EFFICIENT COMBINATION OF TYPE-IN AND WIZARD-OF-OZ TESTS IN SPEECH INTERFACE DEVELOPMENT PROCESS

Saija-Maarit Lemmelä and Péter Pál Boda

Nokia Research Center, Speech and Audio Systems Laboratory, Helsinki, Finland
(saija.lemmela; peter.boda@nokia.com)

ABSTRACT

Effective methods and tools are needed for designing, developing and testing speech-enabled applications. Traditional methods have been collecting natural dialogues, using Wizard-of-Oz experiments and different type-in tests. Collecting and analysing natural dialogues is always time consuming and in many situations even impossible. Wizard-of-Oz studies need well defined test scenarios and a lot of time to collect and analyse data. Type-in studies are one possibility to collect big amount of data quickly and analyse it semi-automatically. This paper discusses Wizard-of-Oz and type-in data collection methods as implemented at our site. The test set-ups and the identified discrepancies are discussed in details. As our study indicates, testing an initial version of a dialogue flow and the corresponding grammar with the type-in method offers quite reliable measures about the final implementation of the interface for a speech-enabled navigation application.

1. INTRODUCTION

The development of a reliable, high quality speech interface is an iterative process which requires time, resources and effective methods. Suitable tools can help the developer to shorten the development cycle and to provide speech-enabled prototypes that require only a small number of iteration cycles. The basic development process needs supporting tools to:

- collect words that form the vocabulary;
- find out commonly used grammatical expressions;
- identify interaction patterns;
- give ideas for designing prompts and effective feedback mechanism;
- ensure the functional requirements of a system.

There are well known, widely used methods to collect user data prior implementing a speech-enabled prototype. The situation is simplest and most straightforward when an already existing human-operated application is to be enhanced with a speech interface. In that case the developer needs to observe users’ behaviour and collect all possible spoken inputs. In most of the cases a human-operated application does not exist. In this kind of situation the developer can simulate the application by a Wizard-of-Oz (WOZ) approach, which is a smart way to test users’ reaction when interacting with a “virtual” application. The main advantage of this approach is that spoken interaction can be assessed, realistic user inputs can be gathered and analysed. On the other hand, a vital and demanding task of a Wizard is to design realistic scenarios which constrain the subject as tightly as possible to the application domain of the future system, yet, giving him/her as much motivation and freedom in terms of linguistic expressions as possible. The subject is free to decide what to ask, in what order and what wording fits best at certain stages of the dialogue [1]. Error situations and forcing the users to exhibit various dialogue acts are especially challenging.

Type-in tests are another way of assessing the status of an early implementation. Both off-line and on-line set-ups are possible. The former case refers to a data collection when the users are asked to fill up a questionnaire by using an imaginary application and interacting with it. The major disadvantage of this approach is the minimal control over the user, however well it is explained in the questionnaire. The on-line data collection differs entirely from the former case. Once a first version of a dialogue flow is fixed, depending on the development environment, the developer can run the application by requesting typed-in user inputs only. This is a perfect way to eliminate erroneous speech recognition performance, however, the challenge is to achieve similar linguistic quality with the type-in input than with spoken input.

The main goal of this paper is to provide insights to these two different approaches, namely to Wizard-of-Oz and on-line type-in tests, and to indicate how the achieved results support each other and contribute directly to the implementation of the final speech-enabled version.

The paper is organised as follows. Section 2 explains the two methods, their technical implementations and particularities. Section 3 describes the experiments conducted for a selected application. Section 4 summarises the achieved results and outlines future directions.

2. DATA COLLECTION METHODS

2.1. Wizard-of-Oz platform

When designing a spoken dialogue system, it is desired to formulate a detailed specification of how the future system is expected to behave [1]. A human operator, who simulates the system as precisely as possible, can then test the plan with potential users.

The main purpose of an experiment is to identify major design flaws of the plan and to gather useful information about users’ interaction with the system. This information can be utilised in later stages of the design and implementation process, namely, when the grammar and the vocabulary of the
speech recogniser will be fixed, the system prompts will be selected and the dialogue flow of the interaction will be implemented.

It is believed that a unified tool to carry out Wizard-of-Oz (WOZ) testing task and log all the necessary information obtained for later analysis is essential. Our WOZ-tool has the following features:

- Easy re-configuration for any application. It includes the functionality to create and modify different setups with pre-defined sets of prompts;
- Available system prompts are displayed graphically and it is easy to navigate through them. Prompts can be grouped, thus parts of a dialogue can be defined beforehand;
- During tests the Wizard can select the system prompts in any order and play them back;
- Text-to-speech engine generated prompts can be created and saved by the tool when configuring an application or dynamically during actual tests;
- The system is connected to a telephone line;
- Both the incoming and the outgoing speech signals can be recorded and saved for later analysis;
- Stored recordings can be listened back and analysed by the Wizard.

