy content such as large images are tangled. In addition, some restrictions of what a TL sees will occur and a side channel tapping TP's monitor can be useful (but making mobile experimentation harder).

2.2.1 Shell Builder – designer's/test leader's workspace

An interaction shell can be designed and edited in the Shell Builder (see Figure 2). To the left a General Panel is placed where Scenes are listed and general Shell Settings available. In the middle of the interface the chosen scene is displayed. The size of the scene can be altered under Shell Settings. In Figure 2 the scene area (which is what the user will be able to see during test) is 500 by 400 pixels (displayed as a white sheet). The designer can add generic objects (seen in Object Panel to the right) to the interaction shell by dragging and dropping them from the panel to the scene area. In Step 1, released summer 2013, 7 generic **objects** were available:

- *Button*
- *Image*
- *Input field*
- *Label*
- *Dropdown menu*
- *Radio button*
- *Checkbox*

The Image object can hold several common file formats such as .jpg, .png and .gif. The Label object can hold text or be used to embed iframes (iframes can be used to open up a Google map for instance; however, TL cannot see what TP is doing when interacting with such an element unless there is a video tap from the TP screen to some monitor near the TL, but see the beginning of 2.2 about Jack-in-the-box wizardry). Text in Label objects can be formatted with different fonts, sizes and colours.

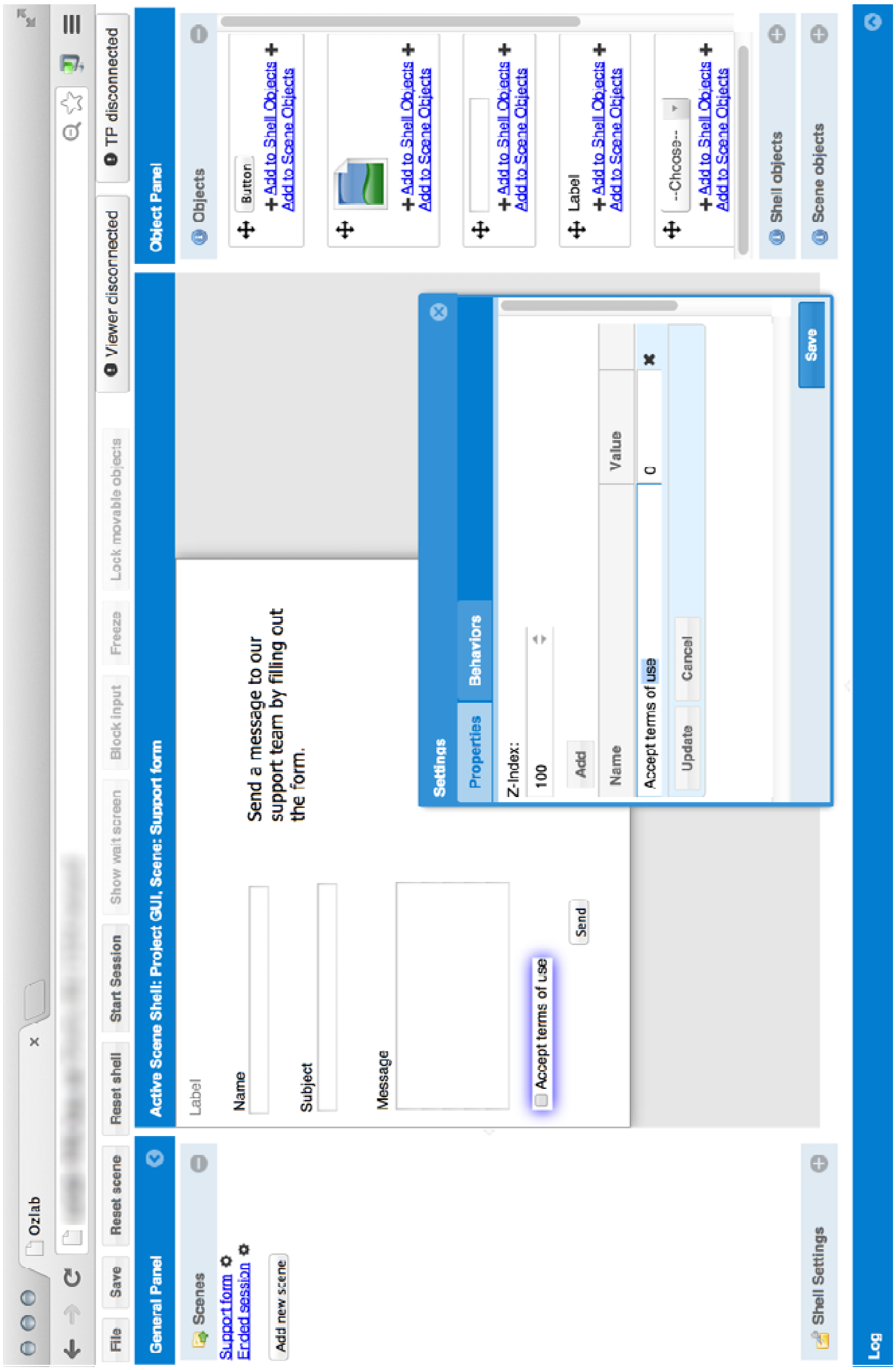
All objects come with settings, such as changing the text for a checkbox (as seen in the lower middle of figure 2).

All objects can be made invisible whenever TL so decides. In addition, Ozlab provides the designer with a set of optional **Behaviors** that can be added to objects. In Step 2 (March 2014) of the web-based Ozlab system, there are ten optional Behaviors: *GoToScene* provides an automatic link to another scene in the same shell; *MakeObjectSnap* will centre a movable object over a snap point; *ObjectMovableForTL* and *ObjectMovableForTP* allow Test Leader and/or Test Participant to drag the object which has the behaviour; *ObjectInvisibleMoveForTL* and *ObjectInvisibleMoveForTP* work like the other "Movable" options but the object is automatically invisible when being dragged (so far only the TL version has been used, e.g., to introduce alert boxes quickly); *OpenLink* will, if the object is clicked during a running session, open a link to an external website in the browser window; *SaveValue* stores input values in hidden fields which are called from other scenes from label objects – in this way the shell constructor can arrange for summaries of certain TP selections and TP text input during a session (selections in Dropdown menus, Radio buttons and Checkboxes will be stored by the text value of each selected row, or by a pre-specified alternative value); *SendAudio* allows pre-recorded audio to play; and *Vibrate* will make an Android device vibrate when TL calls this function (for iPhones this will just result in a dialogue box popping up informing on the attempt to call the vibration function).

The Behavior OpenLink may require some further comments to demonstrate the possibilities and thinking that a cloud service solution entails. With the label objects one can insert a link by HTML code (<a href="http://..."> </a>) as mentioned above. Alternative, in the label settings one can choose "Hyperlink" which makes the selected text into a link. This is of course easier for people who are not familiar with HTML coding. However, the advantage of embedding HTML code in a label object is that TL can select part of a text to become the link rather than the whole label object. What is more, TL can decide whether the link shall open in the present tab/window or in a new one. (However, due to browser security restrictions, TL cannot open an external web page for TP. TP has to click the link her/himself.)

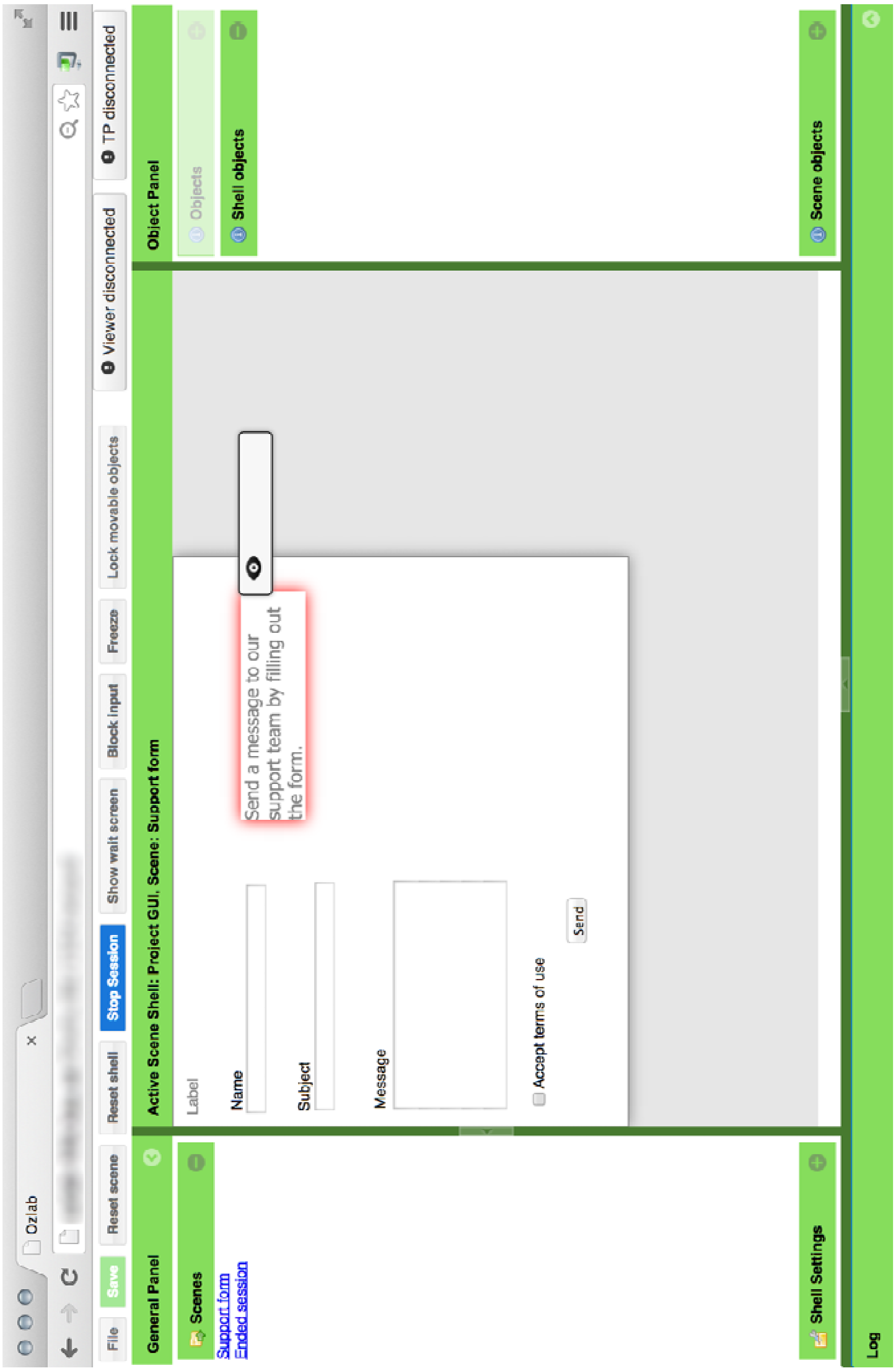
Complex or commonly used objects can be added to either the panel *Shell objects* (available throughout the whole interaction shell) or to *Scene objects* (available at the current scene only) for re-use. The Scene or Shell objects can also be used as holders for formatting and style, as in Microsoft Word where styles can be reused and added to a text throughout the whole document. Objects dragged and dropped at the scene area directly from the generic Objects pane cannot be reused, which is why the designer must start by adding an object to the Shell or Scene objects pane if re-use is expected (we plan to change this in the future).

Because the interaction shell and different panels are shown simultaneously in the Shell Builder, it runs best on a large screen.



Send a message to our support team by filling out the form.

Figure 2. The Shell Builder interface (Step 1). Note the button "start Session" which is a way to start Test Runner for wizards (TL).



Send a message to our support team by filling out the form.

Figure 3. The Test Runner interface as seen by the wizard, in version 1 of the web-based Ozlab system.

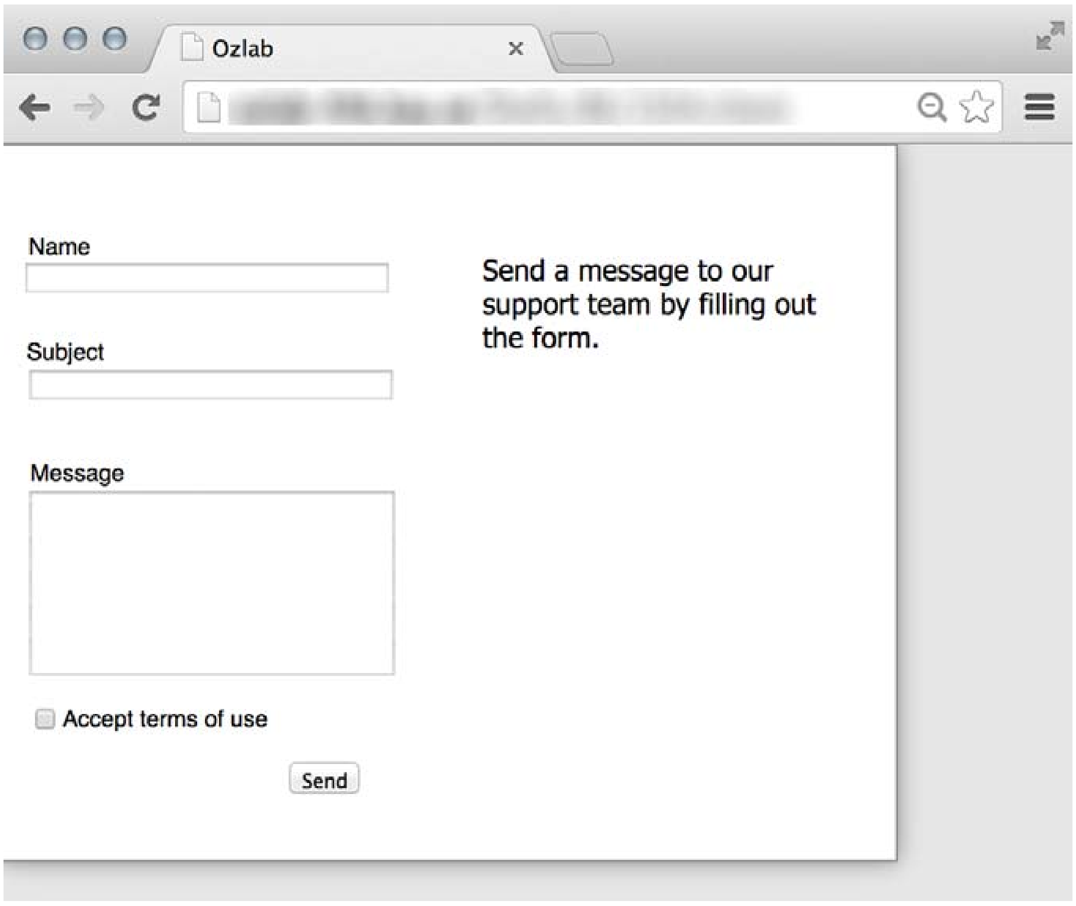
2.2.2 Running a test session (or a demo session) as TL

The Test Runner is the test state of the Ozlab system. When clicking the button “Start Session” or by choosing the Wizard role (TL) at the landing page, the Test Runner is started in the browser window. The wizard’s interface has the same overall look in Shell Builder and Test Runner, which is why a green colour is used to differentiate between states (see figure 3). The wizard can follow the TP’s actions in real-time as they appear in the scene area (in the middle), where also the TP’s mouse cursor is duplicated in an enlarged version, followed by a fading trail. User actions that can be followed are for example clicks, text inputs, dragging and right/left click. As the whole TP user interface is displayed (mirrored), Test Runner runs best on a large computer screen in TL mode.

There are some differences between the Shell Builder and the Test Runner, as some functions are removed and some are added. For example, the wizard cannot add objects to the shell in runtime in the current implementation, which means that all content and all objects must be created prior to tests. Conversely, objects that have been added as Shell or Scene objects can be dragged and dropped to the scene. The wizard cannot change the settings of objects during runtime either, but all objects can be hidden from the user (by clicking the eye icon in the tool popup displayed in the middle of the scene area in figure 3). The wizard can choose from a set of functions at the top of the interface. By clicking *File*, an interaction shell can be opened during run-time; *Reset scene* and *Reset shell* will reset all objects to their original state in either the active scene or the entire shell; *Stop Session* will terminate the test session and take the wizard back to the *Shell Builder*. The remaining four buttons allow the wizard to: show a black waiting screen for the test participant (*Pause*); show the test participant’s cursor as a wait cursor (*Lock*); make it look like the test participant’s cursor is entirely frozen (*Freeze*); and lock all movable objects for the test participant (*Lock movable objects*). Pause and Freeze allow the wizard to make changes in the scene or navigate to a different scene, without displaying the changes to the test participant. The test participant cannot continue to interact with the interaction shell during these states.

2.2.3 Participating in a test session (or a demo session) as TP

When running the Test Runner as a test participant, the interaction shell is displayed without the wizard’s controls (see figure 4). An interaction shell as shown in the figure where the scene area is smaller than the browser window, will display a grey area outside the scene area. Objects added by the wizard in the grey area will not appear in the test participant’s interface. The browser can be run in full-screen mode if the test leader wants to hide the chrome for the user, i.e. hide every trace of the window being a web browser's window. Hiding the typical browser window properties are essential, especially if conducting tests on a prototype of a system which is not intended to be viewed in a browser window. Hiding the browser controls is important also when running a test of a mocked up web application. Otherwise the test participant will be bewildered. Furthermore, TP can go to other web sites or close the browser window, which obviously will make the wizard lose control of the experiment.



Send a message to our support team by filling out the form.

Figure 4. The Test Runner interface as seen by the test participant (when the browser controls are not hidden and when the browser window is much larger than the GUI tested).

2.2.4 TV – Test Viewer

Test Viewer works like a video monitor as it shows running sessions without letting the spectators engage in the on-going interaction. It displays the interaction shell and participant's enlarged cursor without any wizard controls.

2.2.5 Design of the Ozlab start page

During an internship in February-March 2014, two students proposed a re-design of the web-based Ozlab start page, which presently allows the user to select shell building or test running as TL, TP, or TV. Their idea was that the URL would determine which view of Ozlab the user was presented with. If accessing the Ozlab system with the default URL, a start page adapted to test participant would be shown – thus not revealing that there are different roles. If "/wizard" was added to the end of the default URL, a start page adapted to test managers would be shown (or a 'full' start page can be shown allowing the user to select any of the four roles as we assume this user to be a designer or wizard). Two separate start pages addressed in this manner would decrease the potential risk of giving away the test method (Wizard of Oz) to the test participant and perhaps more importantly, simplify the graphical interface for test participants as well as test leaders.

2.3 Recording and logging sessions

Collecting data during a session could be made in several ways:

- Recording TP (screen recording, voice recording, video of TP)
- Logging (events in the Ozlab system are stored for analysis)
- Note-taking by TL (written and spoken notes)
- Post-session questionnaire for TP (also interview which can be recorded in screen capture with sound is being made)
- Post-session annotations (from memory and from recordings made)

The present logging function in Ozlab merely records a number of events. As of spring 2014 there are no functions to make analyses of different features, nor is there a possibility for a TL to make notes connected to the stream of logged events (this would typically be made by a different TL than the one acting as Ozlab wizard).

There are sometimes log data which are needed during the actual running session. For instance, if an interaction shell has several buttons that can be clicked in random order by TP (a prototypical case is the numbered keys on a telephone), the wizard may not be able to follow TP's clicking and act on these (for instance, putting them in an output text field such as "Sending message to [telephone number]."). So having a pane with log information can be useful even during a session, not only afterwards for analysis. The rudiments of such a function exist in Ozlab. As noted before, there is also a copy function *Save Value* so that selections and text inputs can be reused in a text object within the same or another scene.

Despite the possibility to make more use of the logging, during spring 2014 the idea to turn it off or selectively turn it off in order to improve speed was also discussed. The present logging functions are suspected to slow down the system – presumably the operating system is busy copying data to the hard disc.

3 Generic solutions besides the Ozlab system

As already mentioned, what one team finds 'generic' might well be regarded as specialized or limited by other teams, but because quite many systems have been developed with reusability in mind the intention of the present chapter is to gather as many such examples as possible in order to see general trends and problems. In order to provide an overview for the reader, the first section merely lists all the systems with very brief annotation and some core bibliographic reference. Then section 3.2 provides some additional lines or sometimes more extended discussions for each system to comment on trends or special features. Finally, section 3.3 discusses a list of requirements for generic Wizard-of-Oz tools that one of the teams had composed and the following chapters will further expound on such topics.

3.1 List of generic WOz tools reviewed

ActiveStory a web-based Wizard-of-Oz testing tool which appear to be discontinued (http://activestoryenhanced.codeplex.com/).

ConWIZ is a part of the Contextual Interaction Framework (CIF, see Zachhuber et al. 2012 for description and evaluation of the Contextual Wizard of Oz framework), for ubicomp environments (Grill, Polacek & Tscheligi 2012). Through an Android device tool, **Mobile Wizard**, the wizard can "send commands to external applications" (Zachhuber et al., ibid., p. 231).

DART: The Designer's Augmented Reality Toolkit is described by MacIntyre, Gandy, Dow, and Bolter (2004) as targeted on testing in early design phases. DART is integrated in Macromedia Director.

DiaWOz-II is a configurable software environment for WOZ studies where a combination of mathematical input and natural language is used. DiaWOz-II, based on TEXMACS, is not an improved version of the predecessor DiaWoZ system. (Benzmüller, Horacek, Kruijff-Korbayová, Lesourd, Schiller & Wolska 2007)

d.tools, see SUEDE below and in 3.2. "d.tools extends SUEDE's framework into a new application domain – physical user interfaces." (Hartmann, Klemmer, Bernstein, Abdulla, Burr, Robinson-Mosher & Gee 2006)

Jaspis is a distributed software architecture that can be used for WOz studies on speech user interface and ubicomp (Mäkelä, Salonen, Turunen, Hakulinen & Raisamo 2001).

LIVE presented by Li and Bonner (2013) seems to simply duplicate ("mirror") wizard's screen onto something that the test subject(s) can see (and act upon).

MDWOZ is a spoken interaction dialog systems. (Munteanu & Boldea 2000)

Mobile Oracle is described as a "novel tool for eliciting user requirements early in the design process of mobile applications" (Magnusson, Anastassova, Tolmar, Pielot, Rassmus-Gröhn & Roselier 2009).

Momento is dedicated to ubicomp applications and experience sampling method (long-term experiments, multiple TPs), according to Cater and his co-workers (e.g. Carter, Mankoff & Heer 2007). SMS and MMS are used for communication but also WLAN.

MultiCom consists of an observation laboratory where WOz studies can be conducted. The laboratory also includes other software and hardware. (Caelen & Millien 2002)

MuMoWOz is a tool for conducting tests on multimodal mobile systems. (Ardito, Buono, Costabile, Lanzilotti & Piccinno 2009)

NEIMO is a platform for multimodal interaction. NEIMO supports a several wizards setup. (Coutaz, Salber, Carraux & Portolan 1996; Coutaz, Nigay & Salber 1995)

OpenWizard is a "component-based approach for the rapid prototyping and testing of input multimodal interaction" (Serrano & Nigay 2010, p. 224). OpenWizard replaces non-fully developed components in a prototype with generic wizard components.

Ozlab was constructed as a GUI WOz system as there were no general graphics-supporting WOz systems at the time (Pettersson 2002). From 2013 this system is being replaced by a web-based version with similar features (www.kau.se/en/ozlab).

SketchWizard supports early prototyping of user interfaces incorporating pen-based interaction. (Davis, Saponas, Shilman & Landay 2007)

SUEDE (Klemmer, Sinha, Chen, Landay, Aboobaker & Wang 2000) is a tool for prototyping speech interfaces. All (simulated) system output needs to be added to the prototype on beforehand.

Topiary is a tool for prototyping location-aware applications. (Li, Hong & Landay 2004)

UISKEI++, reported by Segura and Barbosa (2013), is the intended evolution of the tool UISKEI (User Interface Sketching and Evaluation Instrument). UISKEI++ is envisioned to support WOz experiments and prototyping on multi-devices by providing multiple abstraction levels. The multiple abstraction levels are argued to allow the designer to compare the prototypes, regardless of device. (Segura & Barbosa 2013)

WebWOZ is web-based and focuses on flexible incorporation of Language Technology Components (LTC). (Schlögl, Doherty, Karamanis & Luz 2010) Experiments resulted first in sketches (Schlögl et al. 2010) and later in prototypes (Schlögl et al. 2011). Schlögl, Chollet, Milhorat, Deslis, Feldmar, Boudy, Garschall and Tscheligi (2013) report on their progress of offering voice controlled Home Care and Communication Services, vAssist, which in the future will be developed using WebWOZ.

Wizard of Oz tool for Android allows digitally created or scanned paper prototypes to be tested. The prototype must be developed beforehand because the tool automatically creates a folder with prototype specific objects that must be transferred to the Android phone. Communication TP – TL enabled by modified open-source VNC client for Android. (Linnell, Bareiss & Pantic 2012)

WOEB by Bellucci, Bottoni, and Levialdi (2009) is developed for the rapid setting up Wizard-of-Oz experiments. Progressive refinement finally replaces the wizard and results in a multimodal mobile application.

WozARd is a tool for WOz experiments on mobile phones, tablets and glasses. WozARd is location aware and accepts images, video, and sound to be uploaded into the prototypes. It logs test results and visual feedback on removable media. (Alce, Hermodsson & Wallergård 2013)

WOZ PRO is a tool for constructing low-fi prototypes and utilising these in WOz experiments. (Hundhausen, Trent, Balkar & Nuur 2008)

3.2 Review of the generic WOz tools

As noted in the introduction, there are several tools that incorporate the Wizard-of-Oz technique. Some are developed for testing on specific devices, some for simulating certain modalities or aspects of a prototype, while a few are more generic. Below, the more generic tools found in the literature review are presented.

Reasonably, one could ask why there are so many attempts at developing generic tools supporting WOz experiments, and why new tools are still being developed. If one generic tool has been developed, should there be any need for additional tools? After all, WOz tools are constructed to support prototyping without programming so why all this programming of tools?

First of all, there are new interaction devices coming up, and to control and sometimes to collect inputs, the generic system has to be expanded. Of course, different application areas will find different wizard controls the most essential. The systems described here are probably all non-generic if the term 'generic' indicates that no traces at all of some specific application domains are present. On the other hand, some controls can be reused for quite different studies. Therefore, tools that can be configured and re-used for several studies are included here.

Secondly, other kinds of system changes occur at regular intervals. For Ozlab, the discontinuing of the system the tool relied on to make the prototypes, along with the always on-going changes in operating systems, finally made it hard to simply run it. Naturally, avoiding tool-specific programming environments is to be recommended, but the changes in operating systems will regularly call for up-dates. Failing to maintain one's system will always threaten a generic system. It seems that few system have survived for a decade as Ozlab did, and it was limping in the end. Even if the new version of Ozlab is based on the 'latest' web technology, this technology, as well as supporting technology such as servers, will gradually be less suitable in the ever-changing digital ecosystem. This applies, of course, to any of the newer tools reported here.

For the account below, it should be noted that the software of the other systems has not been downloaded and tested; the information provided here is based on published posters and papers, and in some cases on project websites.

ActiveStory is a web-based Wizard-of-Oz testing tool which appears to be discontinued. "ActiveStory is a tool for designing and performing usability testing on an application in a manner that is in line with Agile principles.

Designers can sketch UIs, add interactions and export the design to the Internet via a built in web Wizard of Oz system." (http://activesto ryenhanced.codeplex.com/; the documentation page is empty but some other pages provide information; there seems to be no update since 2009).

ConWIZ is a part of the Contextual Interaction Framework developed at Universität Salzburg. "For integrating the WOz method into the framework we developed a set of 'contextual Wizard of Oz tools' as part of the presented framework", Zachhuber, Grill, Polacek, and Tscheligi (2012, p. 237) explain. ConWIZ, as described by Grill, Polacek, and Tscheligi (2012), provides an opportunity for designers to create and evaluate prototypes in mobile and ubicomp environments, that is, application domains where it is difficult for a wizard to observe the participants.

According to the developers, by focusing on mobile and ubicomp applications, "[…] the requirements for a mobile Wizard of Oz tool have been elaborated and taken into account when designing the ConWIZ system. The goal was to design and develop a system that is applicable to a large range of WOz studies without investing much effort into the development of the study. The ConWIZ system has already been used or is planned for use in contexts such as navigation, factory, smart home, or car where it already showed its usefulness." (p. 1)

The last example actually illustrates the difficulties pointed out earlier to define what a generic WOz tool would cover. How much of the functionality in a car should the wizard control? Well, it depends of course on what the future systems to be prototyped are imagined to do. For the Salzburg group, speech processing is within the limits: "The system can be easily applied to a variety of simulated or real contexts like e.g. the car context where it is possible to simulate e.g. handling phone calls by voice where the speech recognition is replaced by the wizard." (p. 2) On the other hand, 'wizarding' a driverless (or self-driving[10]) car would entail quite another WOz system than handling phone calls.

An interesting feature of ConWIZ is that even if its developers stress out-of-lab contexts, they have also included ways to trigger actuators: "The simulation of the *navigation system context* is described in this paper where the wizard simulates a navigation system by sending voice commands to the participant and controls contextual parameters such as wind simulated by a fan and vibrations for expressing danger. In a study using a simulated

[10] See, e.g., http://googleblog.blogspot.se/2014/04/the-latest-chapter-for-self-driving-car.html or http://time.com/79315/google-car-city-streets/

factory context the ConWIZ system has been used to simulate and control the behavior of machines and ambient alerting modes." (pp.1f, referring to the above cited paper by Zachhuber et al. 2012).

Another interesting feature is an annotation function added to the Mobile Wizard. The mobility of the wizard would be greater if not other test managers such as observers have to run after him/her when he is following a test user. Including an annotation function for the wizard may thus be considered. However, as anyone who has experience with Wizard-of-Oz experiments knows, writing usability reports simultaneously with conducting the prototype's behaviour is not easy (rather, even the wizard's work may be distributed over two wizards, as described for Ozlab; Pettersson 2003). Grill and co-workers report that "In the *Mobile Wizard*, the note-taking functionality was implemented in a way that it supported 12 different categories of errors which could be logged via simply pressing a button. In a stressful situation, such diversification of possible errors is too high as the wizards do not have time to explicitly search for the error category. However, the interviews showed that the functionality itself makes sense and is desired by the wizards." (p. 7)

As for the development of a WOz tool it is noteworthy that in the study reported in Grill et al. (2012), 4 actors were employed as "TP" to phrase it in Ozlab terminology, while the "TLs" were 8 in number testing wizardry by ConWIZ. That is, the number of wizards was double that of what one normally would call 'test users', reflecting the study's focus on the wizards. Each TL ran two sessions, thus the TP actors had to perform four times each. One remarkable fact is that the developers of this *mobile* WOz tool dared to use the think-aloud protocol: "The wizards were asked to use the think-aloud protocol […] as in standard usability testing. To avoid an influence of the think-aloud protocol on the actors, they were instructed to react only on commands they obtained through the prototyped navigation system". (p. 6)

Interestingly for the present work, the authors also present a comparison of 18[11] WOz tools and requirements that good WOz tools should meet; see

[11] The number does not correspond to the number of generic WOz tools in this section. We have not included CSLU Toolkit for NLP (Sutton et al. 1998), Polonius for HRI (Lu & Smart 2011), iCAP (see chapter 1), and Humaine (again, see chapter 1), or "Wizard of Oz 2" (Grill does not give references to all systems), while Grill and co-workers have not included Ozlab, neither Jaspis, LIVE, MDWOZ, Momento, MultiCom, Linell's Android tool, and of course not UISKEI++ and WozARd as these had not been presented in 2012.

further discussions in section 3.3 where these requirements are cited in full, and Chapter 4 where further discussions are made on requirements for generic WOz tools. For the longevity question raised in the beginning of this section, it can be noted that there is a plan to release the system as open source (see Grill and Tscheligi 2013 about the ConWIZ protocol and the project web site http://cif.hciunit.org/CIF/joomla/).

DART, The Designer's Augmented Reality Toolkit, was built on Macromedia Director. "Our work focuses on supporting early design activities, especially a rapid transition from storyboards to working experience, so that the experiential part of a design can be tested early and often. DART allows designers to specify complex relationships between the physical and virtual worlds, and supports 3D animatic actors (informal, sketch-based content) in addition to more polished content." (MacIntyre, Gandy, Dow & Bolter 2004, Abstract). "DART contains the necessary building blocks for distributed WoZ interfaces, but we have just begun to explore the potential for WoZ interfaces to AR experiences." (ibid., p. 206)

Dow, Lee, Oezbek, MacIntyre, Bolter, and Gandy (two publications 2005) used DART to evaluate a Mixed Reality application: the "Voices of Oakland". The application gives the user an audio experience in an historic site using location tracking. The WOz tools in DART were used to build the prototype and the wizard's interface controlling the prototype. Through three iterations the wizard's interface and the tasks for the wizard altered, from a high (simulating location tracking and media presentation, while monitoring environment) to lower (letting the user trigger the chosen audio segment) cognitive load for the wizard.

DiaWOz-II is a configurable software environment for WOZ studies where a combination of mathematical input and natural language is used. DiaWOz-II is not an improved version of the predecessor DiaWoZ system as the researchers wanted to base the new system on the WYSIWYG editor TEXMACS. DiaWOz-II consists of two interfaces, one for the student and one for the tutor (the wizard), which are connected via a server. (Benzmüller, Horacek, Kruijff-Korbayová, Lesourd, Schiller & Wolska 2007)

Jaspis is a distributed software architecture that can be used for WOz studies on speech user interface and ubicomp, as reported by Mäkelä, Salonen, Turunen, Hakulinen, and Raisamo (2001). In Jaspis a wizard can replace every module and therefore allowing different aspects of the system to be tested and evaluated.

LIVE presented by Li and Bonner (2013) is hard to classify. LIVE is said to be a "platform to support different ambient media applications" (p. 4). It seems to simply duplicate ("mirror") Wizard's screen onto something which the test subject(s) can see (and act upon). The authors present their setup as more generic than other WOz setups: "structural improvements to the conventional wizard-of-oz method" (p. 3). It is claimed that in traditional WOz, "the designer had specifically designed control panels to interact with the system and users" (p. 3[12]). However, employing a shared area of action is used in other systems (e.g., Ozlab, Pettersson 2003) as this indeed helps the wizard to "interact with and manipulate the user operations", but specific controls for the wizard even further facilitate interaction if these controls have been found useful in widely different experiments (ibid.). For the full range of Li's WOz experiments, see Li's dissertation (2012).

MDWOZ is a module-based development environment, running on desktop-to-desktop configurations, for the design and test of spoken interaction dialog systems. The system incorporates WOz by letting the wizard simulate the systems understanding of user dialogue input, and by giving the wizard a possibility to generate the output. (Munteanu & Boldea 2000)

Mobile Oracle is described in a 2-page poster presentation as a "novel tool for eliciting user requirements early in the design process of mobile applications" (Magnusson, Anastassova, Tolmar, Pielot, Rassmus-Gröhn & Roselier 2009). The original study generated a lot of data that has been used in several other works and influenced further method development, but the Mobile Oracle itself has not been re-used (p.c. Charlotte Magnusson 2014-09-04).

Momento was dedicated to facilitate situated experimentation to avoid shortcomings of basing ubicomp development on pure lab experiments (Carter, Mankoff, Klemmer & Matthews 2008; Carter & Mankoff 2005a&b, "Prototypes in the wild", "The role of media in diary studies"). Carter, Mankoff, and Heer (2007) mention three specific problems for *situated evaluation*, namely (1) remote testing, (2) adoption and retention (participants

[12] They give only one example, viz. Ruyter et al. (2005), who present a WOz experiment for a robot whose "head" was controlled by the wizard (facial expressions and head movements). Obviously, for such applications the simple duplication of a screen image is not possible or at least, the appropriateness of its fidelity could be questioned relative the experiment goals. For other application areas, the structural improvements claimed by Li and Bonner seem already to have been made by others as this subsection demonstrates.

might not remember afterward so certain recordings/photos were made), and (3) the infrequency with which some events of interest may occur. In Momento, SMS and MMS are used for TP-TL communication but also WLAN when possible. TL can prompt TP to give more details. TL typically works at a desktop computer interface to the Momento server, but mobile access is also implemented – this extension was probably easy to make as the system allows for multiple test participants and these typically communicate with the Momento server from mobile devices. So-called *Experience Sampling* (Larson and Csikszentmihalyi 1983, or cf. Consolvo et al. 2007 referred in section 1.2) was in focus for the development, but WOz is also supported: "To support experimentation at the early stages of design, as well as to support experimenters with limited coding experience, it is important not to require complete applications. One way to facilitate this is to support a Wizard of Oz protocol in which experimenters can do some of the work normally done by an application." (Carter et al. 2007, p. 3) Interestingly, as the work was based on the Experience Sampling Method, annotations functions for the TL was found useful (in contrast to the experienced referred above for ConWIZ) – in fact, Carter and his co-workers even found TLs suggesting that TPs "should be able to annotate media captures via the web" (2007, p. 8).

For mobile experimentation, the developers found TPs unwilling to use special mobile devices, so integration in participants' handsets was made possible even if all functionality might not be available (this also demands compensation for the cost TPs has for the SMS and MMS they send). Naturally, this geared Momento towards what was the typical hardware around the years 2005-2007, i.e. *not* towards smartphones. (The "Documentation" page of the website www.m0ment0.com starts by saying "Please note that this software is not currently being supported".)

For local (site-constrained) experiments, the developers seem to have found LAN as useful as the Ozlab developers, but also the pitfall of server configuration: "However, the server configuration had involved some complicated manual processes that the experimenters initially found too difficult to complete. To address this, we streamlined server configuration to the point that the experimenters needed only to run four commands to configure the core system and their study." (2007, p. 9) "The experimenters also ran a Momento server on the laptop and configured the laptop for peer-to-peer wireless networking. In this way, the mobile devices could connect directly to the server running on the laptop […]" (2007, p. 8)

MultiCom can be described as a platform or facilities for the design and evaluation of interactive systems. MultiCom consists of an observation laboratory where WOz studies can be conducted. The laboratory also includes other software and hardware. MultiCom was used in a field setting together with WOz to test a Residential Gateway and Home Service system. The "Wizard of Oz technique served to simulate some unimplemented actions such as automatic motion detection, alarm notification, appointment reminder, or the execution of spoken commands", according to a report by Caelen and Millien (2002, pp. 157f).

This article opens by an extensive quotation from Bernsen's, Dybjkaer's, and Dybkjaer's *Designing interactive speech systems – From first ideas to user testing* (1998). In the quotation it is explained how difficult it is to produce a new software engineering tool. A central problem is *generalisation*: the tool must work in many systems and for several domains of application to be of interest to developers. But there is also a second problem of *objectivity*. The benefits of a new tools should originate from the tool (or method) itself, not from its inventor." Caelen and Millien use this and the observation that "now[adays] usability activities are realised continuously during the lifecycle of the system" to argue for a platform "defined as a *service centre* devoted to experimentation." (p. 150) Thus, the WOZ system MultiCom is not only a piece of software but also consists of dedicated rooms for experimentations and hardware environment.[13]

MuMoWOz is a tool for conducting tests on multimodal mobile systems. The authors argue by using the appropriate multimedia content, MuMoWOz can be used for testing any scenarios. Though, the content must be created before the test (except text to speech output, which can be generated during runtime). The wizard simulates the system's output via a computer, connecting to the user's handheld device via WiFi. (Ardito, Buono, Costabile, Lanzilotti & Piccinno 2009)

NEIMO is described as a generic and flexible usability platform for testing, observing, conducting WOz experiments and analysing multimodal interaction. NEIMO supports a multiple-wizards setup. There is a heavy emphasis on data collection, both automatic and by annotating wizards.

[13] In experimentation in 2014, a MultiCom configuration called EmOz bases smart home, service robot, emotional interaction all in the same experimentation on how to make elderly people accept to control a smart home by speech directed to a little service robot (preliminary report by Aubergé et al. in May 2014). EmOz contains many features such as pre-recorded and voice-disguised answers, live voice disguise, emotional sounds output, robot movements, and, "in order to facilitate the use by non-programmer researchers", generation of user interface for the wizard (ibid.). http://domuslab.fr/projects/ ; http://iihm.imag.fr/en/publication/caffiau/

(Coutaz, Salber, Carraux & Portolan 1996; Coutaz, Nigay, & Salber 1995; see also section 1.2 for citations.) However, it does not seem to have been the case that the research group in Grenoble ran extensive tests generating data which could be statistically analyzed – the WOz method does not easily lend itself to this because of the risk of 'wizard fatigue' (if not more restricted, pre-programmed/pre-recorded outputs are what the wizards are to produce).

It could be noted that in a more recent paper, Serrano and Nigay (2010; see OpenWizard below) do not count NEIMO as a general tool, because it is specific to certain modalities (speech input and direct manipulations). That is not the approach taken here when collecting examples for this chapter as there always seem to be some inputs or outputs that systems are not ready to provide for, at least not without extensions. Rather, the 'generic' aspect of a tool should be that is allows for quick re-use between (some) application areas and that the tool also allows for extensions even if these potentially are more time-consuming (WOz should allow for specific setups to grab opportunities available in the circumstances, but therefore the WOz tool itself could not always be a part of every detail of the inputting and the outputting).