In our implementation the following groups of prompts are defined: “Welcome”, “Bye”, “Error” and “Help”. General, frequently used prompts that are useful across applications are “You said”, “What next?” and a beep that indicates to users that it is now their turn to talk.

### 2.2. On-line type-in approach

In the past five years we have mainly focused on telephony application development. However, recently also other platforms are in our interest and the need to have a unified approach to develop speech-enabled applications arose. In 1997 we implemented a simple dialogue manager that enabled users to access a web-based interface of a city bus timetable database [3]. That early version of the dialogue manager is now developed further and it provides a parameterised approach with basic features of system-driven interaction for various applications on various platforms [4].

The Nokia Dialogue Engine (NoDE) provides a parametric definition of a dialogue. The developer fills up a sort of style sheet and defines what type of dialogue is needed, how verification is handled, what prompts are used for subsequent acquisition of user input, etc. A web-based user interface for NoDE, Remote Nokia Dialogue Engine (rNoDE), was created for testing and developing dialogues designed for speech-enabled applications. Two versions, one for developers and one for test users are available. Both can be used in two modes:

- using only a dialogue definition without any grammar;
- using a dialogue definition and corresponding grammar(s).

### Developer’s version of rNoDE

The Developer’s version of rNoDE is primarily meant for dialogue and grammar creation. It gives the developer a possibility to implement iteratively the structure of the dialogue and the corresponding grammar(s). If a developer is using rNoDE in the first mode, he/she is forced to enter the input in a “semantic_key=value” format. For example, in a navigation application a valid input might be: “DIRECTION=left; AMOUNT=two”. The other mode supports the use of finite state grammars and now the user input must conform to the pre-defined grammar(s). This mode is useful when measuring the coverage of the used grammar(s). A good practice is to experiment the dialogue first without grammar(s), in order to tune the desired dialogue flow, and only after that to continue the process by developing grammar(s).

### Tester’s version of the rNoDE

The Tester’s version of the rNoDE can be used to collect large amount of user data. The version enables test users to test the application remotely via the web interface whenever it suits best to them. This gives better chances to get more persons involved in the data collection and the tests. The whole conversation between a user and the system is saved into a log file that can be processed semi-automatically by the developer.

Figure 2 below shows the web interface of rNoDE for testers.

### 3. EXPERIMENTS

We chose the navigation application implemented on the Nokia Mediaterminal [5] as a show case for our experiments. An earlier study already indicated how potential users take advantage of different input modalities, namely remote control and speech-enabled navigation [6]. The conclusion from that study was that people preferred to use shortcuts supported by
the speech interface. Furthermore, speech input was more effective since it took less steps than navigation. The subjective data however revealed that subjects like to use the remote control, therefore a combination of speech input and traditional navigation, a complementary multimodal interface was suggested.

3.1. Tested functionalities

It is shown that spoken and written language differs from each other in many ways [7], but this kind of research is often limited to formal situations. The target of our experiment was to detect linguistic variability and richness of user expressions in the control task of the speech-enabled graphical menu by using both type-in and WOZ approaches. The graphical menu structure is a two dimensional cross of bars, consisting of folders on the horizontal bar and bookmarks on the vertical bar. A snapshot of the Navi™Bars is shown in the figure below.

Figure 2: The Navi™Bars has folders on the horizontal bar and bookmarks on the vertical bar. On the vertical bar the bookmarks of the folder in the intersection are shown.

The test scenario included the following basic functionalities:

- showing and hiding the menu structure;
- moving up, down, left or right inside the menu structure;
- moving into a specified position within the menu structure. A position can be defined by a name of a bookmark or a folder;
- activating a bookmark.

3.2. Test scenarios

Data was collected first by using the type-in method. The users received the test scenario descriptions and were asked to access the test site remotely. The graphical menu structure was explained to test users but it was not displayed during the type-in tests.

The WOZ studies were conducted to detect how well the utterances collected via these type-in tests reflect a more realistic situation, when spoken input can be used and the system feedback is visual.

The first type-in tests were conducted (Type-in_1) with 13 participants having no previous experience in using Navi™Bars. In the second phase (Type-in_2) five persons having previous experience with Navi™Bars completed the type-in tests. In both phases only 2-2 participants were female and from the 18 persons only one had non-technical background. The WOZ studies were conducted with seven persons (2 females and 5 males). Five of them did not have any previous knowledge about Navi™Bars and two of them had never used a speech-enabled application. Testers were not aware of how a WOZ set-up is implemented.

3.3. Results

It was noticed that subjects are using rather similar linguistic expressions in both WOZ and type-in methods. The average lengths of utterances were 2.11 words for type-in and 2.71 words for the WOZ case. In practice this means 2-3 words/utterance, which is indicative for the speech recognition engine to be used. The results of the objective analysis for all the three tests can be seen in the Table 1. It is interesting to notice the equal amount of different words in all cases. This implies that a rather limited vocabulary is adequate for the navigation task.