OpenWizard is a "component- based approach for the rapid prototyping and testing of input multimodal interaction." (Serrano & Nigay 2010, p. 224) OpenWizard incorporates the Wizard-of-Oz technique by replacing one (or several) component(s) in a non-fully developed multimedia prototype with generic wizard component(s). The work was not continued beyond the 2010 study (Serrano, p.c. 2014-04-30)[14].

As for NEIMO, this 3rd generation wizardry from Grenoble (MultiCom we count as 2nd) emphasises "a multi-wizard approach where each wizard has an identified role and is responsible for a specific well-defined part within the dataflow of input multimodal interaction, from devices to tasks." (p. 216) The reason is the multimodality, which also motivates the researchers to develop an experimental approach that mixes functioning and WOz modules – the authors provide several reasons for why a mixed WOz tool is to be preferred when it comes to multimodal (input) interaction (p. 218):

---

[14] Laurence Nigay, Head of the Engineering Human-Computer Interaction Group at Joseph Fourier University in Grenoble, explains in an email letter that the EHCI team is not working on a WOz platform but instead conduct case-by-case experiments. Nigay prefers to make a clear distinction between WOz platform and WOz experiment. (p.c. 2014-05-28)

1. For technical reasons it can be hard to integrate some input devices even if they work well separately.
2. "Testing a multimodal prototype [where] the integrated modalities are not at the same level of robustness will lead to biases in the results. As pointed out in [Oviatt 1999, p. 79], 'when a recognition error does occur, users alternate input modes'."
3. "Existing devices usually impl[y] a very specific physical installation."
4. Test participants may differ in how they integrate two specific modalities (e.g., Oviatt ibid.), and therefore it can be better to use a human operator (wizard) to interpret their intentions rather than trying to fix such calculations prior to testing the multimedia system concept.

The OpenWizard demonstration reported in the 2010 paper – a multimodal map navigator – was built on the OpenInterface framework and software platform (www.oi-project.org). While that demonstration shows the workability of the OpenWizard approach, a technical problem is indicated: test wizards "pointed out that the current latency of the system makes it difficult to simulate certain types of interaction, such as direct manipulation." (p. 224)

Ozlab was conceived as a "GUI articulator" as its inventor found this concept to capture what was missing in multimedia development in the 1990s and also missing is WOz setups reported: swiftness in the production of real-time GUI responses from a mockup to a test participant (compare the communication theory developed in Pettersson 1996; cf. 1997). Building a 'permanent' WOz setup, i.e. a WOz tool, should also make it more natural to include other groups than pure UI designers as wizards (Pettersson 2002; 2003). Field tests used to be made by a well-equipped laptop that connected to any other computer in the field. Several student thesis projects were facilitated in this way. However, a large screen helped the wizard much better to monitor and control not only objects in the test participant's screen but also accessing general wizard controls. (Pettersson 2003) Also more elaborate field setups have been made (e.g., Larsson & Molin 2006). Originally based on Macromedia's Director for prototype making, it is from 2012 being re-developed as a web system, both for making a prototype and running a demonstration or a test. Touch-specific input events are still undeveloped. Local-area WiFi configurations are planned to be developed (see Chapter 2 and in section 5.4, the point on "Connectivity outside the Internet").

SketchWizard supports early prototyping of user interfaces incorporating pen-based interaction. (Davis, Saponas, Shilman & Landay 2007)

SUEDE (Klemmer, Sinha, Chen, Landay, Aboobaker & Wang 2000) is a tool for rapid prototype creation of speech interfaces; they give an example of a telephone system for "reading and sending email" by voice. All (simulated) system output needs to be added to the prototype on beforehand. The prototype is generated as an HTLM file once the designer presses the "Test" button and the wizard then uses a web-like interface to control the pre-recorded responses. Klemmer, Sinha, Chen, Landay, Aboobaker and Wang (2000) argue that non-experts as well as professional designers can use SUEDE.

Notably, "**d.tools** extends SUEDE's framework into a new application domain – physical user interfaces." (Hartmann, Klemmer, Bernstein, Abdulla, Burr, Robinson-Mosher & Gee 2006, p. 307) While not design explicitly for WOz, the d.tool developers say that "A designer can later connect the corresponding physical control or, if preferred, even manipulate the behavior via Wizard of Oz at test time." (ibid., p. 301)

Topiary is a tool for prototyping location-aware applications, programmed in Java on top of a toolkit for pen-based applications called SATIN. Topiary supports testing prototypes on "a wide variety of PDAs and phones" (Li, Hong & Landay 2004, p. 223). When running tests in Topiary the wizard's interface and the user's interface can be run on either separate devices or on the same device. Location data are collected by searching for nearby WiFi points. The wizard can simulate location information.

UISKEI++, reported by Segura and Barbosa (2013), is the intended evolution of the tool UISKEI (User Interface Sketching and Evaluation Instrument). UISKEI++ is envisioned to support WOz experiments and prototyping on multi-devices by providing multiple abstraction levels. The multiple abstraction levels are argued to allow the designer to compare the prototypes, regardless of device. (Segura & Barbosa 2013)

WebWOZ is a web-based WOz tool with a focus on flexible incorporation of Language Technology Components (LTC). WebWOZ supports one wizard per test participant. (Schlögl, Doherty, Karamanis & Luz 2010) During the development of WebWOZ, much focus was on the wizard interface. The inventors argue that many wizard interfaces have been built but that they are often designed for specific experiments. The authors instead strive for a generic wizard interface of WebWOZ, suitable for different experiments focusing on LTCs. Further work aiming for a generic interface is reported by Schlögl, Schneider, Luz and Doherty (2011). To reach the goal, WOz experiments were conducted, using WebWOZ, while observing the wizards' behaviour and interaction with the wizard interface. Schlögl et al. (2011) identify two major problems for WOz interfaces: (1) supporting the wizard but still keep him/her in control over the interaction; (2) how to deal with response time issues. The experiments resulted in preliminary sketches (see two works from 2010 by Schlögl and colleagues) and later in prototypes (Schlögl et al. 2011; 2014).

Schlögl, Chollet, Milhorat, Deslis, Feldmar, Boudy, Garschall, and Tscheligi (2013) report on their progress of offering voice controlled Home Care and Communication Services, vAssist, "for seniors with chronic diseases and/or (fine-) motor skills restrictions" (p. 517). The authors plan on collecting data for the design of the voice interfaces using WOz, by incorporating the WebWOZ tool presented above with other systems needed. WebWOZ was chosen as it is open-source and "allows for testing interaction scenarios which employ one or more Language Technology Components (LTCs)" (ibid. p. 514; see also http://vassist.cure.at/home/).

Wizard of Oz tool for Android allows digitally created or scanned paper prototypes to be tested. User action, wizard action, time or the user's location can trigger screen transitions. The prototype must be developed beforehand because the tool automatically creates a folder with prototype specific objects that must be transferred to the Android phone manually. Communication TP – TL enabled by modified open-source VNC client for Android. (Linnell, Bareiss & Pantic 2012)

WOEB – the Wizard of Oz Experiments Builder – belongs to the 'evolution'-prototyping tool of the generic WOz tools: progressive refinement finally replaces the wizard and results in an application, in this case a multimodal mobile application. It uses WiFi (but also GPRS) for sharing audio and video streams between wizard and the monitored and controlled mobile unit. "In WOEB, the construction of WOz modules logic and interfaces relies on a metamodel and a reference architecture. A typical WOz environment provides a set of tools organized in a Client-Server architecture." (Bellucci, Bottoni & Levialdi 2009)

> "In a way similar to the Server Module, the WOz Client Module is automatically generated by the *XML Interpreter*. This module is essentially a data viewer, which can display content sent by the server and manage user pointing and oral interaction. Pointing interaction is captured through the trigger base, while audio inputs are processed by a *Speech Capture* engine. A description of a triggered event is sent, while the

> captured speech is streamed directly to the Dialog Manager on the server (e.g. the human wizard).” (ibid.)

This was actually the initial modus operandi for the Director-based Ozlab in 2001. However, in the first trial the persons used as test wizards were inexperienced in software design and could not manage two different files for each prototype: a source file to edit and an object file to run – it differed too much from what they were used to (cf. documents in Word, PowerPoint, and Excel which are never run as programs of their own). Ozlab was quickly re-implemented as a test-running program (instead of a compiler), in which the wizard opened the prototype file (in Director’s .dir format). A prototype builder was already provided, namely Director. (Pettersson, 2002)

Protoype construction is managed in the WOEB environment by the *WOz Builder*. “The builder can automatically organize the layout of the WOz Server GUI, so that the buttons are distributed over columns labelled by the request context.” (Bellucci et al. 2009). Again, this seems to deviate from the Ozlab thinking, even if it is a bit hard to judge the extent of the differences from a short paper. There is a difference between letting a tool generate the wizard’s interface rather than building the interaction scenes including the wizard’s controls, which should be placed right at the heart of where TP’s GUI actions take place. But as Ozlab’s Testrunner provides some ‘permanent’ wizard controls, perhaps the difference is not so big between the two approaches in this respect.

The work on WOEB was discontinued a few years ago (p.c. Bottoni 2014-05-07). A more complete description of a previous incarnation of the framework can be found in a technical report from the project (in Italian).

WozARd is a tool for WOz experiments on Android mobile phones, tablets and glasses that afford communication between the wizard and puppet device over wireless and Bluetooth. “It aims at offering a set of tools that help the test leader control the visual, tactile and auditive output that is presented to the test participant. Additionally, it is suitable for using in an augmented reality environment where images are overlaid on the phone’s camera view or on glasses” (from the Abstract of Alce, Hermodsson & Wallergård 2013). WozARd is location aware and accepts images, video and sound to be uploaded into the prototypes. WozARd logs test results and visual feedback on a SD card.[15] Content can be added during test. In Alce et al. (forthcoming) it is explained that "WozARD lets the user interact with the system through a SmartWatch". Thus, TP does not only see things through the augmented glasses but can also make some inputs to the system (i.e. to TL). This forthcoming article also gives data from a test with 21 test participants. WozARd was released as open-source in the summer 2014.[16]

A deliverable report from the 7FP EU project VENTURI ("immersive ENhancemenT of User-woRld Interactions") gives more details and contains screen capture for all the different modes of the wizard control, which are *Filebrowser*, *Camera* ("it can be started in the background without the test person knowing it", p. 11), *Puppet* (gives TL a view of the TP device), *Navigation* (to help TP in real world walking), *Notification* (sending different kinds of messages to TP), *Tours* ("With the tour function the wizard can trigger different actions on different locations.", p. 13), *Predefined sequence* (to allow click-through for the wizard when running a test session), and *Log*. (Alce, Hermodsson, Lasorsa, Liodenot, Michel, Razafimahazo & Chippendale 2013; see also https://venturi.fbk.eu/.)

WOZ PRO is an attempt to resolve the problem with certain low-fi tools being easy to use while not making testing easy, especially WOz testing which the developers of WOZ PRO seem to regard as the obvious method for low-fi prototyping. Hundhausen, Trent, Balkar, and Nuur (2008) find neither "simple art supplies (e.g., pen, paper, and scissors)" nor computerised supplies, such as PowerPoint and software specifically designed for low-fidelity prototype creation "to be optimized for the key, complementary activities of (a) rapidly creating a user interface prototype, and (b) running wizard-of-oz tests." (p. 86)

They identify two problems for these "existing technologies" for low-fi prototyping (ibid.):

1. "*Design change is cumbersome.*"
2. "*Running wizard-of-oz studies incurs a potentially high cognitive load.*"

For (1), the developers phrase the relieving requirement as the ability to "propagate a design change to other related screens". Indeed, we experienced from the use of our Ozlab, in its previous incarnation relying

[15] SD is short for "Secure Digital" according to *Wikipedia* http://en.wikipedia.org/wiki/Secure_Digital [2014-01-22] Official website is https://www.sdcard.org/home/ but it does not seem to explain the abbreviation.
[16] Cf. https://github.com/sonyxperiadev/WozARd

on Director and its film metaphor, that the use of sprites made it very easy to simply stretch graphical elements over as many frames as we liked (each frame could be made into an actual scene in a prototype or "interaction shell"; see Chapter 2 above). This made changes easy to propagate, and it has been a valuable feature, which is not as easy to replicate in the new, web-based Ozlab, as it lacks the "Score" overview of Director. However, being aware of the great utility of "propagation of changes" for a rapid-prototyping tool, we are extending the reuse of scenes and similar things.

For (2), Hundhausen and his co-authors made it easy to define a subset of scenes for the wizard to navigate to for each scene. This minimizes the number of alternatives for a wizard when running a test session. However, their system does not seem to support ad hoc scene transitions. (Good screen shots are provided in an earlier paper: Hundhausen et al. 2007.) Other possible wizard functions than navigation is not mentioned (such as hiding / making visible different objects including popups, which is a way to decrease the number of scenes dramatically).

In an experiment with 19 computer science students, the WOZ PRO developers found that these untrained designers tended not to use the "clone screens" and "propagate change" functions, and that defining scene transitions was cumbersome when the state chart grew beyond a single screen. It should be mentioned that WOZ PRO is pen-based because its developers' intention was to ensure that it is as easy to use for creating UI prototypes – cf. (a) above – as the three other low-fi methods mentioned. Their experiment notably showed some trade-offs between ease of navigation during a WOz test and providing navigation clues for the wizard. (The project web page is still available via http://helplab.org/projects/woz.)

It can be mentioned that in addition to the experiment with WOZ PRO, Carter and Hundhausen made a questionnaire at the CHI 2008 conference where they asked participants about how and why prototyping tools are used. They presented the results at the 2010 IEEE Symposium on Visual Languages and Human-Centric Computing. In their conclusion, they reiterate the need for "better scaffolding for interface simulation." (p. 211) But the main result was that the majority of practitioners still used "art supplies" for prototype construction of which more than half also used these for usability studies, while some 40% used graphics editing software and presentation software for usability studies. The authors sum it up thus:

> "Another key take-home message of our survey is that, even after over 20 years of research aimed at developing custom

> user interface prototyping tools, few, if any, user interface designers appear to be using these custom tools in practice."
> (Carter & Hundhausen 2010, p. 211)

At the same time many of the respondents cited "Cannot test complex interactions" as a weakness for all three of the simple methods (i.e. pen, ps, ppt). Carter and Hundhausen found that many practitioners were using two or more prototyping techniques. "We were somewhat surprised by the finding that the tools used by our respondents to create prototypes were not always the same tools that they used to test prototypes with users." (p. 210) But could one hope for a tool providing the exactness of Photoshop with the swiftness of pen and paper, and the interactivity of a multimedia product? Seeing the tendency throughout this literature review that the generic WOz tool has not been long-lived, this question merits further discussion, which will be a recurrent concern in the following chapters as well as in the immediately following section.

*

After the above literature review of WOz tools was made and the present working paper was more or less complete, a pre-print of an interesting paper by the WebWOZ group appeared (Schlögl, Doherty & Luz 2014, May). In it, they report from an interview survey with twenty researchers/developers using WOz (see our brief mentioning in section 6.5 and Chapter 7), as well as devoting a special section to "Existing Wizard of Oz tool support" including a subsection on "Challenges to generic tool support" where problems for re-usable systems, especially Dialogue Management systems, are discussed.

3.3 Requirements on generic WOz tools

Grill, Polacek, and Tscheligi (2012, p. 3), who presented ConWIZ (see above), compared 18 WOz systems and concluded that the following requirements should be met by generic tools:

> *Functionality:* A WOz tool shall provide support in all phases of a study, i.e. during the prototyping as well as conducting a study. ["Study" presumably refers to one experiment rather than a full development cycle from idea to a system launch.]
>
> *Flexibility:* A WOz tool shall be usable in multiple contexts. This refers to the flexibility of the tool which also should be applicable in mobile contexts.

Observation: The wizard needs to know what the user does. This can be achieved via proper real-time logging and data provision to the wizard as well as through the possibility of the wizard observing the user directly (in-situ or via video transmission). In addition, functionalities like screen capturing are advantageous if the WOz prototypes are based on a graphical user interface.

Configuration: In order to be re-usable in multiple studies the WOz tool needs to be adoptable and configurable in a multitude of situations.

Simulation. The WOz tool needs to be able to simulate functionality of the WOz prototype as well as to control objects in the study context. Regarding mobile and ubicomp contexts the tool needs to be able to control such remotely.

Collecting Data: The WOz tool needs to be able to record all the data required. The particular data depend on a concrete study scenario, which requires a flexible logging mechanism of data about the user interaction as well about parameters about contextual objects and situations.

Real-time Functionality: The WOz tool needs to be able to simulate scenarios in real-time.

Study Support. The WOz tool shall support the wizard during a study scenario by providing functionality such as setting the current participant-ID, note-taking, starting/stopping of the study session.

General applicability: The WOz tool shall be suitable for supporting WOz studies for multiple domains and contexts.

Usability: The WOz tool shall be applicable and appropriate to support human wizards throughout the whole study process. The usability of the WOz tool shall be good.

Perhaps the *Configuration* requirement should be named *Configurability*, but without going into such formal details of this list, it is noteworthy that most if not all of these ten requirements could be further divided into sub clauses. Now, a further division would make the list unmanageable if the systems gone through so far were to be tabulated and checked requirement by requirement. Grill and co-authors actually used a table to compare the eighteen WOz systems they considered. However, already that table is marred with exceptions and footnotes. In contrast, the present work puts the details in the text while also trying to indicate the major 'historical' trends – if the expression may be allowed – of moving away from focusing solely on natural-language processing, the acceptance of multimedia output and multimodality input, the embrace of service robots and lately of smaller mobile units that the user carries around.

In order to evaluate the requirements put up by Grill et al., it should be noted, that for the input to the wizard there is no need – in 'first iteration' – to delimit the modalities to what the future system should detect. The same goes for the output from the wizard: even if the WOz tool itself should help the wizard to produce the output envisaged for the future system, the tool itself does not have to constitute the whole experimental setup for each test.

From such reasoning the next chapter will deal with wizard production of some admittedly limited but nevertheless very basic forms of output, thus providing a framework for requirements of this most essential part of any WOz system. Here, some notes will be made concerning output production and WOz simulation in connection to Grill's list of WOz tool requirements.

To start with our own system, the original idea behind Ozlab was to find a way to make *GUI articulation* swift in order to allow more dynamically created man-machine dialogue (with faked machine expressions, as Ozlab is made to facilitate Wizard-of-Oz experiments). Notably, an "articulator" differs in scope from a WOz tool that provides "support in all phases of a study, i.e. during the prototyping as well as conducting a study" (Grill's *Functionality*); there is not necessarily a drawing tool or a logging tool as long as UI-specific dialogues can evolve relatively unplanned, that is, by letting the wizard use appropriate UI widgets to make the test participant understand what the UI is intended to convey.

An interesting aspect surfacing in some of the papers reviewed above is the concept of a configurable user interface for the wizard (e.g., ConWIZ, WozARd, and WOEB above). Probably this reflects an idea of the wizard tool as any other machine with a UI for its operator. In contrast, the development of Ozlab centred much on the interaction space for the TP. Hence the TP screen became much of the TL control window, which thus was unique for every test except for some general wizard functions as explained in Chapter 2. The development challenges were never put as making the TL-UI "configurable". Surely, some helpful features were implemented; but that the TL-UI should not be built for every shell (i.e. prototype) was never on the agenda as Ozlab was made as a GUI articulator for TL. (See also comments made above in section 3.2 on WOEB.) This means also that if several TLs are using the same shell, they can adapt on-scene widgets before their own test sessions to fit their own preferred workflow. Li's evaluation of different factors affecting consistency in WOz testing found that "flexible layout design assisted [wizard] to build up an efficient work space which fitted with personal operation preferences" (2012, p. 159; Li & Bonner, 2011).

The use of the word *Simulation* in Grill's requirements list is not wholly well chosen. Surely, a wizard simulates something when conducting a demonstration or test, but it is only system internal functions that are simulated. In contrast, the extrovert actions of the mockup system will have to be real. Indeed, they often have to look as real system output, at least when WOz is used to conceal the true nature of the noesis. In any case, the output actions have to be real in the sense of being perceptible for test participants. Other words fit better for the mock system responses, such as "production", "generation", or "output control", or, as indeed preferred by Pettersson (2003), "articulation".

To support simulation (when this word is used in its proper sense), the wizard(s) will have to have good overview of the doings of the test participant(s) – however, there is a problem if the wizard has more input than the future system can possible have. For instance, mood recognition (by the human wizard) can be realistic in a futuristic WOz, but it can also be very helpful in a simple GUI-based pedagogical product – however, this very helpfulness will give the wizard more input than the simple GUI-based application can have once it has been programmed. On the other hand, this cheating can be very useful in 'first iteration' as noted above. For instance, in the very first use of Ozlab in 2001, the experiment run with special educator progressed for several months by drawing the blind to the test room and as a final step there was the possibility to shut off the microphone in the test room. And long before this study commenced, the special educators had already developed their training materials consisting of laminated, coloured hand drawings and methods of engaging the small children as well as their parents as tutors to the children. Thus, the tapering of the communication channel, as Mavrikis and Gutierrez-Santos (2010) call it, could be a planned working procedure and extend far beyond (far *before*) the initial steps of the prototype(s) for a system.

Reflecting further on Grill's and co-workers' requirements, we can notice that there are some main objectives behind the requirements. We identify four goals:

Props Construction (Part of *Functionality*, namely construction of "prototype")

Broad Applicability (*Flexibility*, *Configurability*, and *General Applicability*)

Production of Output (*Simulation* and *Real-time Functionality*)

Protocol Support (*Observation*, *Collecting Data*, and *Study Support*)

The first part of *Usability*, i.e., "the WOz tool shall be applicable and appropriate to support human wizards throughout the whole study process", seems to repeat the idea in *Study Support*, so it is left outside this grouping, while the second part, "the usability of the WOz tool shall be good", is desirable but perhaps not necessary if professional designers use the tool to trigger some responses they cannot have otherwise before there is an implemented version (in contrast to large-scale scientific experiments where the usability is more urgent to avoid wizard fatigue, but then one might wonder if WOz is the right method). The swiftness of the **Production of Output** is furthermore dependent on how much work is put on the wizard who is dependent on how the mockup and setup is designed in each case; thereby, this swiftness is not directly dependent on the tool itself. A standard usability requirement is probably far easier to employ on the **Props Construction** functions of the tool rather than on the wizard functions.

The next chapter will concentrate mainly on the **Production of Output**. The chapter following will focus on the question of platform dependency. This issue is related to the question of the longevity of a generic WOz system, which was raised in passing several times in 3.2. Slightly platform-averse, we adopted web technology for the new implementation of Ozlab despite problems in **Props Construction** and even in the timely and quick **Production of Output**, as mentioned in Chapter 2 and as will be further discussed in Chapter 5 and also in section 6.5, "Delays and time lag".

4 How interaction is supported by the WOz tool

The WOz technique began as a way to explore and develop systems in an area where technical limits were prominent, namely NLP, natural language processing. When NLP researchers found that the WOz technique had to adapt to the emerging GUI standard, the WOz method became more cumbersome as specialised GUI systems had to be programmed even if the idea was to avoid programming the NLP functions to be explored. In addition, there was a lack of opportunities to do GUI experimentation in the WOz way. This was the reason for the development of the generic, graphical WOz system Ozlab (Pettersson 2002). Similar thoughts seem to have resurfaced in the last decade, hence the development of several re-usable WOz systems.

The following subsections attempt to characterise the various aspects which were important for the original Ozlab and the new web-based variant. The aspects overlap in some cases or are subordinated, but for the sake of clarity, each aspect will be discussed within its own section, namely:

1. What output is supported by the WOz tool?
2. The continuum from demonstration over explorative testing to evaluating design proposals
3. Explorative WOz: open response space for the wizard (that is, insights [hypotheses] of the moment which can be "tested").
4. Exploration by short interruptions to make changes to a prototype during a test session
5. Web features in WOz experimentation.

So the first issue is about the wizard's role, but limited to the materiality of the output. It is not about input; it is not about the wizard's ability to get various sorts of input, because this could be made experiment-dependent and is not strictly about the tool. For instance, God-like properties such as seeing how the test subjects feel by watching a video monitor or peeping through a mirror glass are independent of the WOz tool used – a video camera and a monitor can always be externally added to any tool, or the tool can be placed in a usability lab with a one-way window between the control room and the test room.

Issues 2-4 deviate from the perspective of many WOz papers by not focusing solely on the test-as-verification benefits of faking the implementation of interaction design. Issue 2 (and by implication 3 and 4) discusses WOz as Kelly once did, namely from a systems development perspective. Test as exploration lends high degree of openness to the wizard, in terms of the person's capacity as a human being to understand the requirements during a session of interaction. Issue 4 is related to both of issues 2 and 3, but it puts an extra dimension to the exploration perspective, viz. modifying the prototype after an interactive session has started.

Issue 5 targets a vulnerable point of the generic solutions, namely the possibility of being able to run them in the context of an ever-changing infrastructure where operating systems and other features vary and are not backward compatible. At the same time, the World Wide Web supports compatibility and general executability so the effect of web features on WOz experimentation merits special attention.

4.1 What basic actions is the wizard supposed to execute?

There is a conceptual difference between what the wizard is supposed to simulate and what kind of output the generic WOz tool supports. The wizard can, for example, be supposed to simulate the system's interpretation of a person's body movements, as in Höysniemi, Hämäläinen, and Turkki (2004), but the output that the wizard provides the user with is graphical: e.g., if the user moved as if he/she was swimming the wizard showed a "swimming" avatar. Thus, the wizard provides the correct output and does not simulate it. What is simulated is instead the system's recognition of bodily movements.

In this section, the intention is to show the wizard's output possibilities in the generic WOz tools rather than what the wizard is supposed to simulate. Simulation might concern such things as the recognition of body movements, of body position in relation to a physical object, of speech, of written language, of a click on a link, of a mouse-over, of a swipe, etc. After such recognitions the wizard has to produce some output, and this is what this section will discuss, because the reusability of a WOz setup depends on the possibility to do "any" kind of system response rather than only those specific to a certain experiment.

However, the literature on WOz experiments makes clear that there is a difference between a mockup and the total experimental setup. Example: in the first experiments conducted with the Director-based Ozlab system, a separate microphone and a voice disguiser were used, thus allowing the wizard to give the user audio responses that purport to come from the characters on the screen. It must be understood that certain experiment setups provide the wizard with output possibilities outside the generic WOz tool.

Below we develop this framework based on the smallest parts needed to articulate a user interface (mainly for standard applications on standard devices). We divide the elementary articulations into three main types: visual, audible, and actuationary.

4.1.1 Visual output

SCENE MANAGEMENT

Graphical interface changes: Displaying or switching between images/scenes/pages in the mockup as in Ozlab and in the Android tool by Linnell, Bareiss and Pantic (2012).

"Narrative" as in Lee, Mott and Lester (2010; see sec. 1.2 above), that is, the wizard opens up further parts of a programmed system (bringing the user to the "next level", so to speak).

GRAPHICAL OUTPUT

Pre-recorded animation/video as in the systems mentioned for scene management. Live video does not seem to be included in any system except as captured by the TP device (WozARd by Alce et al. 2013). Switching on the video channel might be the wizard-part of the output.

Beautified user sketches as in SketchWizard by Davis, Saponas, Shilman and Landay (2007).

Make an object visible/invisible. Can be used for notifications, alert boxes and a range of other features such as blinking for attention.

Repositioning of objects visible to the test participant during a test session.

Drag-and-drop with visible drag in order to make (simple) animations without pre-recording or pre-programming (see TownMap in Bergmann et al. 2006). This function and visibility (hide/show) were two of the essential requirements for the first Ozlab system as it made it possible to WOz prototype GUIs in the multimedia area.

AUGMENTING OUTPUT

For prototyping so-called augmented reality applications, WozARd gives the wizard the possibility to control the "augments" (at least to select among pre-installed system output; Alce et al. 2013). For the development of our framework, the question is if the output itself differs from the other things listed here (like video + make an object visible); the augmentation presupposes TP video and possibly, the wizard can switch it off or hide it.

TEXT OUTPUT

Choose from pre-written texts as almost all systems seem to allow. This facilitates speedy execution and correct spelling.

Generating sentences by assembling text from pre-written words or phrases, which Dahlbäck et al. (1993) found useful to make text output correctly spelt and grammatical. Also MDWOZ allowed only selection during a running session.[17]).

Wizard-written text as in Ozlab with its emphasis on explorative interaction, and a few other systems. DiaWOz-II (Benzmüller et al. 2007) combines free-writing with selection of standard words and mathematical symbols, and the output is put in the right place by the system.

Re-using input from the Test Participant: this is standard in NLP experiments but often fully automatized. For setups relying on speech recognition or machine translation, there is both selection and editing and it mixes with the interpretation task of the wizard (see Fig. 9 in Schlögl, Doherty, Karamanis, Schneider & Luz 2010). In Ozlab the inclusion of TP inputs in certain output, e.g., summaries, is possible but depending on the design of the shell (prototype). The output is automatic but it can be seen as wizard-generated as an Ozlab wizard can always choose to make it visible or not. The input included in text outputs can consist both of TP-typed text and of selections made by TP by lists, checkboxes, and radio buttons, as was explained in section 2.2.1.

[17] In fact, whether MDWOZ allowed only selection or also composition is hard to say on the basis of the paper by Munteanu and Boldea (2000) but in an email letter Munteanu explains: "the wizard could not edit the output once it was generated, but the interface for managing the dialog states was designed to be reconfigurable, and the wizard could easily add new template texts to each state. The output interface had the technical capability to allow the wizard to edit the output text, but if I remember correctly I was worried that this would overload the wizard's tasks and also run the risk of out-of-vocabulary items for the TTS system." (p.c. 2014-07-27)

NUMERIC OUTPUT

Numbers can be seen as text, but they have a correctness standard that is of a special kind: making a correct statement of something measurable. Topiary calculates correct real-world distances between graphical objects the location of which the wizard controls (on top of a map in this case; Li et al. 2004). Sums would constitute the prototypical example but this fact was not mentioned in the literature reviewed.

Displaying time digitally could also be viewed as a textual output. However, a timer, especially a timer displaying seconds, has a correctness standard that is of a special kind as it could quickly be noticed by the test participant if the time-keeping is simulated by a human. The same holds for an output displaying a clock.

4.1.2 Audible output

SPEECH OUTPUT

Text-to-speech (TTS) is speech composed synthetically and controlled by the wizard by typing or selecting text, such as “Departing to London at 10:45 AM.”

Direct speech (a wizard may be reading a manuscript as in DART by Dow et al. 2005; in Pettersson 2002 the wizards used voice disguiser to conceal from the test subjects that the test leader were acting as characters in their animated training exercises). This allows for exploring new responses during a test session, but voice does not seem to be inherent in any of the mockup controlling software.

Pre-recorded speech responses as in SUEDE (Klemmer et al. 2000). Any system allowing for playing video-snippets on wizard’s demand can of course also provide pre-recorded speech output.

Re-using input from the Test Participant: we are not aware of this being made in any system by simple recording. In some spoken-language systems, there is of course a transformation to machine-readable text which is then used in speech production (but in WOz setups it seems there has always been a wizard selecting among possible outputs before one text is read aloud by the system).

SOUNDS OUTPUT

Pre-recorded music/notifications/sounds. Can be realised as specific controls for specific sounds or by pre-recorded sounds as in the paragraph immediately above.

4.1.3 Actuator output

TACTILE OR SENSORY OUTPUT

Tactile or sensory output as in and Ozlab (vibrations on Android devices). These are directly directed to a specific TP.

CONTEXT PARAMETERS

Switching on/off e.g. lights but also varying intensity; Grill et al. mention the fan they switched on and off (2012; see also Zachhuber et al. 2012, p. 227, who mention light intensity but also briefly mention "all human sense […] and olfactory sense").

MOVEMENTS OF MACHINERY PARTS

Movements of mechanical limbs are more complex than just switching on and off: there is a risk that other objects or the TP or other people interfere with the movement. There are situations where the precise coordination belies any attempt to bring it down into more elementary output forms.

MOVES OF MOBILE OBJECTS

Controlling the moves of i.e. service robots is again more complex than controlling switches as the constituent x- and y-axis movements have to be coordinated precisely and in concord with the speed adjustment. (Riek 2012)

4.1.4 Examples

Here we give only a few examples of how different kinds of wizard output can be used together in a system:

In Topiary by Li, Hong and Landay (2004) the wizard can indicate location and spatial relations in prototyped applications by using a number of the above listed outputs such as: graphical output by managing scenes, showing objects, and reposition objects; numeric output by showing distances between objects, also time can be shown; and textual output.

WozARd by Alce, Hermodsson and Wallergård (2013) supports several wizard-triggered outputs, especially AR. The wizard can generate/activate visual and audio output on the user device, such as default or wizard-created notifications (SMS, etc.), audio by TTS, visual navigation instructions by, e.g., navigation arrows and by showing images depending on the user's location.

In MDWOZ the wizard can simulate the speech recognition and semantic analysis functionality of the system, and provide TP with a text and/or text-to-speech response to TP's utterances. In their reported experiments, however, Munteanu and Boldea (2000, p. 106) used pre-recorded words "to synthesize the speech signal after the text is normalized".

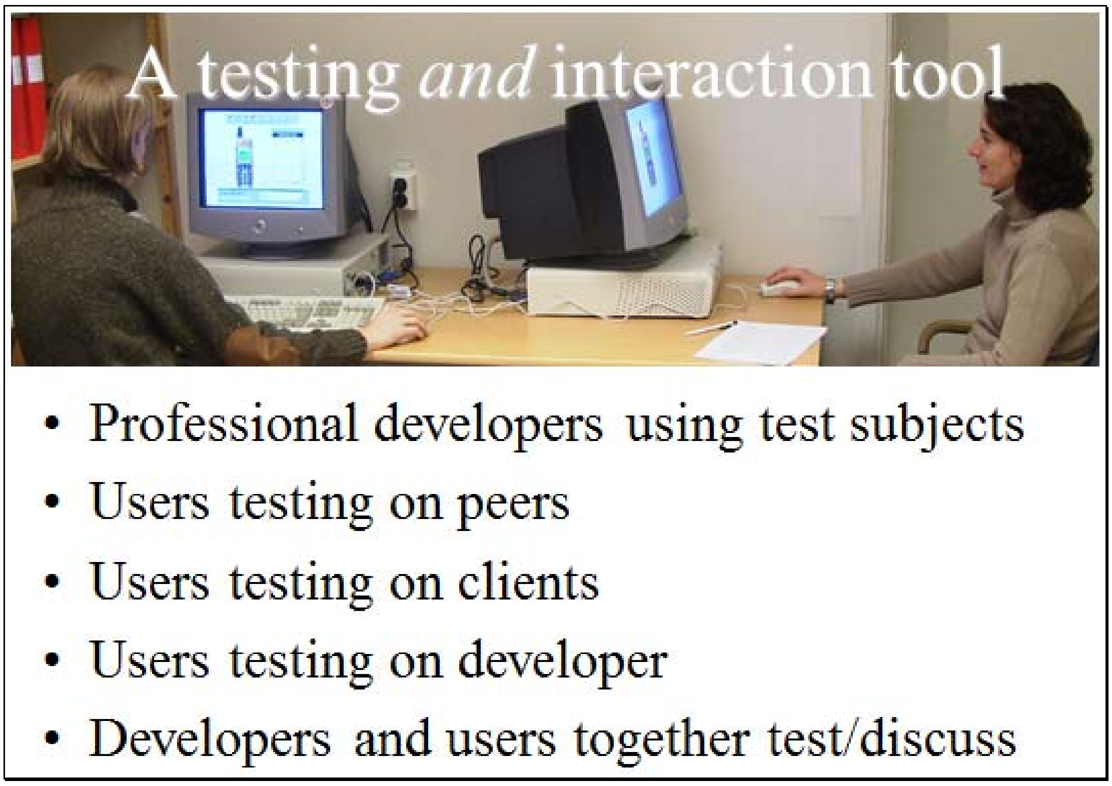
4.2 The Continuum from demonstration to evaluation

A Wizard-of-Oz prototype can be used within project groups to demonstrate solutions. No papers are discussing this aspect of the generic tools. Being able to demonstrate the interactive aspects of the proposed solution should add insights to the development team and the other stakeholders even if it is obvious that it is a faked interactivity. In fact, there is a whole spectrum of various constellations of the use of a WOz prototype from demos to final evaluations.

An experimental setup that is in-between the evaluation and explorative setting is the one where the "test participant" is fully aware of the wizardry. Such a setup has been used in Ozlab by using two adjacent PCs. It has also been used with connected laptops. The wizard and the "test participant" (a content expert in the project group) then sit next to each other, exploring and discussing the mock-upped system, as in the photo in Figure 4 (connected laptops have also been used for real testing, esp. by student groups).

New insights can also be gained by letting one of the project group members act as wizard or as TP. In fact, there are many combinations. Pettersson (2002) reported on the special educators acting as TLs with their clients (children and parents) as TPs. On one occasion, Pettersson acted as TP to demonstrate to the educator some aberrant behaviours which users sometimes engaged in and that she had to think of how her "computer program" should respond to. The list in Figure 4 gives some major constellations:

1. Professional developers use test subjects
2. Users test on peers
3. Users test on clients
4. Users test on developer
5. Developers and users together test/discuss
6. Developers and content experts together test/discuss (see the photo!)


A testing *and* interaction tool

- Professional developers using test subjects
- Users testing on peers
- Users testing on clients
- Users testing on developer
- Developers and users together test/discuss

Figure 4. A PowerPoint slide used by Pettersson since 2003 to present the multi-faceted use of a WOz tool

Evaluation itself is not further discussed here, because it does not differ from other iterative design approaches but a note on teaching usability testing can be worth making. As Ozlab has often been used by beginners (students) over the years, the recording of wizards or wizards' screens has been a commonplace. Such wizards have always been designers to various degrees but of course we have noticed weaknesses in notations from such sessions. It is not uncommon that students believe they can remember their thoughts also after the test session is over. Similarly, when a user representative is TL while another user representative acts as TP one might wish to rely on more than the wizard's memory for making post-test annotations; to compensate, the wizard's *runtime* comments may constitute good memory aids if there are no experienced designers or usability testers available when sessions are run. Such comments can be made orally and recorded together with the screen capture of TP's or TL's screen, provided that the TP cannot hear the comments.

4.3 Explorative WOz

Conducting tests as exploration leaves a high degree of openness to the wizard by respecting the wizard's capacity as a human being to understand the requirements during a session of interaction. This means that in explorative WOz sessions the script for the wizard (i.e., the interaction scheme or response scheme) can be of three kinds:

1. non-existent;
2. existent but subject to further development; or
3. existent but can be deviated from.

By explorative WOz tests the design team is developing the script as tests are conducted. In order to conduct experiments without a pre-conception of exactly what the interaction should look like, or what responses the wizard should give, the experimental setup must allow the wizard to interpret the test participant, and the WOz tool itself must provide a possibility for the wizard to come up with responses on-the-run. For exploratory WOz, the tool should enable the wizard to adapt the prototype to the user's comments, responses and input, in a quick and easily conducted manner.

In earlier experiments conducted with the Director-based Ozlab, it has been shown that Ozlab facilitates development of the interaction script by enabling alterations of the interface or the interaction patterns in the prototype according to the user's actions or preference. Figure 5 below was an early attempt to illustrate how the Director-based Ozlab system was used in this manner, as it intends to show how the interaction script as well as the interaction shell is developed through a series of iterations.

As cited in 1.2, Mavrikis and Gutierrez-Santos in their work on intelligent tutoring systems acknowledge a much larger development cycle where pre-prototype interaction between facilitator and learner is also included. Similarly, in the first Ozlab study the test wizards used exercises from their ordinary work as was mentioned in 1.1 and 3.3.

In addition, it can be noted that the explorative ideas that the wizard comes up with during a session may go outside the limits of the WOz mockup. In such a case the designers have to change something in the test setup in order to carry through the novel ideas. It could be changes in the file(s) that the generic WOz tool runs during that test session. This kind of changes is discussed in the following subsection.

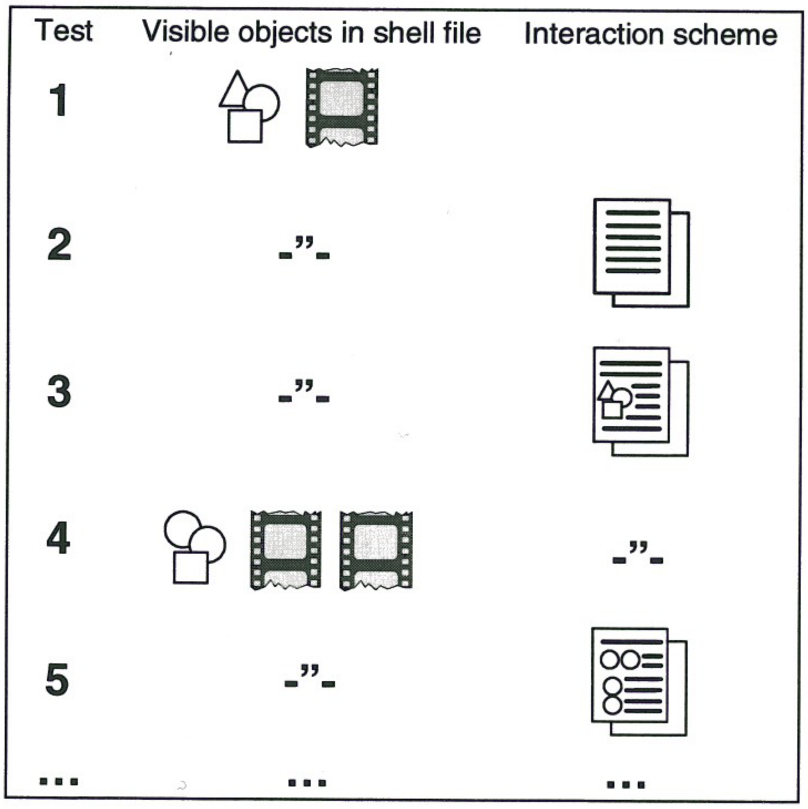


Figure 5. The evolution of an interaction shell and its concomitant interaction script during five iterations (source: Pettersson 2003, p. 166).