<table>
<thead>
<tr>
<th></th>
<th>WOZ</th>
<th>Type-in_1</th>
<th>Type-in_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of commands collected</td>
<td>219</td>
<td>299</td>
<td>191</td>
</tr>
<tr>
<td>Average length of commands (words/utterance)</td>
<td>2.71</td>
<td>2.29</td>
<td>1.84</td>
</tr>
<tr>
<td>Number of different words</td>
<td>59</td>
<td>58</td>
<td>58</td>
</tr>
</tbody>
</table>

Table 1. Results from WOZ and type-in tests. (NB.: names of folders and bookmarks are not included in the number of different words.)

The results of the WOZ and type-in tests carried out by people not having any previous experience with the menu structure can be divided according to the basic functionalities of the Navi™Bars. The derived measures are shown in Table 2.

<table>
<thead>
<tr>
<th></th>
<th>Number of utterances</th>
<th>Number of different words</th>
<th>Av. Number of words/utterance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>W</td>
<td>T</td>
<td>W</td>
</tr>
<tr>
<td>Showing/hiding menu</td>
<td>22</td>
<td>15</td>
<td>13</td>
</tr>
<tr>
<td>Moving to direction</td>
<td>72</td>
<td>37</td>
<td>19</td>
</tr>
<tr>
<td>Moving to destination</td>
<td>66</td>
<td>61</td>
<td>19</td>
</tr>
<tr>
<td>Activating</td>
<td>37</td>
<td>55</td>
<td>15</td>
</tr>
</tbody>
</table>

Table 2. Results from WOZ (W) and type-in (T) tests, divided according to the basic Navi™Bars functionalities.
The following differences were found when the WOZ test results were compared to the type-in results:

1. During the WOZ study 13 totally new words (e.g. 'then', 'you', 'now') were found. These words appeared only in 6% of the user utterances.
2. Difference in the frequency of the applied words. When testers wanted the Menu to be displayed they typically used ‘Show Menu’ expression in the type-in test, while during the WOZ test they tended to say ‘Close’ to the currently active service. Also, a command ‘Open folder’ was meaningful during WOZ tests but was not used in the type-in case.
3. Subjects tended to use two-word utterances in the type-in tests; typically a command verb (e.g. ‘go’, ‘activate’, ‘show’) and a reference (e.g. ‘this’, ‘it’ or a name of a folder/bookmark). In the WOZ study they used more eagerly slightly longer utterances and defined action’s target more clearly (‘Open this link’, ‘Select TV folder, ‘Open channel TV3’).
4. Test users referred a couple of times to the previous states during the WOZ test but not during the type-in test (‘Go back to menu’).
5. Scrolling was more popular in the WOZ case, where the testers were supposed to find some specific bookmark from the menu (‘Scroll right’). In this kind of situations users preferred commands (e.g. ‘right’) that automatically moved the folders to a specified direction until they gave a new command.
6. The language used in the WOZ test was slightly more natural than the language collected in the type-in case. For example, one tester started almost every second utterance with ‘OK’. In some cases users changed their mind during a command (‘Right, go left, go right’). Also, some unintentional, spontaneous utterances appeared.

Example utterances collected from both test set-ups can be seen in Table 3. There has been studies suggesting that translating graphical interfaces into speech-enabled ones is not likely to result in an effective solution [2]. In our tests we observed that certain expressions are certainly triggered by the graphical interface itself. Especially points 2., 3. and 5. in the above list are indicating that visual feedback during the WOZ studies were affecting the user utterances to certain extent.

### 4. CONCLUSIONS

The results presented in this paper indicate that testing an initial dialogue flow implementation and grammar(s) with on-line type-in method offers quite reliable measures about the quality of the realisation. Type-in tests followed by WOZ type of validation allows developers to detect expressions specific only to spoken input. Furthermore, if not only speech modality is used, linguistic expressions triggered by the other modality, in our case visual feedback, can also be detected.

As we found, the required modifications in the grammar after the WOZ test are only small (but important) additions and not radical changes. The dialogue flow required no modifications after the WOZ test.

We believe that by tuning an early implementation with type-in test and finalising it with WOZ experiment can avoid major flops when adding speech recognition to the user interface. Our plan for the future is to continue using type-in and WOZ approaches as vital parts of the application development process.

**Acknowledgements:** The authors want to thank Dari Trendafilov and Vétek Ákos for their contribution to this work. We also appreciate the support of Aseel Ibrahim and Bengt Ahlström (Nokia Home Communications, Linköping).

### 5. REFERENCES


---

**Table 3.** Typical user utterances with significant differences in occurrences between the WOZ and the type-in tests. (direction=left/right/up/down/back, object=bookmark/that/folder/this/it, amount=one/two/three/four).