4.4 Exploration by short interruptions to make changes to a prototype during a test session

For the generic tools listed earlier it is often not clear whether they would facilitate explorative testing that goes as far as interrupting a session to make alterations to the prototype (apologizing oneself to the test participants, of course) and then continues without ruining the test session. For instance, WozARD allows for runtime improvements of the responses (3.4.3) as apparent from the following statement: "It is easy to add content and create lists of notifications without recompiling the application." (Alce et al. 2013, p. 602) However, it is hard to see what labour would go into adding new scenes if it at all is possible without recompiling the application.

The same holds for "A Wizard of Oz tool for Android": "Once all of the files have been added to the project, the tool generates a folder containing all the images and videos underlying the prototype that is manually loaded onto the phone's SD card." (Linnell et al. 2012, p. 67) However, the tool seems to support to some extent that changes are made to the prototype during runtime: "The screen transitions and drawing of widgets are all pushed from the laptop to the phone at runtime […] This means that the experimenter can alter widgets or transitions during an experiment, and the project does not have to be re-loaded onto the phone." (ibid. p. 67)

In MDWOZ it seems as if the changes were done *between* each session (that is, neither during runtime or by pausing): "During the simulations period, the interaction model was changed, usually after a dialog session in which the subject asked a valid question, but there were no paths through the dialog graph which allowed answering it. This way, the interaction model size increased from 23 to 32 states. Besides adding states, in certain cases states were removed, and texts associated to wizard nodes were changed." (Munteanu & Boldea 2000, p. 170)

In Ozlab the wizard can either pause the test (by the *Wait* screen or by freezing the TP screen) and apply changes, or terminate the session to apply new objects to the interaction shell, and then restart the session. Adding completely new objects in runtime is not supported in the present version of the web-based Ozlab system (nor was it supported in the earlier Director-based system), but would support conducting explorative tests even further. For the purpose of exploring a multimedia chat within a web site, Malin Wik is experimenting using two browser windows, so that she as wizard can stop sessions, add content, and start sessions in one window while the test participant seemingly seamlessly continue probing in whatever window is available (the preliminary ideas are found in a poster presentation; Wik 2014).

4.5 Web features in WOz experimentation

This chapter has analysed the generic WOz tools' capacity to support WOz sessions, "production" in particular as well as the management of sessions in general not from the viewpoint of how log all possible data but from the viewpoint of ensuring an interesting interactive chat between TL and TP. Admittedly, the latter aim often recurs in usability and corpus-generating works, but it is of secondary importance in a general treatment of Wizard of Oz as this technique can be used as in 4.2 – 4.4 where the aim is primarily to enlighten the designers (even stakeholders) to come up with new ideas. Video capturing of everything involved may often suffice for rehearsal and debates rather than a detailed tagging of data files.

To round off, in the perspective of the present work, the discussion of the problem of sustaining generic WOz tool in this chapter and the discussion in the next chapter on web-based WOz tools, it may be appropriate to briefly mention the major impacts of a webified solution.

Sluggishness: several things make communication via the web rather slow. Without a direct link to TP's unit, this will impact on TL's ability to quickly notice what TP is up to and it will of course impact on the time it takes for TL's actions to take place in TP's device(s).

Scrolling: in a GUI-based WOz-experiment where (much) scrolling is involved, there will be a problem monitoring and controlling the TP pane and also managing TL's own pane.

Surfing: as many resources are on the web, it is on many occasions natural to let TPs access other sites. This can entail some problems to maintain control of the experiment.

Browsers: restricted access to assets on the TP device limits what can be tested. Also the problem of easily setting a browser in web pane only mode limits what can be tested.

While the latter two can be solved, the first two are harder to overcome. Especially the one on transmission speed is crucial: there is no experimental setup where it will not matter.

5 Platform-independent WOz tools

As said in Chapter 3, the present work has indicated some major trends in the history of the evolution of the Wizard-of-Oz method: moving away from focusing solely on natural-language processing, the acceptance of multimedia output and multimodality input, and the embrace of service robots and lately of smaller mobile units that the user carries around. Naturally, some of these directions partly overlap and the idea of very general WOz tools suggests itself. On the other hand, the 'ad hocity' of the method is lost if a research team is to develop an all-covering tool. From one of the Grenoblers, to whom we sent a draft of this text inquiring about any use of generic WOz tools in her group after NEIMO, we received the following comment:

> "We still develop Woz-based tooling on a case per case basis (as opposed to NEIMO whose goal was to be as generic as possible for the study of multimodal interaction). For example, 3 years ago, we have developed a 'just-good-enough' Woz-like component to simulate typical situations in the context of end-user programming for the home, situations that are difficult to produce in real world scenarios because of middleware malfunctions, or because the hardware and services (e.g., sensors/actuators, weather forecast) are not integrated yet. […] In these days, I do believe that Woz tooling is necessary but it has to be highly tailorable to the domain ('ad-hocity' is a must)."
>
> (Joëlle Coutaz, p.c. 2014-05-28)

With this pertinent caveat as a prologue we run three discussions below on (1) tools for development for different platforms, (2) threats to generic WOz tools, and (3) web-based tools respectively. A fourth section attempts to present an overview of (4) possible limitations of web-based WOz tools.

5.1 WOz tools for tests on several platforms

From 1.2 and 3.2 one can see that there are many systems and tools developed to support using the Wizard-of-Oz technique in the development process of a system. As the demands for using systems on a large variety of devices increase, Segura and Barbosa (2013) argue that the prototyping of such systems should follow the demands. The authors furthermore argue that even when using responsiveness, the user interfaces need to be designed with different screen sizes, resolutions, input techniques etc. in mind. However, all WOz systems are not totally generic. Many of the tools are designed for testing prototypes on a specific platform (see, for example, the WOz tool for Android by Linnell, Bareiss & Pantic 2012).

Some steps towards a WOz prototyping tool for multiple platforms and devices have been made: Segura and Barbosa (2013) proposed an evolution of their prototyping tool UISKEI in this direction to support the Wizard-of-Oz technique and multi-device prototyping; in SUEDE (Klemmer et al. 2000) the prototype is generated as HTML, perhaps enabling tests on other devices than computers; Ardito, Buono, Costabile, Lanzilotti, and Piccinno (2009) developed MuMoWOz for testing multimodal systems on mobile phones or desktop computers; and Alce, Hermodsson and Wallergård (2013) present WozARd for WOz experiments on mobile phones, tablets and glasses – and this group at SonyMobile in Lund is about to release it as open source.

The tools might support tests of prototypes on different devices but it does not mean that all kinds of devices are compatible with the tool. For example, Segura and Barbosa state that they are developing UISKEI++ for Windows 8 as well as Android devices, implying that running prototypes on Apple's products is excluded. In re-developing Ozlab as a web-based tool, we have found the same problem, because even if the intention is that it can run on a (Chrome) web browser on any device, there are different restrictions in different mobile operating systems as to what hardware can be accessed from a web browser. There is also the problem of the nativeness of the browser engine – Apple and Google have used different engines for their browsers since 2013. These and other aspects of basing a re-usable tool on web technology will be brought up later on in this chapter.

5.2 Threats to generic WOz tools

A shared issue amongst the generic WOz tools is that they are vulnerable in a longevity perspective. The intention in this subsection is to present different aspects of this perspective, by making notes on how a tool can be affected by changes in the system environment.

Tools based on a program

A WOz tool that is based on a specific program is at risk if the program is either updated or outdated. The updates of the program can end in a mismatch between the WOz specific functionality and the input/output that the underlying program will accept. If the underlying program gets outdated, it may become impossible to run, install, etc., as the program is not supported by the distributor.

For the Director-based Ozlab system, also described in section 2.1, this became an issue as Macromedia Director was acquired by Adobe and not further supported.

Tools depending on a specific programming language

Vulnerability can be an issue for WOz tools that depend on a specific programming language. One example is that of the Director-based Ozlab system, which was dependent not only on Director, but also LINGO, the programming language used in Macromedia Director. A programming language will decrease the chances of further developments if it is not commonly used.

This issue may be related to all generic WOz tools presented in section 3.2.

Tools adapted for tests on a specific platform (e.g., Android devices)

Some tools are adapted for running tests and prototypes on a specific platform. Such tools make use of specific possibilities granted by the platform or are adapted to specific limitations of the platform. For example, if conducting tests on handheld devices that make use of touch input, the tool may be adapted to forwarding such inputs (if it is hard for the wizard to follow TP's actions by direct observation).

Adapting the WOz tool to a specific platform, however, makes it vulnerable regarding its durability. If the platform is redesigned in some way, e.g., what input forms it allows or what output it enables, the generic tool needs to be adapted to the redesign.

In tools such as WozARd by Alce et al. (2013), and the Android tool by Linnell et al. (2012), this vulnerability could be an issue, but working with the platform developer puts the longevity issue in quite a special perspective.

The very recent discussion by the WebWOZ team on challenges of generic tools is worth mentioning here (Schlögl et al. 2014) as they identify factors inhibiting re-use: in some cases the integration of a WOz tool in a larger system is the cause, and for other applications the WOz setup has been merely a throw-away prototype. Hardly any system has been directly available for download.

5.3 Generic web-based WOz tools

Schlögl, Doherty, Karamanis, and Luz (2010, p. 113) argue that "[…] 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."

As argued by Schlögl et al., a web based Wizard-of-Oz system enables remote experiments to be conducted. If the system furthermore is accessed and run in a web browser, the setup difficulties and dependency to a certain platform is decreased:

"Existing WOZ and DM [Dialogue Management] 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.)" (ibid.)

In order to conduct tests with the previous Director-based Ozlab system, especially if installing it on a new machine, a cumbersome environment setup procedure was necessary. Several steps were needed to set up Macromedia Multiuser Server (see section 2.1). In order to allow the communication and writing/reading of files between the two computers used in the tests, firewalls and sometimes sharing settings had to be modified. The Director-based version was furthermore platform- and software-dependent: it ran on computers with Windows installed and needed Macromedia Director 8.5 or MX as well as Multiuser Server.

The web-based Ozlab system is less platform-dependent than the Director-based version. If the web-based Ozlab system is running on an IIS 8 web server, the wizard only needs a computer with the web browser Google Chrome installed to access and use Ozlab.

The web-based Ozlab system is intended for experiments on graphical interaction, just as the old Direct-based version was. Even though the web-based Ozlab does not support integration of language technology components as did, e.g., WebWOZ (Schlögl et al. 2010; 2013; 2014) or is specifically developed to support design and tests on speech user interfaces as was, e.g., SUEDE (Klemmer et al. 2000), experiments on speech interfaces in Ozlab can be conducted by using a microphone and/or voice disguiser (we have already at several occasions referred to the very first experiment with the first version of Ozlab; Pettersson, 2002). However, this would be a setup outside of the actual web system.

One could argue that Ozlab prototypes are runnable on every device that can run a modern web browser. However, this statement is not entirely true as noted above in 5.1. The web-based Ozlab system is software-dependent when it comes to which browser that renders the system properly, namely Google Chrome. In addition to Android devices, Google Chrome is available for iOS devices but for iOS devices, the web browser engine[18] WebKit must be used. Both Chrome and Safari used to run on this web browser engine, but since April 2013 Chrome instead uses Blink,[19] while Safari continues to use WebKit.[20] This means that even though the look and feel of a Google Chrome browser is the same on iOS devices as on Android devices, the interpretation, rendering, and display of mark-up language and formatting information are executed differently.

In addition to the Google Chrome dependency, the framework used for the interface of the web-based Ozlab system is based on Sencha Ext JS. In addition, it depends on IIS 8 with web sockets. These software dependencies can make the web-based Ozlab vulnerable to the issues brought up in 5.2. (Schlögl et al. report that "The WebWOZ platform has been implemented using the Google Web Toolkit which supports the construction of web interfaces using the Java programming language." 2013, p. 514; op. cit. also provides data on components.)

The next section attempts to present an overview of possible limitations of web-based WOz tools. The problems mentioned are derived from the experience with the on-going development of the web-based Ozlab presented in section 2.2.

---

[18] Web browser engine can also be called 'layout engine' or 'rendering engine'. See Wikipedia on *Web browser engine* http://en.wikipedia.org/wiki/Web_browser_engine [2014-02-28]

[19] The Chromium Blog (2013-04-03) *Blink: A rendering engine for Chromium.* Available: http://blog.chromium.org/2013/04/blink-rendering-engine-for-chromium.html [2014-02-28]

[20] CNET, Stephen Shankland (2013-04-03) *Google parts ways with Apple over WebKit, launches Blink.* Available: http://news.cnet.com/8301-1023_3-57577790-93/google-parts-ways-with-apple-over-webkit-launches-blink/ [2014-02-28]

5.4 Limitations of the present web-based Ozlab system

Thus, in addition to the platform dependency just mentioned, this section will bring up some points presently straining Ozlab experimentation.

1. Sluggishness

Ozlab has slow response times. Regarding ConWIZ, Grill and Tscheligi say in a recent paper: "Future work includes enhancements of the ConWIZ protocol towards the possibility to use UDP communication for specific purposes." (2013, p. 441) In the Director-based Ozlab UDP was used because TCP had quickly turned out to be too slow but that was many years ago.

Furthermore, as the Shell Builder is also a service provided by Ozlab, making a prototype lends it the languorous feeling of a typical cloud service.

2. Connectivity outside the Internet

As the new Ozlab system is accessed through web browsers over internet connections, the connectivity could be an issue when using Ozlab. If experiments are conducted in environments where the connectivity is less reliable (or non-existent), one solution could be that of running the server locally. The TP device would then connect to the wizard's computer through some local connection. Preliminary data show that a laptop is noticeably slower to run the Microsoft server, while downloading heavy pictures can be speedier as there is no congestion to struggle with when there is a local WiFi used only by TL and TP.

3. Security

With a web-based Ozlab system security issues follow as well. One needs to protect the system and the network on which the system is running.

However, using firewalls also limits the access possibilities for "friendly" accesses. This will be an issue if access is granted through specific ports, and the same ports are blocked in the network from which the wizard is connecting.

4. Limited TL access beyond the web browser

Apple's restricted policies for access to system functions from web browsers will make it hard to run more elaborated version of Ozlab on iOS devices.

In general, the restrictions that operating systems set on web browsers (to prevent malicious web sites from overtaking a user's computer) pose problems for simple web-based WOz tests.

5. Browser controls and out-of-scope browsing

If the mockup runs in a web-browser, the experiment can be revealed if the browser-specific controls and panels are not hidden. The easy and available means to hide them is to run that browser in full-screen mode. A full-screen browser, however, makes it harder to situate a prototype within an environment with functioning applications.

Moreover, if the controls are not concealed, the user might navigate to different websites other than the experiment web page (this is further described in section 2.2.2).

6. Intended lack of responsiveness

By the design of Ozlab, an interaction shell does not automatically recognize the size of the TP web browser window. This means that interaction shells built in Ozlab are not responsive (Marcotte 2011) and do not adapt to the screen size of the device on which the interaction shell is running. Instead, if the content of the interaction shell is too big for the TP window, then scrollbars show up in the browser. This means that when designing and building the interaction shell, the designer must take the screen size into consideration. If the system had adapted to different TP device sizes automatically, the wizard would lose control over how the design is displayed. Such automatic adaptation makes the comparative testing of several design solutions hard and defeats the whole purpose of conducting WOz tests. On the other hand, the lack of adaptation might make for some problems if each test session is to be run vis-à-vis TPs who are running their browsers in their own modes on their own machines.

7. No automatic scrolling on the wizard's side

Graphical interfaces might be continuing below "the fold" (cf. e.g. Krug 2006 or Nielsen 2010) and the user only gets to this part by scrolling down the page, that is, the content might be longer than the device's screen height (or, more correctly, the window height). When the interaction shell is larger than the screen of the device (the TP window), and TP is scrolling in any direction, the Ozlab wizard must manually scroll the wizard interface to be able to see what TP is viewing and interacting with (this should not be automatic as the wizard might like to fix a few things on the upper part of the scene while the TP is browsing the lower parts). On a computer setup this is easy, because the wizard can see the TP's pointer at all times. For touch devices it is not that simple. For future versions of the Ozlab system the TP browser must be made to signal back to the system which part of the scene is visible to TP.

6 Limitations of the Wizard-of-Oz technique

After the account just given of limitations inherent in a specific *technology* used for the implementation of a generic WOz tool, this chapter will give an account of limitations in the Wizard-of-Oz *method* itself. These problems have been touched upon here and there in the literature review and will now be summed up and expounded on. All limitations are not applicable to every conducted WOz experiment – different purposes set different standards. In an exploratory setting, for example, the reliability issue mentioned below is not really an issue as researchers may indeed seek unforeseen interaction patterns and responses.

However, as with all techniques and methods, being aware of the limitations and disadvantages is necessary as the experiment can then be properly adapted and issues dealt with. The limitations presented below are in some settings issues to be accounted for when conducting WOz experiments.

6.1 Validity

This sections starts by addressing the question: Valid for what? Some experimenters stress the experience of the wizard and perhaps the whole design team. For instance, among the lessons learned in the Turvy project, the experimenters comment: "*The designer benefits greatly by becoming the Wizard.* [¶ …] By acting as Wizard, facilitator, and interviewer, the experimenters become immersed in the experiment and many important results become obvious." (Maulsby, Greenberg & Mander 1993, p. 283)

The experience of working inside the user interface gives the designer a better feeling for what the means of production can actually express and also for what the targeted user group would need. This feeling is hard to 'validate' and it would be easy to brush it aside by saying that the experimental setup must bias the experience in some ways. However, analysing the premises of the experiment should not be harder in WOz than in other prototype testing or when using interviewing techniques.

Consequently, we argue that such concerns should not make developers and designers hesitant to meet representatives of the future user groups face-to-interface rather than only face-to-face.

When gauging the validity of an experiment from this perspective – i.e., designer's experience of the interaction ("DX", as it were, not UX) – the standards must be those of design support and idea generation.

The main issue regarding the validity of Wizard-of-Oz experiments is otherwise – or was initially at least – whether such man-made elicitations of user behaviour could be comparable to real human-computer interaction. As seen in section 1.2, the Swedish NLP group working with ARNE addressed several aspects of the validity issue:

- Existing systems elicit other behaviours than the imagined system would trigger; thus, one needs to mimic the imagined systems to make valid HCI experiments.
- People are indeed fooled by the Wizard-of-Oz setup.
- Laboratory-based studies entail rather artificial situations, but depending on the purpose this might not affect the validity of a study at all (one may add that nowadays many Wizard-of-Oz studies are run in the wild).

Concerning the last point, Dahlbäck's and co-workers' argument was that to generate a natural-language corpus, there is simply no chance for ordinary people to role-play on the relevant linguistic levels. However, Eklund (2010) later pointed to the fact that informing test participants that they are being recorded may change some of their behaviour – in the study he related, "filled pauses" turned out to be much more frequent than in laboratory studies. But this is more a question of reliability than of validity. One might note that for a usability testing purpose, a system that does not pass even a lab test will have fairly small chances to work in the wild anyhow.

The reasons for hiding the wizard, and thus deceiving the user, can vary. In NLP research it is often vital that the test subject believes that he/she is talking to a real computer system, as the goal is to find out how such interaction would look (sound) like (Dahlbäck et al. 1993). Benzmüller et al. (2007) argue that it is crucial when conducting WOz experiments that the user is led to believe that he/she is interacting with a fully developed system. Deceiving the subjects is not always easy. Munteanu and Boldea (2000, p. 107) conducted their experiments with students in computer science as test subjects, who were "very suspicious about the system" which made deceiving the subjects harder.

However, some of the articles reviewed show that the wizardry necessarily does not need to be hidden from the user:

- In the Turvey project, "While we did not deceive users, they quickly bought into the illusion. They spoke more curtly to Turvy than to the facilitator, and referred to Turvy and the Wizard as two separate entities" (Maulsby et al. 1993, p. 281).
- White and Lutters (2003) conducted a WoZ study where they discussed the methodology with the participants prior testing.
- Molin (2004 p. 427) conducted studies with surgeons who participated during the GUI design process with "the effect that the 'computer' was given a human face".
- Lee, Mott and Lester (2010) told their participants that they would interact and cooperate with another human during the test (i.e. the wizard acting as a director agent).
- In explorative WOz tests the wizard does not need to be hidden from the test subject, as discussed in 1.1.
- In general, and in contrast to natural-language experiments, GUI interaction may carry the user away from any human-to-human behaviour; thus, secrecy is often not essential (cf. refs. in section 2.1.1 on Ozlab-based studies).

It should be noted that some systems should not be simulated by using WOz even if they are very graphical and not based on NLP at all: attempting to simulate action games is highly inappropriate as the whole point in some games is that the computer is quicker than the human player. In general, a human wizard cannot compensate for the simulation environment's response time, no matter how fast he or she works. Using a simulation environment with a large latency would make it impossible for the wizard to simulate direct manipulation and to interpret the subject's behaviour quickly (Salber & Coutaz 1993; Serrano & Nigay 2010). Similar, a wizard must not perform better than the intended system. An example is given in Pettersson's (2002) report on a study where one of the educators acting as wizard started to converse with the test subject in a too humanly manner. If not faithful to the method, the experiment will simulate a different system than the intended one.

This relates to the danger of endowing a mock system with a functionality that no real system could have. When evaluating new concepts, this might not be wrong, but if the system is to be developed in the near future, the wizard's powers should be considered and constrained to a reasonable level (again, one can refer to Maulsby, Greenberg & Mander 1993; see section 1.2

above). That said, the iterative development procedure should also be emphasised because at the first stages the interaction space is larger than what the intended system would allow for. One example of narrowing the communication between the user and the wizard is the study concerning linguistic training reported by Pettersson (2002). During a test session a few months into the project the wizard "was not allowed to peep into the test room but only get the stimuli a real computer program would have got, in this case the mouse. For the same reason, audio feedback from the test room was prevented." (p. 153) Mavrikis and Gutierrez-Santos (2010, p. 644) argue that reducing face-to-face interaction down to what the proposed system would grant the facilitator should be made gradually, making a stepwise refinement of the system's pre-defined feedback possible. Similar tapering methods are not commonly reported. These two examples stem from the educational area where there already are methods for guiding people, namely by teachers, and hence the first rounds may not need any WOz technology at all: it is more about capturing how good teachers perform when guiding students. (Admittedly, teachers are not always successful. Nevertheless, the general presumption is that teachers help students, at least if the teachers are applying good pedagogical methods and if there is enough time; lack of time could potentially be compensate for by educational software).

To conclude this last discussion on validity, interaction spaces out of range of the intended system's interaction capacity are not wrong in themselves. They simply belong to preliminary and early design iterations.

This conclusion will affect how corpus generating experiments are valued (NLP experiments in particular have often aimed at presenting interaction corpora). The reviewers of Wooffitt et al.'s book-length study on *Human, Computers, and Wizards* (1997) find it hard to see how the researchers can know which system capacity should be simulated in order to generate useful data. "Arguably, if we were ever able to develop a computer system whose speech abilities were indistinguishable from those of a human, then none of the differences between human-human and human-computer dialogues reported in this book would occur," argues de Vicente (1998, p. 81) and suggests that "many of the differences reported here are caused by the particular 'implementation' of the system". Two other reviewers note that "no example is given of how exactly the data analysis presented in this book could be used in the development of speech recognition systems" (Hak 1999, p. 587) and "it is disappointing that no practical demonstrations are provided of how systems designers are to benefit from the study" (Button 1998, p. 897). In systems development, WOz data from one iteration highlighting some specific unwanted features will influence the re-design and, hence, the data collection will in principle not be valid any longer because the interaction design tested will not be further used. This invalidation is of course not that definitive in reality, corpora can be used and re-used, but the 'C' in HCI is always a human construct as much as the 'H' is depending on target groups. A prototyping perspective sheds light on this as prototyping works by the model: $HC_1I \rightarrow HC_2I \rightarrow HC_3I \rightarrow \ldots$ (and even this un-fixed C_i in this model is actually too linearly developing for an explorative phase of a systems development where branches will be allowed for).

The next issue to be discussed is reliability, that is, whether or not the method (compared to other methods) produces results with such high precision (randomlessness) that these results are really useful.

6.2 Reliability

One criterion of good reliability is that if B repeats a 'reliable' experiment originally made by A, B would reach the same results as A. Is this possible with Wizard-of-Oz methods? Perhaps not. The wizard would suffer from fatigue if not already in A's experiment, at least when re-employed by B. This line of reasoning hinges on W being used as wizard all the time. If the crucial component, i.e. W, is replaced between A's and B' experiment, we do not have the same experiment. The reason for having tens or hundreds of test subjects in experiments is to even out individual differences, or at least to make such differences reappear in representative proportions for each experiment. There is definitively a question of how much the individual characteristics of the wizard(s) affect in the results.

This is compounded by the risk that one and the same wizard may behave differently from one session to the next even if properly trained in pilot tests. Just as we rarely recommend test sessions lasting for several hours because people will be exhausted,[21] we must take wizard fatigue in WOz experimenting into account. Even within one (extensive) session a wizard may not perform uniformly.

As already stressed in the preceding section, variance in wizard performance may not be detrimental and sometimes actually intended. But for corpus building or for some test-as-evaluation setups, the result of the study can be less reliable due to variations in human behaviour. Commonly discussed is

[21] Well, there are other reasons too, as Jakob Nielsen reminds the reader in his blog post "Time budgets for usability sessions" (2005).

the consistency of the wizard, but for clarity, let us recognise that there are two different dimensions: one about consistency within and between test sessions (could also be said to be within and between test subjects), and one consistency dimension that pertains directly to wizards. To start with the first one:

- *Intra-session consistency*. There is a risk that the wizard will be tired or stressed causing more errors at the end of the session (but, if TL does this for every session, there is both an intra-wizard consistency at whole-session level and inter-subject/session consistency). In explorative WOz tests the level of inter-sessional consistency should be low, at least during early iterations. (Fig. 1, p. 166, Pettersson 2003, or see 4.3 above)
- *Inter-session consistency*. Between different sessions (often then, between different subjects (i.e. test participants) the responses from the faked system remains the same.

Then, for the wizard consistency dimension, it should be noted that all studies are not using a single-wizard setup, which is why wizard consistency can concern the following:

- *Intra-wizard consistency*: The study can have one single wizard who interacts with all participants in the study. Intra-wizard consistency pertains to the requirement (or tacit presumption) that the wizard interacts (simulates the system) in a consistent way with all participants.
- *Inter-wizards consistency*: The study can have several wizards but only one wizard interacts with the test subject during each session, i.e. the wizards are switched between sessions. Inter-wizards consistency means that all wizards interact with test subjects (i.e., simulate the system) in a similar way.
- *Multi-wizards consistency*: The study can have several wizards who interact with the test subject during each session. Multi-wizards consistency means that all wizards interact (simulate the system) with the one test subject in a consistent way.

In the study reported by Pettersson (2002) the educators acting wizards found the spoken feedback to the participants harder to produce than the graphical ones. The bullet about multi-wizard consistency is important considering the observation by Oviatt that, "when a recognition error does occur, users alternate input modes" (1999, p. 79). As noted in section 1.2, Oviatt was referred to by Serrano and Nigay (2010) when explaining why mixed systems are necessary for fair evaluation and unbiased development of systems with multimodal input: the human stand-ins for the less reliable parts make for a unbiased interaction, but it entails the requirement that all wizards perform on an equally skilled level.

In general, before conducting WOz experiments the wizard must be aware of how to act in the different situations that may occur during the sessions. Further, the wizard must know what information is possible to provide to the user and how to acquire it. Otherwise, the wizard's actions will make the results less generalizable. Pilot tests conducted before the real experiments can provide the wizard with such knowledge (Dahlbäck, Jönsson & Ahrenberg 1993, p. 264); see the final section of this chapter for pertinent examples of lazy students who did not do the homework (pilot straining) before the real tests! Intra-sessional lack of consistency will reveal the human source of the system responses and may make test participants behave as speaking to a human. However, Dahlbäck et al. argued that their use of several wizards ensures that participants' sentence constructions are "not the reflection of the idiosyncrasies of one single person's behaviour" (p. 265) – thus thwarting the ordinary rule "less reliability threatens the validity of the study". Also Höysniemi and co-workers in their study on gesture control of animated characters regard idiosyncrasies as the real problem:

> "The collected movement corpus is context dependent and is influenced by the wizard's abilities to adapt to the user's actions. To decrease the wizard's effect on the test data, it is advisable to use several wizards. This, on the other hand, leads to more time-consuming tests and data analysis."
>
> (Höysniemi et al. 2004, p. 33)

At any rate, some argue that by providing the wizard with guidelines on how to act and respond the simulation can be made more consistent, even when switching wizards between sessions (i.e. ensuring the inter-wizards consistency) (Mäkelä, Salonen, Turunen, Hakulinen & Raisamo 2001). Maulsby et al. (1993) state that in order to keep the simulation honest, it should be based on an algorithm. They add that such algorithms could be used to code the results if too complex for the wizard to follow during runtime. However, Lee et al. (2010) experienced that wizards can have different styles of interacting with the test subject even though the wizards received procedure, interaction and narrative protocols developed during pilot studies, and the same training before the experiments were conducted.

Li's dissertation from 2012 aims at giving an understanding of the threats to consistent system operation. As mentioned in section 1.2, Li demonstrates through a series of WOz experiments the impact on the consistency of system operations of four factors: interaction schema, wizard user interface, wizard's interpretation of participant's actions, and inter-wizard variations. E.g., observations of details in idiosyncrasies and inter-wizard variability as concerns interpretations of test participants' activities and of participants' intentions are illuminating even if Li's sample is very small (ibid., pp. 11f, 146f). The individual findings are probably not unknown to designers of WOz experiments but perhaps seldom considered in their totality as Li does. Li seems to suggest more control and re-work to reach more perfect (consistent) WOz sessions (pp.156-160), which place this work far from the explorative applications of WOz methodology and from the practical, resource-constrained world of systems development.

To conclude, wizard variability must be reckoned with but not desperately avoided. There is only one rule: if no variability is tolerated, don't bother to call in the Wizard from the Emerald City but use a programmed prototype instead.

6.3 Efficiency and reuse of prototypes and results

Some of the authors in the reviewed literature criticize the Wizard-of-Oz technique for not enabling reuse of prototypes and results in the iterations following the WOz experimentation in a systems development cycle.

Li and Bonner (2013, p. 3) mention that the method is criticized for "the repetitive development of separate interfaces for system facilitation and interaction". Dow et al. (2005, p. 18) argue that "designers tend to use WOz studies once (or perhaps twice) during a system's evolution".

If the underlying simulation system is built each time an experiment is to be conducted, the efficiency of WOz experimenting is not very impressing. A generic WOz tool would take care of this issue. When it comes to reuse of the prototypes and results, the present authors find argument for letting WOz remain a rapid-prototyping technique, or rather, a throw-away prototyping technique. Thus, WOz is used to find the best possible idea or design, not to produce source code, while acknowledging that the Wizard-of-Oz technique is not applicable for all purposes, just as paper prototyping cannot be used for all purposes in design and development (noted by, inter alia, Davis et al. 2007, p. 119).

Also the ConWIZ team repeats the assumption in Dow et al. (2005), saying that "designers have to bridge the gap between the wizard's role and actual system implementation" (Zachhuber et al. 2012, p. 226). Enabling rapid prototyping and reuse of WOz should be possible to incorporate the WOz prototype into the real prototype, as Zachhuber et al. (2012) and Dow et al. (2005) argue. Serrano and Nigay (2010, p. 218) launch the argument that some imperfect system parts should be replaced with WOz. This then makes it valuable to incorporate the WOz modules in the rest of the implemented system.

Now, it is quite a burden to maintain a WOz system and also a development environment. Serrano and Nigay were working on a framework rather than a system (see esp. the footnote to OpenWizard in section 3.2). Building on existing platforms for development such as Director, which DART did (as well as the original Ozlab albeit with a lesser scope), entails the risk of being stranded when the producer no longer maintains the platform. For corporate labs, mixed systems such as WozARd, developed in the labs of Sony Mobile Communications, may last longer than the first experiment. However, based on the review of (more or less) generic WOz tools it is fair to conclude that the mixed systems designed for inclusion in programming environments do not live long. This is perhaps no argument for not including WOz facilities in multimedia/UI development tools, but most programmers will probably not see the use of such modules while designers cannot be limited to tools which extend far beyond their scope, namely into implementation.

6.4 Ethical considerations

Commonly, the wizard is hidden to the participants when conducting Wizard-of-Oz experiments. We have argued earlier in this work that in many employments of the WOz technique it is not necessary to hide the wizard or the fact that there is a test leader monitoring and controlling the system. However, in studies where the user "needs" to be and actually is deceived, one must take the ethical aspects of such experiments into consideration.

Dahlbäck, Jönsson, and Ahrenberg (1993) debriefed their participants on how the experiment was conducted after each session as a solution to these aspects. None of their participants showed any hard feelings. The authors argue that this might be explained by the nature of the research. Conducting studies where the subjects are put in uncomfortable situations, the subsequent reactions might be different.

Interestingly, Höysniemi, Hämäläinen, and Turkki (2004) make a pro-ethical argument for WOz: they argue that by using WOz, their study became less discriminating than the testing of a "fully" functional prototype would have been, as the wizard could interpret the children's body movements better. Thus, the children were not put in compromising situations as the "system" could understand them. (Compare this with the scientific argument by the multimodalists: WOz is needed for validity by increasing the reliability; see the discussion on multi-wizard consistency above).

While all authors relying on deception seem to agree on the necessity to inform afterwards about the non-existence of a system with the functionality just tested by the participants, there is quite a tricky type of situation which calls for special attention, namely when small children are involved (or people with learning difficulties – or both, as in the first Ozlab study reported by Pettersson 2002). Parents will of course be informed but the most pertinent issue in this particular case is not to let the wizard engage in a fully human-like capacity as this will not only make a wrong impression on the child of what a computer can understand but probably also a lasting one (ibid.).

Finally, conducting WOz experimentation in web pages that are already up and running (or in other applications which are running online) obviously has the drawback that users are not aware of being supervised. While it may not differ much from post-hoc monitoring of web pages (for instance for web analytics; Peterson 2005), there is a risk in human real-time interpretation performed for an organisation's intranet or smaller community that the wizard recognises the user and that the user displays behaviours (clicks-streams) or text or other input that he/she would not have made if conscious of the human monitor. On the other hand, queries in search boxes are most likely not the same when put to a search engine as when put to a human. Thus, information of the possibility of human monitoring should be displayed at the relevant web pages even if the true nature of the production of answers is not revealed. (This is comparable to so-called interactive FAQs where an answer is promised within, say, 24 hours, even if the human behind the answers in the FAQ is revealed.)

6.5 Delays and time lag

"In terms of problems researchers were facing it seems that delays coming from the wizard constitute the biggest challenge, specifically mentioned by 9 of the 20 interviewees" Schlögl, Doherty, and Luz report from a telephone interviews with researchers from industry and academia (2014; see their Table 1 and sec. 4.2). In fact, there are several reasons why time lag or delays can occur. This is either due to the wizard's interpretations of the user's input, the wizard's actions, and/or perhaps due to time lag in the used system/WOz tool. Dahlbäck, Jönsson, and Ahrenberg (1993, p. 264) note the importance of the wizard's knowledge of the simulated system, the simulation environment, and its information when conducting experiments. Without such knowledge the wizard will cause simulation delays.

Time lag and delays can affect the participants in the study. Akers (2006, p. 459), for example, note that "[u]sers found it frustrating that some of their actions took over a minute to simulate using the Wizard of Oz controls." Akers suggests that the frustration could perhaps have been avoided if "artificial delays to even out the timing" were implemented but argues further that "additional delays would have introduced further frustration" (ibid.; cf. Robins et al., 2008, even if their study of children relied on 2-minute interactions, which is too short to predict real system usage).

By comparing WOz experiments with other prototyping techniques, the time-lag issue can also be put into perspective. For example, when conducting evaluations with paper prototypes, the test participants know and play along with the evaluation technique. The Wizard-of-Oz experiments could be adapted in the same way. Even if the test participant is not told that all interactions are simulated by a human acting wizard, the participant can be told that what is being tested is a prototype which is why the response time can be longer than normal.[22] Another solution is that the wizard simply simulates some time lag, which can lead the test subject to play along – a slow pace interaction sets the tone for the interaction (cf. the tendency to convergence; Leiser 1989). One can also ask TP to "think aloud". This generally slows people down. Admittedly, thinking aloud might make people think more than normally, or at least differently. In a longer design cycle with several iterations this will generally not be a problem – the time will come when the wizard and the prototype shell are prepared for what people will do.

It should be noted, though, that all wizards are not causing delays that affect the study, as noted by Höysniemi, Hämäläinen, and Turkki (2004). The wizard in their study did not cause significant delays when interpreting and acting on the children's movement when playing their (simulated) body movements controlled game. The task of this wizard and that of the wizard in the case reported by Akers differ markedly. As noted in Chapter 1,

[22] From experience we also know how frustrating it is when the alleged functioning prototype is introduced, replacing the paper and Ozlab mockups, and it turns out that response time delay counts in minutes for some functions or else the prototype is caught in an infinite loop without any immediate sign of this. It is really not a pleasant experience for the experimenter to expose test participants to such meaningless usability testing.

Akers's setup allowed participants to design and test gestural interface for 3D selection. As also described in Chapter 1, Skantze's and Hjalmarsson's "incremental" method for building system's utterances included self-repair strategies, which allowed a wizard to work efficiently. "The experiment also shows that it is possible to achieve fast turn-taking and convincing responses in a Wizard-of-Oz setting. We think that this opens up new possibilities for the Wizard-of-Oz paradigm, and thereby for practical development of dialogue systems in general." (Skantze & Hjalmarsson 2010, p. 8)

Li and Bonner (2013 p. 3) acknowledge that WOz is criticised for "the delay of the designer's responses which are incurred by the designer's interpretation of user interactions between two types of interfaces". However, Li and Bonner state that in their study, "[a]lthough there was a noticeable speed decrease from study one to three, users' feedbacks suggested that slow responses were not often sensed." (p. 10) On the other hand, Li and Bonner used an experienced wizard already in the first study, so perhaps the delays were not long to begin with. (It is perhaps somewhat remarkable that Li and Bonner used an experienced wizard while they claim their system makes WOz much easier than traditional WOz; cf. LIVE in section 3.2). From the accumulated wisdom gained by Ozlab users, we can add that the people making the shell (the prototype) have a much shorter learning period to act as wizards than other persons.

6.6 Cognitive load – Wizard stress and fatigue

Many of the reviewed articles discuss the wizard's role when conducting Wizard-of-Oz experiments. Depending on what kind of system or aspects of a system are being simulated, the wizard's cognitive load can be higher or lower. Overall, however, the Wizard-of-Oz technique does seem to put the person(s) acting wizard under some stress. The most reported reason for stress and cognitive load is that of interpreting the user and making up responses in a timely manner. In an interview with Jenny Nilsson, an expert user of the Director-based Ozlab system, it became clear that simulating a system is demanding. Nilsson stated that she could perform about three test sessions a day, as the wizardry demanded such a high degree of concentration and energy. She was then working alone, including receiving and debriefing participants and making notes.[23] In the experiment of

[23] Of 1-1.5 hour time with a test participant (i.e. excluding note taking afterwards), 0.5-1 hour was spent on the actual Ozlab-based prototype interaction. (p.c. 2014-06-12)

Mäkelä, Salonen, Turunen, Hakulinen, and Raisamo (2001) the members of the test group changed roles, as the wizard role was demanding and required alertness. Dow, Lee, Oezbek, MacIntyre, Bolter, and Gandy (2005) argue that their wizard(s) suffered under huge task load. During the iterations in the experiments by Dow et al. (2005), the wizard role altered from monitoring the context and controlling the prototype to stepping in when needed. The wizards' interface was also redesigned during the iterations, hence the tasks for the wizards could be carried out in a more effective manner.

It should be noted that when simulating a system, the time lag does not always have to be a big issue ruining the experiment. However, being new to the technique might mean that one believes this to be the case. Inexperienced designers, new to the field of usability testing and the Wizard-of-Oz technique in particular, can be sensitive to their lack of experience, which in itself can be a source of stress (compare the following section).

Some work on reducing the workload, and thus possibly the cognitive load, for the wizard in WOz studies has been reported. For example, if location tracking is simulated, it requires high attention and gives the wizard a great task load. Therefore, Li, Welbourne, and Landay (2006) tested four simulation techniques for continuous location tracking. The authors compared two benchmarking techniques ("Pick&Drop" and "Drag&Drop") with two new techniques ("DirectionalCrossing" and "Steering"), and found that the latter two significantly reduced the work load for the wizard as well as achieved similar tracking accuracy.

Most of the WOz tools in the reviewed articles support a single-wizard setup (see for example ConWIZ, WebWOZ, MDWOZ and DiaWOz-II in section 3.2). However, in WOz experiments on multimodal systems the *Information Bandwidth* (Salber & Coutaz 1993), meaning the (subject's) possibilities for input, increased and this affects the task load and cognitive load for the wizard. If interpreting and simulating system responses to several input modalities at the same time, Salber and Coutaz (1993) argue that the wizard's responses could become inconsistent. The authors suggest that in such cases a multi-wizard setup should be preferred, where each wizard handles each modality. Examples of such WOz setups are NEIMO (Coutaz, Salber, Carraux & Portolan 1996) and OpenWizard (Serrano & Nigay 2010). In both NEIMO and OpenWizard, every wizard can be designated to simulate a specific modality. The wizards in the OpenWizard study pointed out that the "multi-wizard configuration helped them managing stress" (ibid., p. 224)

However, several wizards controlling different modalities or aspects of the system are not always wanted, nor needed. The special educators acting wizards in the trial version of the first Ozlab system (Pettersson 2003), obviously managed both the graphics and voice (disguised to GUI characters' voices), and also took care of test participants before and after test. Dow, Lee, Oezbek, MacIntyre, Bolter, and Gandy (2005) argue that several wizards could have been used during their "Voices of Oakland" experiment, although only one wizard was used. On the other hand, Höysniemi, Hämäläinen, and Turkki (2004) argue that a multi-wizard setup in their study would have affected the test subjects and the collected data, as a multi-wizard setup would have resulted in more adults than children in the test environment.

Some arguments have been found in the reviewed articles for modifying the WOz system to do some of the work for the wizard – automatically (see for example Klemmer, Sinha, Chen, Landay, Aboobaker & Wang 2000). However, implementing automatic responses makes the test-setup more rigid, and the possibility to conduct explorative WOz tests is reduced. Using wizard guidelines, as discussed above, for increasing reliability, could potentially reduce the cognitive load if it is easy for the wizard to follow both TP actions and the script. However, it might also make the wizard a bit less open-minded: "One test person tried to use the time axis to shift scenes (i.e. to navigate between pages). The wizard rejected this attempt and thereby missed the opportunity to allow for unintended use of this feature." (Larsson & Molin 2006, p. 368)

In conclusion, therefore, wizard stress and fatigue are factors to expect in experiments using the Wizard-of-Oz method. The effect of the results depends on the purpose of the experimentation and the degree of automated wizard support as well as on the training done involving a particular setup, session length, and distribution of workload across several test leaders.

6.7 Example: Wizards' interaction patterns when learning Ozlab

Acting wizard can come with heavy cognitive load and stress as discussed above. Here we give some glimpses of problems in using the WOz technique when designers are new to user testing and to this technique in particular. Data come from a course run in the autumn 2013 with the first release of the web-based Ozlab. Screen recordings were made of TL screen in a mandatory pilot test and some days later when "real" test sessions were made.

The stress of acting wizard were especially noticeable in the students who had not practised the wizard role (enough) prior to conducting the tests.[24] The screen recordings show that some groups had not practiced the possible navigation paths through the interaction shell or discussed how the wizard should act and respond to potential ways in which a participant could interact with the prototype.

However, when analysing the screen recordings from the test sessions, it became clear that *the time needed for the students to learn how to act as a wizard was short*. Often the role of being a wizard was refined already during the first session. For example, in some interaction shells the test participant could not move to new scenes without the wizard producing a scene change, so to make a hasty change of scenes the wizard should keep the mouse cursor over the list of scenes to the left of the interface. However, during the first test session several wizards held the mouse cursor in the scene area, close to where the test participant interacted with the prototyped interface. Later in the same session, or in following sessions, the wizard seems to have learnt that if the mouse cursor lingered over such areas, he/she must move the cursor to the scene list, locate the link to the scene corresponding to the test participant's choice, and then click that link in order to effectuate the scene change. Of course, the amount of time for doing this is not extensive, but it could at least be argued to be more stressful than just clicking the link to go to the expected scene.

There are also other instances of learning which have more to do with prototype development: 4 of 12 student groups included "Quit" as an available option for TP but included no scene indicating that the game was terminated, for instance a mockup of the Windows desktop. Of course, these groups immediately realised their error when running tests with a TP not from their own team (again, see the previous footnote; for more details on this trial, see a report by Malin Wik available on the Ozlab web site[25]).

[24] That is, prior to a pilot test and "real" test; such practicing could of course also be labelled "pilot testing" but in the Ozlab team, with our long experience of WOz, we reserve the word "test" – including "pilot test" – to cases where more or less "real" users participate, that is, when participants are not from the design group.
[25] http://www.kau.se/en/ozlab/research-and-development-projects/student-works

7 Concluding Remarks

The preceding chapters have dealt with the characterisation of prototypes and prototyping, the development of the Wizard-of-Oz method, Ozlab's structure and other generic WOz tools. Then the focus shifted to more targeted discussions on the practicalities of WOz work, the platform-dependency of WOz tools, and finally on issues inherent to the Wizard-of-Oz method itself.

Classification schemes for prototypes may easily overlook some essential parts of Wizard-of-Oz prototyping, especially since a WOz mockup may be used in an explorative manner, and later to evaluate an interaction paradigm that largely lies in the hands (and mouth) of the wizard. The discussion of this in Chapter 4 centred on exemplifying explorative WOz prototyping and the different experimental roles that the stakeholders in a development process can take; for instance, the prospective users may end up as wizards in some runs. Using the human wizard as a *human being* was stressed already in Chapter 1, because if the wizard is used only as a machine substitute it will largely restrict interactive prototyping to testing and exclude explorations of details. Moreover, hiding behind the curtain is often not needed – rather, different stakeholders can "speak" via a WOz mockup to better understand the limits and demands of certain types of UI solution. Such exercises reveal many requirements for a future user interface.

Chapter 4 also elaborated on the classification of the wizard output production. This is only indirectly related to what test subjects experience. Some interesting development for the future relates to using test subjects' input in the wizards' output. The important thing in the present work was to highlight the many aspects of wizard-supporting functions of what might be seen as the same sort of output directed to test subjects (even if the enumeration did not go into the detail of the movements of physical objects in 3D space).

The historical exposé in section 1.2 show many purposes for which WOz is employed. In each case, there are one or several humans to execute the system responses. Our own Ozlab fits into this history as an attempt to cater for more flexible prototyping of GUI communication between test subject and wizard while at times we also used it in conjunction with speech output and robot arm movements. It was also important to provide a system rather than a setup so it could be reused for a wide range of applications. However, even if a re-usable WOz tool is made to circumvent the need for programming prototypes, the tool itself will eventually need re-programming to accommodate changes in operating systems and UI hardware. Chapter 3 demonstrated how hard it is to have a generic system just waiting for someone to use it. In the case of Ozlab, the repeated use not only for different research purposes but also for undergraduate courses and in student projects made it possible to keep it running for a decade before it had to be replaced completely. The problem to fit more elaborated WOz systems into ordinary university curricula is acknowledged in the recent WOz survey by the WebWOZ group:

> "From a design perspective, students studying Human-Computer Interaction (HCI) and Interaction Design will generally be introduced to WOZ, yet only a small proportion of these will actually experience the method when compared with exercises based on the use of paper prototypes. One reason for this lack of practical usage might be that in order to be applicable in an HCI teaching context, any approach would have to have a low logistical and technical overhead to enable students to quickly design and carry out evaluations."
>
> (Schlögl, Doherty & Luz 2014, p. 3)

Ozlab's simplicity came from the focus on ordinary GUI applications. A group of students could easily make a meaningful design to test. However, the dependence on a multimedia production tool eventually made the first implementation of Ozlab crumble.

Now we are elaborating a web-based solution as mentioned in Chapters 2 and 5; indeed, some other WOz teams are doing the same, as shown in Chapter 3. While the web is a fairly generic technology popping up in nearly every modern hardware, it also has some distinct disadvantages, especially the prolonged response time, but also the problems of keeping control of scrolling, surfing, and browsers, as discussed in Chapter 5.

Bringing this report to a close, we would like to remind the reader of some of the problems with Wizard of Oz noted in Chapter 6, especially wizard learning and fatigue, and wizard UI discussed in some contexts in Chapter 3. Our point was not that the dimensions of production enumerated in section 4.1 would find a uniform and consistent wizard UI. Rather, we maintained that except for some standard functions such as "lock input" and "freeze screen", each WOz mockup deserves its own wizard UI even in a generic WOz tool. Other WOz experimenters have highlighted the design of the wizard UI but there seems to be different paradigms for this; in Ozlab, the GUI of the test subject is the natural place for the design of the wizard user interface. This makes prototype design and wizard UI design concurrent events. It seems, moreover, that the possibilities of web browser based UIs strengthen such a claim.

The topic of the design of wizard controls should merit more attention in the years to come, not least as new devices and applications are growing in numbers in areas such as augmented reality, social network supported applications, service robots, and systems with active attendants (support staff, wizards). It is to be noted that different interaction modes are increasingly integrated in new products. This is very obvious in smartphones with their accelerometers, cameras, GPS, microphones, touch sensitive GUIs, flash lights, vibrations, and loudspeakers.

In conclusion, there are several aspects one could pay attention to in future WOz-supported studies and studies on WOz tools. In discussion with various colleagues (in particular Cosmin Munteanu, Univ. of Toronto), we see especially the following points worth to comment on (or explore) in future studies:

- Platform-dependence of WOz tools
- Efficiency and reuse of prototypes and results
- Validity and reliability issues arising from the WOz method (time lags, variations in wizard performance, etc.)
- Issues of cognitive load (wizard stress and fatigue; wizards' user interfaces)
- WOz setups in the wild (with and without physically present wizards)
- Applicability and challenges of using WOz with mobile platforms
- Definition of the various approaches to the deployment and support of human wizards during experimentation / development cycles

References

Akers, D. (2006). Wizard of Oz for Participatory Design: Inventing a Gestural Interface for 3D Selection of Neural Pathway Estimates. In *CHI'06 Extended Abstracts on Human Factors in Computing Systems*, pp. 454-459.

Alce, G., Hermodsson, K., Lasorsa, Y., Liodenot, D. Michel, T., Razafimahazo, M. & Chippendale, P. (2013). D3.2 Interface design prototypes and/or mock ups. Submitted 2013-01-29. VENTURI. https://venturi.fbk.eu/results/public-deliverables/ [2014-05-30]

Alce, G., Hermodsson, K. & Wallergård, M. (2013). WozARD: A Wizard of Oz Tool for Mobile AR. *MobileHCI 2013*, Munich, Germany, August 27-30, pp. 600-605.

Alce, G. & Hermodsson, K. & Wallergård, M. (forthcoming) Evaluation of WozARD: a Wizard-of-Oz tool for wearable devices.

Ardito, C., Buono, P., Costabile, M. F., Lanzilotti, R. & Piccinno, A. (2009). A tool for Wizard of Oz studies of multimodal mobile systems. *HSI 2009*, Catania, Italy, May 21-23, pp. 344-347.

Aubergé, V., Sasa, Y., Bonnefond, N., Meillon, B., Robert, T., Rey-Gorrez, J., Schwartz, A., Antunes, L., De Biasi, G., Caffiau, S. & Nebout, F. (2014) The EEE corpus: socio-affective "glue" cues in elderly-robot interactions in a Smart Home with the EmOz platform. Presentation at *ES^{3}LOD 2014*, *the 5th International Workshop on Emotion, Social Signals, Sentiment and Linked Open Data.* Reykjavik, Iceland, May 26-27, 2014.

Baum, L.F. (1900) *The Wonderful Wizard of Oz*. With illustrations by W. W. Denslow. Georg M. Hill Company, Chicago.

Beaudouinn-Lafon, M., Mackay, W., (2003). Prototyping tools and techniques. In *The Human-Computer Interaction Handbook. Fundamentals,* *evolving technologies, and emerging applications*. Eds. Jacko & Sears. Lawrence Erlbaum Associates. Pp. 1006 – 1031.

Bellucci, A., Bottoni, P. & Levialdi, S. (2009). WOEB: Rapid Setting of Wizard of Oz Experiments and Reuse for Deployed Applications. *Proceedings of the IUI'09 Workshop on Model Driven Development of Advanced User Interfaces*, MDDAUI'09. Ed. by Meixner et al.. Sanibel Island, USA, February 8, 2009. *CEUR Workshop Proceedings*, Vol 439.

Benzmüller, C., Horacek, H., Kruijff-Korbayová, I., Lesourd, H., Schiller, M. & Wolska, M. (2007). DiaWOz-II – A Tool for Wizard-of-Oz Experiments in Mathematics. In *KI 2006: Advances in Artificial Intelligence.* Springer Berlin Heidelberg, pp. 159-173.

Bergmann, M., Rost, M. & Pettersson. J.S. (2006). Exploring the Feasibility of a Spatial User Interface Paradigm for Privacy-Enhancing Technology. Presented at ISD´2005, Karlstad. Published in *Advances in Information Systems Development*, eds. A.G. Nilsson et al., Springer-Verlag, pp. 437-448.

Bernsen, N.O., Dybjkaer, H. & Dybkjaer, L. (1998). *Designing interactive speech systems. From first ideas to user testing.* Springer.

Berry, D.C., Butler, L.T. & de Rosis, F. (2005). Evaluating a realistic agent in an advice-giving task. *International Journal of Human-Computer Studies*, 63(3), pp. 304-327.

Boehm, B. (1986). A spiral model of software development and enhancement. *SIGSOFT Softw. Eng. Notes,* 11(4), pp. 14–24.

Bönisch, B., Held, J. & Krueger, H. (2003). Prototyping.ppt – Power Point ® for interface-simulation of complex machines. In Jacko J. & Stephanidis C., *Human computer Interaction –Theory and Practice (Part II), Volume 2* (Proceedings of HCI International 2003, 22-27 June, Crete). LEA Lawrence Erlbaum Associates, pp. 1066-1070.

Broen, P.A. & Siegel, G.M. (1972). Variations in normal speech disfluencies. *Language and Speech*, 15(3), pp. 219-231.

Button, G. (1998). Book Reviews. [Wooffitt et. al. (1997).] *Sociology* 32, pp. 896-898.

Buxton, B. (2007). *Sketching user experience: getting the design right and the right design*. Elsevier/Morgan Kaufmann.

Caelen, J. & Millien, E. (2002). MultiCom, a Platform for the Design and the Evaluation of Interactive Systems. Application to Residential Gatewats and Home Services. In *Les Cahiers du numérique*, 3(4), pp. 149-171.

Carter, S. & Mankoff, J. (2005a). Prototypes in the Wild: Lessons from Three Ubicomp Systems. *IEEE Pervasive Computing*, Vol. 4(4), pp. 51-57.

Carter, S. & Mankoff, J. (2005b). When participants do the capturing: The role of media in diary studies. *CHI 2005*, April 2-7, 2005, Portland, Oregon, USA. Pp. 889-908. ACM.

Carter, S., Mankoff, J. & Heer, J. (2007). Momento: Support for Situated Ubicomp Experimentation. *CHI 2007*, 28 April - 3 May, 2007, San Jose, California, USA. Pp. 125-134 ACM.

Carter, S., Mankoff, J., Klemmer, S. & Matthews, T. (2008). Exiting the cleanroom: On ecological validity and ubiquitous computing. *Human-Computer Interaction* 23(1), pp. 47-99.

Carter, A.S. & Hundhausen, C.D. (2010). How is user interface prototyping really done in practice? A survey of user interface designers. In *Proc. 2010 IEEE Symposium on Visual Languages and Human-Centric Computing*, pp. 207-211. IEEE Computer Society, Washington DC, USA.

Cavalluzzi, A., Clarizio, G., De Carolis, B. & de Rosis, F. (2005) A Persona is Not A Person: Designing Dialogs With ECAs After Wizard of Oz Simulations. Report presented at HUMAINE WP6 workshop on "Interaction and Communication", Paris, March 2005. Available at http://www.di.uniba.it/intint/Humaine/H-Woz.html [2014-04-29]

Cavalluzzi, A., de Rosis, F., Mazzotta, I. & Novielli, N. (2005) Modeling The User Attitude Towards An ECA. *UM'05* workshop on "Adapting the Interaction Style to Affective Factors". Edinburgh, July 2005. http://www.di.uniba.it/intint/Humaine/H-Woz.html [2014-04-29]

Consolvo, S., Harrison, B., Smith, I., Chen, M. Y., Everitt, K., Froehlich, J., & Landay, J. A. (2007). Conducting In Situ Evaluations for and With Ubiquitous Computing Technologies. *International Journal of Human-Computer Interaction*, 22(1-2), pp.103-118.

Coutaz, J., Nigay,. L. & Salber, D. (1995). Multimodality from the User and System Perspectives. In *ERCIM'95 Workshop on Multimedia Multimodal User Interfaces*, Crete, Greece.

Coutaz, J., Salber, D., Carraux, E. & Portolan, N. (1996). NEIMO, a Multiworkstation Usability Lab for Observing and Analyzing Multimodal Interaction. Video presentation at *CHI'96*, Vancouver, British Columbia, Canada, April 13-18, pp. 402-403.

Dahlbäck, N., Jönsson, A. & Ahrenberg, L. (1993). Wizard of Oz Studies – why and how. *Knowledge-Based Systems*, 6(4), pp. 258-266.

Davis, R. C., Saponas, T. S., Shilman, M. & Landay, J. (2007). SketchWizard: Wizard of Oz Prototyping of Pen-Based User Interfaces. *UIST'07*, Rhode Island, USA, October 7-10, pp. 119-128.

de : de Rosis, de Ruyter, and de Vicente are sorted as Rosis, Ruyter, and Vicente, respectively.

Dey, A.K., Sohn, T., Streng, S. & Kodama, J. (2006). iCAP: Interactive Prototyping of Context-Aware Applications. In K.P. Fishkin et al. (Eds.): *PERVASIVE 2006,* LNCS 3968, Springer Verlag. Pp. 254 – 271.

Dow, S., Lee, J., Oezbek, C., MacIntyre, B., Bolter, J.D. & Gandy, M. (2005). Wizard of Oz Interfaces for Mixed Reality Applications. Poster presentation at *CHI 2005*, Portland, Oregon, USA, April 2-7, pp. 1339-1342.

Dow, S., MacIntyre, B., Lee, J., Oezbek, C., Bolter, J.D. & Gandy, M. (2005). Wizard of Oz Support throughout an Iterative Design Process. *Pervasive Computing* Oct-Dec 2005, pp. 18-26.

Edlund, J., Gustafson, J., Heldner, M. & Hjalmarsson, A. (2008). Towards human-like spoken dialogue systems. *Speech Communication*, 50(8-9), pp. 630-645.

Eklund, R. (2010). The effect of directed and open disambiguation prompts in authentic call center data on the frequency and distribution of filled pauses and possible implications for filled pause hypotheses and data collection methodology. *Proceedings of DiSS-LPSS Joint Workshop 2010*, September 25-26, 2010, Tokyo, Japan, pp. 23-26.

Erdmann, R. L. & Neal, A. S. (1971). Laboratory vs. Field Experimentation in Human Factors–An Evaluation of an Experimental Self-Service Airline Ticket Vendor. *Human Factors*, 13(6), pp. 521-531.

Fournier, H., Lapointe, J.-F., Kondratova, I., Emond, B. & Munteanu, C. (2012). Crossing the barrier: a scalable simulator for course of fire training. *Proceedings of Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC)* 2012, 2012-12-06, paper no. 12187, 10 pp. NRC Publications Archive / Archives des publications du CNRC, Canada.

Gould, J. D., Conti, J., & Hovanyecz, T. (1983). Composing letters with a simulated listening typewriter. *Communications of the ACM*, 26(4), pp. 295-308.

Green, A., Eklundh, K. S. S., Wrede, B., Li, S. (2006). Integrating miscommunication analysis in natural language interface design for a service robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4678-4683. IEEE.

Green, A., Hüttenrauch, H., & Eklundh, K. S. (2004). Applying the Wizard-of-Oz framework to cooperative service discovery and configuration. In *13th IEEE International Symposium on Robot and Human Interactive Communication (Ro-MAN)*, pp. 575–580. IEEE.

Grill, T., Polacek, O. & Tscheligi, M. (2012). ConWIZ: A tool supporting contextual Wizard of Oz simulation. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia*, Ulm, Germany, December 4-6, pp. 21-28.

Grill, T. & Tscheligi, M. (2013). The ConWIZ Protocol: A Generic Protocol for Wizard of Oz Simulations. *Computer Aided Systems Theory - EUROCAST 2013*. Springer Lecture Notes in Computer Science, Volume 8112, 2013, pp. 434-441.

Hak, T. (1999). Book Reviews. [Robin Wooffitt et al. (1997)] *Discourse & Society* 10(4), pp. 586-588.

Hartmann, B., Klemmer, S.R., Bernstein, M., Abdulla, L., Burr, B., Robinson-Mosher, A. & Gee, J. (2006). Reflective physical prototyping through integrated design, test, and analysis. In *UIST06*, October 15–18, 2006, Montreux, Switzerland. pp. 299-308. ACM.

Hauptmann, A.G. (1989). Speech and Gestures for Graphic Image Manipulation. *CHI'89*, pp. 241-245.

Hong, J.I. and Landay, J.A. (2000) SATIN: A toolkit for informal ink-based applications. *CHI 2000*, pp. 63-71.

Höysniemi, J., Hämäläinen, P. & Turkki, L. (2004). Wizard of Oz Prototyping of Computer Vision Based Action Games for Children. In *Proceeding of the 2004 conference on Interaction design and children*, Maryland, USA, June 1-3, pp. 27-34.

Hudson, S., Fogarty, J., Atkeson, C., Avrahami, D., Forlizzi, J., Kiesler, S., Lee, J.C. & Yang, J. (2003). Predicting Human Interruptibility with Sensors: A Wizard of Oz Feasibility Study. In *Proceedings of the SIGCHI conference on Human factors in computing systems,* Florida, USA, April 5-10, pp. 257-264.

Hundhausen, C.D., Balkar, A., Nuur, M. & Trent, S. (2007). WOZ pro: a pen-based low fidelity prototyping environment to support wizard of oz studies. In *CHI '07 : CHI '07 extended abstracts on Human factors in computing systems*, pp. 2453-2458. New York, NY, USA: ACM

Hundhausen, C., Trent, S., Balkar, A. & Nuur, M. (2008). The design and experimental evaluation of a tool to support the construction and wizard-of-oz testing of low fidelity prototypes. In *VLHCC '08: Proceedings of the 2008 IEEE Symposium on Visual Languages and Human-Centric Computing*, pp. 86-90. Washington, DC, USA: IEEE Computer Society.

Hüttenrauch, H., Eklundh, K. S. S., Green, A., & Topp, E. A. A. (2006). Investigating spatial relationships in human-robot interaction. In *Proceedings of the IEEE/RSJ International Conference on intelligent robots and Systems (IROS)*, pp. 5052-5059. IEEE.

Kelley, J.F. (1983). An empirical methodology for writing User-Friendly Natural Language computer applications. In *CHI'83 Proceedings*, ACM Press, pp. 193-196.

Kelley, J.F. (1984). An iterative design methodology for user-friendly natural language office information applications. *ACM Transactions on Office Information Systems*, Vol. 2(1), pp. 26-41.

Kilbrink, N. (2008). *Användningstester Plattformen*. Working paper in Swedish. Karlstads Universitet.

Klemmer, S. R., Sinha, A. K., Chen, J., Landay, J. A., Aboobaker, N. & Wang, A. (2000). SUEDE: A Wizard of Oz Prototyping Tool for Speech User Interfaces. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, pp. 1-10.

Krug, S. (2006). *Don't Make Me Think! A Common Sense Approach to Web Usability, Second Edition*. Berkeley: New Riders.

Lamberg, C. (2011). HTML5 for Ozlab. Student course report in pdf format available at http://www.kau.se/en/ozlab/research-and-development-projects/student-works

Lamberg, C. & Brundin, A. (2011). *Evaluating the future development options for Ozlab*. Bachelor thesis. Karlstad University. http://urn.kb.se/resolve?urn=urn:nbn:se:kau:diva-795

Larson, R. & Csikszentmihalyi, M. (1983). The experience sampling method. *New Directions for Methodology of Social and Behavioral Science*, 15, pp. 41-56.

Larsson, N. & Molin, L. (2006). Rapid prototyping of user interfaces in robot surgery. Presented at ISD´2005, Karlstad. Published in *Advances in Information Systems Development*, eds. A.G. Nilsson et al., Springer-Verlag, pp. 361-371.

Lee, S. Y., Mott, B.W. & Lester, J.C. (2010). Investigating Director Agents' Decision Making in Interactive Narrative: A Wizard-of-Oz Study. In *Proceedings of the Intelligent Narrative Technologies III Workshop (INT3 2010)*, Monterey, CS, USA, June 18. Article 13, 8 pp.

Leiser, R.G. (1989). Exploiting convergence to improve natural language understanding. *Interacting with Computers* 1(3), pp. 284-298.

Li, A.X. & Bonner, J.V.H. (2011). Improving control panel consistency of wizard of oz design and evaluation studies. In: *Proceedings of the 17th International Conference on Automation & Computing.* Chinese Automation and Computing Society, Huddersfield, UK.

Li, A.X. & Bonner, J.V.H. (2013). Using wizard-of-oz method to build multipurpose platform for domestic ambient media research and applications. *Multimedia Tools and Applications*, pp. 1-16 (on-line pre-publication 2013-03-13).

Li, X. (2012). Improving the Reliability and Validity of Wizard-of-Oz Methods. PhD Thesis. School of Computing and Engineering, University of Huddersfield. eprints.hud.ac.uk

Li, Y., Hong., J. I. & Landay, J. A. (2004). Topiary: A Tool for Prototyping Location-Enhanced Applications. *UIST'04*, New Mexico, USA, October 24-27, pp. 217-226.

Li, Y., Welbourne, E. & Landay, J.A. (2006). Design and Experimental Analysis of Continuous Location Tracking Techniques for Wizard of Oz Testing. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, Montréal, Québec, Canada, April 22-27, pp. 1019-1022.

Lim, Y.-K., Stolterman, E. & Tenenberg, J. (2008). The Anatomy of Prototypes: Prototypes as Filters, Prototypes as Manifestations of Design Ideas. *ACM Transactions on Computer-Human Interaction*, Vol. 15(2), article 7.

Lindström, M. & Nilsson, J. (2009). *Usability test report. Third pilot test of "Trust Evaluation". PrimeLife-project.*

Linnell, N., Bareiss, R. & Pantic, K. (2012). A Wizard of Oz Tool for Android. *MobileHCI'12*, San Francisco, USA, September 21-24, pp. 65-70.

Löwgren, J. & Stolterman, E. (2004) *Thoughtful Interaction Design: A Design Perspective on Information Technology.* MIT Press, Cambridge, Massachusetts.

Lu, D.V. & Smart, W.D. (2011). *HRI '11 Proceedings of 6th ACM/IEEE International Conference on Human-Robot Interaction*, March 8-11, 2011, Lausanne, Switzerland. ACM. Pp. 197-198. (Poster accessible via cse.wustl.edu/~dvl1/publications/polonius-poster.pdf .)

MacIntyre, B., Gandy, M., Dow, S. & Bolter, J.D. (2004). DART: A Toolkit for Rapid Design Exploration of Augmented Reality Experiences. *UIST '04 Proc. ACM Symp. User Interface Software and Technology*, ACM Press, pp. 197-206.

Magnusson, C., Anastassova, M., Tolmar, K., Pielot, M., Rassmus-Gröhn, K. & Roselier, S. (2009). The mobile Oracle: a tool for early user involvement. Poster presentation with extended abstract in *MobileHCI '09 Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services.* Article No. 84.

Mäkelä, K., Salonen, E-P., Turunen, M., Hakulinen, J. & Raisamo, R. (2001). 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*, pp. 115-119.

Malhotra, A. (1075). Design Criteria for a Knowledge-Based English Language System for Management: An Experimental Analysis. PhD Thesis, MIT Massachusetts Institute of Technology, Project MAC TR-146.

Marcotte, E. (2011). *Responsive Web Design.* A Book Apart, New York.

Maulsby, D., Greenberg, S. & Mander, R. (1993). Prototyping an intelligent agent through Wizard of Oz. In *ACM SIGCHI Conference on Human Factors in Computing Systems*, Amsterdam, The Netherlands, May, pp. 277-284.

Mavrikis, M. & Gutierrez-Santos, S. (2010). Not all wizards are from Oz: Iterative design of intelligent learning environments by communication capacity tapering. In *Computers & Education* 54 (3), pp. 641-651.

Molin, L. (2004). Wizard-of-Oz Prototyping for Cooperative Interaction Design of Graphical User Interfaces. *NordiCHI '04*, Tampere, Finland, October 23-27, pp. 425-428.

Molin, L. & Pettersson, J.S. (2003). How should interactive media be discussed for successful requirements engineering? In *Perspectives on multimedia: communication, media and technology*, eds. Burnett, Brunström & Nilsson. Wiley.

Munteanu, C. & Boldea, M. (2000). MDWOZ: A Wizard of Oz Environment for Dialog Systems Development. In *LREC*, Athens, Greece, May 31-June 2, pp.104-107.

Nielsen, J. (2010). Scrolling and Attention. Posted March 22, 2010, at Nielsen Norman Group, http://www.nngroup.com/articles/scrolling-and-attention/.

Nielsen, J. (2005). Time budgets for usability sessions. Posted September 12, 2005, at Nielsen Norman Group, http://www.nngroup.com/articles/ time-budgets-for-usability-sessions/.

Nilsson, J. (2005). *Interaktionsdesign av pedagogisk programvara. En experimentell studie av demonstrationer som hjälpfunktioner i ett övningsprogram för mellanstadiebarn.* Master Thesis Information Systems. Karlstad: Karlstad University.

Nilsson, J. (2006). *Användbarhetsutvärdering av H-RIB XM – Ozlabprototyp 2.* Internal development project report at Räddningsverket, Karlstad.

Nilsson, J. & Siponen, J. (2006). Challenging the HCI concept of fidelity by positioning Ozlab prototypes. Presented at ISD´2005, Karlstad. Published in *Advances in Information Systems Development*, eds. A.G. Nilsson et al., Springer-Verlag, pp. 349-360.

Oviatt, S. (1999). Ten myths of multimodal interaction. *Communications of the ACM* 42(11), pp. 74-81.

Peterson, E. (2005). *Web Analytics Demystified.* http://www.webanalytics demystified.com/downloads/Web_Analytics_Demystified_by_Eric_Pet erson.pdf.

Pettersson, J.S. (1996). Grammatological Studies: Writing and its Relation to Speech. Ph.D. dissertation. *RUUL #29.* Uppsala: Dept. of Linguistics, Uppsala University.

Pettersson, J.S. (1997). Framing the written sign. *Digital Creativity* 8(2), pp. 67-73.

Pettersson, J.S. (2002). Visualising interactive graphics design for testing with users. *Digital Creativity*, 13(3), pp.144-156.

Pettersson, J.S. (2003). Ozlab – a System Overview with an Account of Two Years of Experiences. In Pettersson, J.S. (ed.) *HumanIT 2003.* Karlstad: Universitetstryckeriet Karlstad. pp. 159-185.

Pettersson, J.S. & Nilsson, J. (2011). Effects of Early User-Testing on Software Quality – Experiences form a Case Study. In Song, W.W et al.

(eds.) *Information Systems Development*. Springer Science+Business Media, LLC. pp. 499-510.

Pettersson, J.S. & Siponen, J. (2002). Ozlab – a Simple Demonstration Tool for Prototyping Interactivity. In *NordiCHI*, Århus, Denmark, October 19-23, pp. 293-294.

Poschmann, P., Donner, M., Bahrmann, F., Rudolph, M., Fonfara, J., Hellbach, S. & Böhme, H-J. (2012). Wizard of Oz revisited: Researching on a tour guide robot while being faced with the public. In *RO-MAN, 2012 IEEE*, Paris, France, September 9-13, pp.701-706.

Riek, L.D. (2012). Wizard of Oz Studies in HRI: A Systematic Review and New Reporting Guidelines. *Journal of Human-Robot Interaction*, Vol. 1(1), pp. 119-136.

Robins, B., Dautenhahn, K., te Boekhorst, R. & Nehaniv, C.L. (2008). Behaviour Delay and Robot Expressiveness in Child-Robot Interactions: A User Study on Interaction Kinesics. *HRI '08*, March 12-15, 2008, Amsterdam, Netherlands. ACM. Pp. 17-24.

de Rosis, F., Cavalluzzi, A., Mazzotta, I. & Novielli, N. (2005) Can Embodied Conversational Agents Induce Empathy In Users? Presented at AISB'05: Social Intelligence and Interaction in Animals, Robots and Agents, 12-15 April 2005, Hatfield, UK. *Proceedings of the Joint Symposium on Virtual Social Agents*. pp. 65-72. http://www.aisb.org.uk/asibpublications/convention-proceedings; also http://www.di.uniba.it/intint/Humaine/H-Woz.html [2014-04-29]

Rösner, D., Frommer, J., Friesen, R., Haase, M., Lange, J. & Otto, M. (2012). LAST MINUTE: a Multimodal Corpus of Speech-based User-Companion Interactions. *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12),* Istanbul, Turkey, May 23-25. European Language Resources Association (ELRA). Pp. 2559-2566.

de Ruyter B., Saini P., Markopoulos P. & van Breemen A. (2005). Assessing the effects of building social intelligence in a robotic interface for the home. *Interacting with Computers* 17(5), pp. 522-541.

Salber, D. & Coutaz, J. (1993). Applying the Wizard of Oz Technique to the Study of Multimodal Systems. In Bass, Gornostaev & Unger (Eds.), *Human-Computer Interaction. Third International Conference, EWHCI '93, Moscow, Russia, August 1993, Selected Papers. LNCS* 753, Springer. Pp. 219-241.

Schlögl, S. (2011) Sketching Language: User-Centered Design of a Wizard of Oz Prototyping Framework. Human-Computer Interaction - *INTERACT 2011,* 13th IFIP TC 13 International Conference, Lisbon, Portugal, September 5-9, 2011, Proceedings, Part IV. *Lecture Notes in Computer Science Volume 6949*, 2011, pp. 422-425.

Schlögl, S., Doherty, G., Karamanis, N. & Luz, S. (2010). WebWOZ: A Wizard of Oz Prototyping Framework. *EICS'10*, Berlin, Germany, June 19-23, pp.109-114.

Schlögl, S., Doherty, G., Karamanis, N., Schneider, A. & Luz, S. (2010). Observing the Wizard: In Search of a generic Interface for Wizard of Oz Studies. In *Proceedings of iHCI*, Dublin, Ireland, September 2-3, pp. 43-50.

Schlögl, S., Schneider, A., Luz, S. & Doherty, G. (2011). Supporting the Wizard: Interface Improvements in Wizard of Oz Studies. In *Proceedings of the 25th BCS Conference on Human-Computer Interaction*, Swinton, United Kingdom, pp. 509-514.

Schlögl, S., Chollet, G., Milhorat, P., Deslis, J., Feldmar, J., Boudy, J., Garschall, M. & Tscheligi, M. (2013). Using Wizard of Oz to Collect Interaction Data for Voice Controlled Home Care and Communication Services. In *Proceedings of the IASTED International Conference*, Innsbruck, Austria, February 12-14, pp. 511-518.

Schlögl, S., Doherty, G. & Luz, S. (2014). Wizard of Oz Experimentation for Language Technology Applications: Challenges and Tools. *Interacting with Computers*, Advance Access published May 9, 2014.

Sefelin, R., Tscheligi, M. & Giller, V. (2003). Paper Prototyping – What is it good for? A Comparison of Paper- and Computer-based Low-fidelity Prototyping. Short talk presented at *CHI 2003*, April 5-10, 2003, Fort Lauderdale, USA, pp. 778-779.

Segura, V. C. V. B. & Barbosa, S. D. J. (2013). UISKEI++: Multi-Device Wizard of Oz Prototyping. *EICS'13*, London, United Kingdom, June 24-27, pp. 171-174.

Serrano, M. & Nigay, L. (2010). A wizard of oz component-based approach for rapidly prototyping and testing input multimodal interfaces. *Journal on Multimodal User Interfaces*, 3(3), pp. 215-225.

Serrano, M., Juras, D. & Nigay, L. (2008). A three-dimensional characterization space of software components for rapidly developing multimodal interfaces. In: *Proceedings of ICMI'08*. ACM, New York, pp. 149–156.

Siegel, G.M., Lenske, J. & Broen, P. (1969). Suppression of normal speech disfluencies through response costs. *Journal of Applied Behavior Analysis*, 2, pp. 265-276.

Siponen, J., Pettersson, J. S. & Alsbjer, C. (2002). *Ozlab Systembeskrivning.* Arbetsrapport. Karlstads Universitet: Institutionen för informations-teknologi.

Skantze, G. & Hjalmarsson, A. (2010). Towards incremental speech generation in dialogue systems. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL '10)*, Tokyp, Japan, September 24-25, 2010. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 1-8.

Sohn, T.Y. & Dey, A.K. (2003). iCAP: An Informal Tool for Interactive Prototyping of Context-Aware Application. *CHI 2003* (Poster, Extended abstract), pp. 974-975.

Spindler, M., Weber, M., Prescher, D., Miao, M., Weber, G. & Ioannidis, G. (2012). Translating Floor Plans into Directions. In *Computers Helping People with Special Needs. Part II.* Springer Berlin Heidelberg. Pp. 59-66.

Standage, T. (2002). *The Turk: The Life and Times of the Famous Eighteenth-Century Chess-Playing Machine.* Walker & Co, Bloomsbury. Also published as *The Mechanical Turk: The True Story of the Chess-Playing Machine that Fooled the World.* Allen Lane the Penguin Press.

Suchman, L. (1987). *Plans and Situated Actions: The Problem of Human-Machine Communication.* Cambridge University Press.

Sutton, S., Cole, R., et al. [15 authors] (1998). Universal Speech Tools: the CSLU Toolkit. *Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP'98)*, Sydney, Nov 30 - Dec 4, 1998. Pp. 3221-3224.

Tennant, H.R. (1981a). Evaluation of Natural Language Processors. PhD Thesis, University of Illinois at Urbana-Champaign.

Tennant, H. (1981b). *Natural Language Processing. An Introduction to an Emerging Technology.* PBI Petrocelli Books Inc., New York and Princeton.

Uceta, F.A., Dixon, M.A. & Resnick, M.L. (1998). Adding interactivity to paper prototypes. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, Vol. 42(5), pp. 506-510.

de Vicente, A. (1998) Book review: Humans, Computers, and Wizards. Analysing human (simulated) computer interaction by R. Wooffitt, N.M. Fraser, N. Gilbert, and S. McGlashan. *ReCALL* 10(2), pp. 80-82.

Ward, W., Cole, R., Bolaños, D., Buchenroth-Martin, C., Svirsky, E., Van Vuuren, S., Weston, T., Zheng, J. & Becker, L. (2011). My science tutor: A conversational multimedia virtual tutor for elementary school science. *ACM Transactions on Speech and Language Processing (TSLP)*, Vol. 7 (4), Article No. 1, 29 pages. www.bltek.com/our-work/projects/ies-myst/

Webb, N., Benyon, D., Bradley, J., Hansen, P. & Mival, O. (2010). Wizard of Oz Experiments for a Companion Dialogue System: Eliciting Companionable Conversation. *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC2010)*, Valletta, Malta. Pp. 875-879. http://www.companions-project.org

White, K.F., & Lutters, W.G. (2003). Behind the Curtain: Lessons Learned from A Wizard of Oz Field Experiment. *ACM SIGGROUP Bulletin*, 24(3), pp. 129-135.

Wik, M. (2014). Interactive In-Situ Requirements Gathering: Extending Beyond Questionnaires and Interviews. Poster presentation at *SIDER'14 Student Interaction Design Research conference,* 11-12 April 2014, Stockholm (sider2014.csc.kth.se/wp-content/uploads/sites/8/2014/04/sider14_ submission_11.pdf)

Wooffitt, R., Fraser, N., Gilbert, N. & McGlashan, S. (1997). *Humans, Computers, and Wizards. Analysing human (simulated) computer interaction.* Routledge.

Zachhuber, D., Grill, T., Polacek, O. & Tscheligi, M. (2012). Contextual Wizard of Oz. In *Ambient Intelligence*, Springer Berlin Heidelberg, pp. 224-239.

Perspectives on Ozlab in the cloud

The Wizard-of-Oz method has been around for decades, allowing researchers and practitioners to conduct prototyping without programming. The extensive literature review in the field reported here, however, revealed that the re-usable tools supporting the method do not seem to last more than a few years. Generic systems started to appear around the turn of the millennium, but very few are still in use. New systems are designed nevertheless. The systems and issues presented here should be of interest to people in the field of prototyping interaction design.

This review was inspired by the authors' ongoing re-development of their own Wizard-of-Oz tool, the Ozlab, into a system based on web technology. The report takes stock of some key features of Ozlab as well as reviews and contrasts other re-usable Wizard-of-Oz tools with the ambition to list every generic tool.

The introductory chapter compares and contrasts prototyping in general with Wizard-of-Oz prototyping and provides an historical overview of Wizard of Oz in the development of digital interactive systems, spanning the years 1971-2013. Chapter 2 briefly describes the operation of Ozlab, and Chapter 3 presents the literature review of generic WOz tools. Chapter 4 discusses how interaction is supported by WOz tools and Chapter 5 how platform dependency affects the longevity of generic tools, while Chapter 6 points to the limitations in the Wizard-of-Oz method itself from several perspectives. Chapter 7, finally, presents concluding remarks including a list of points for future methodological analysis and development.

Faculty of Arts and Social Sciences

WORKING PAPER | August 2014