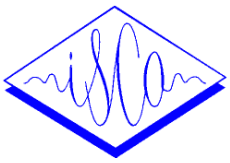

ISCA & IEEE Workshop on Spontaneous
Speech Processing and Recognition
Tokyo Institute of Technology, Tokyo, Japan
April 13-16, 2003

EXTREME EXPERIMENT (XE) METHOD FOR DEVELOPING MIXED INITIATIVE DIALOGUE SYSTEMS

Masahiro Araki, Johji Kurahashi, Takeo Matsumoto

Kyoto Institute of Technology

ABSTRACT

In this paper, we propose a new spontaneous speech collection method and a methodology for rapid test-and-modify development of mixed initiative spoken dialogue systems. Our method uses VoiceXML environment for helping WOZ experiments and uses tool suite for analyzing the user's utterance and for giving feedback to the systems. The core techniques of our method are (1) bootstrapping by Web application framework, (2) WOZ data collection using VoiceXML platform and (3) log analyzing tools for feedback.

1. INTRODUCTION

In recent years, a number of spoken dialogue systems are developed for commercial use, such as flight ticket booking, inquiry of stock prices, sight seeing guide, etc. Almost all systems are designed as a system initiative dialogue system which restricts user's utterance in a word or a simple phrase. These systems are accepted by frequent callers, but they seem to be unwieldy for novice customers.

Considering current level of speech recognition and natural language processing, the goal of spoken dialogue systems, in a near future, is to accept user initiative, spontaneous utterances at the specific dialogue states. For example, in a voice portal site, it is convenient for user to speak spontaneously, *e.g.* "Tell me a weather forecast of Kyoto" than to select "weather" in the main menu and, after that, to select "Kyoto" in a prefecture menu. If the information in such a spontaneous input is not enough for the system to reply, the system takes initiative for additional or clarification questions. Therefore, such systems necessarily behave in a mined-initiative manner.

In order to develop such kind of mixed initiative dialogue systems, it needs some amount of user's spontaneous speech for making grammar rules or learning N-gram language model. At the early stage of system development, Wizard of Oz (WOZ) method [1] is the only way for collecting such user utterances. However, in general, simple WOZ method costs large amount of money and is time consuming.

In this paper, we propose a new user utterance collection method and a methodology in rapid test-and-modify development of user initiative or mixed initiative spoken dialogue systems.

In the same motivation of our agile system development, Degerstedt *et.al.* proposed a method for iterative implementation of dialogue systems [2]. They applied their method to the dialogue module modification, not to the input grammar nor dialogue state transition patterns. Therefore, their method does not include the user test phase. Also, they use their own dialogue tools for the implementation.

Our method uses standard VoiceXML [3] environment for helping WOZ experiments and tool suite for analyzing the user's utterance and giving feedback to the system. It enables test-oriented, agile, iterative development of spoken dialogue systems.

Section 2 shows the actual work for collecting spontaneous utterances using usual WOZ method and explains its difficulty. Section 3 proposes our development methodology, i.e. extreme experiment. Section 4 elaborates our methodology. Section 5 states conclusions and future works.

2. DIFFICULTY IN SPONTANEOUS SPEECH COLLECTION

In this section, we describe our past dialogue collection which aimed to collect spontaneous speech at the specific dialogue states and explain how difficult it was.

In order to analyze the pattern of user initiative utterances in the early stage of development of spoken dialogue systems, we collected human-to-human dialogue at given task. One participant of the dialogue plays a roll of operator who searches for the information and gives advice to another participant, called user. We controlled the operator following dialogue structure shown in Figure 1.

For the sake of naturalness of dialogue, the operator was permitted to deviate from the given structure. However, in order to collect user initiative utterance, the operator did not take initiative when the user was considering the question.

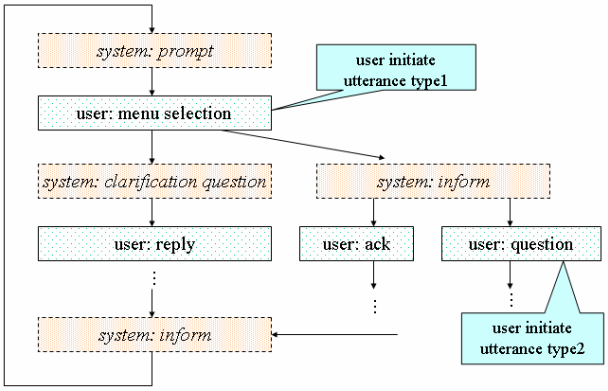

Figure 1 Dialogue structure of the WOZ setting.

The task of this experiment was guide for mountain climbing. We prepared the knowledge of the place which is suitable for one day hiking. The menu items were sightseeing information, course guide, access, facilities for recreations, etc. Typical dialogue began by user's utterance which described user's wish. The operator took it as one of menu selection utterance and fell down to the menu.

30 subjects conversed with one operator and it took five to ten minutes for each dialogue. The speech was recorded by DAT and transcribed following the guideline proposed by the Japanese Discourse Resource Initiative [4]. We annotated each utterance with the name of the dialogue states given in Figure 1. Typical user initiative utterance appeared in the states of the menu selection and user question. The collected user initiative utterances are summarized in Table 1.

Table 1 Types of user initiative utterances.

Class	Uttr. type	Number of uttr.
Locatable at the structure	Understandable with context	43
	Understandable without context	57
Not locatable at the structure		154
Sum		254

In this experiment, we got 100 user initiative utterances which can be dealt with the target spoken dialogue system. In these 100 utterances, 43 were fragmental utterances which needs dialogue context in interpreting it. Therefore, in 30 dialogues, we got only 57 utterances which are useful for designing the grammar and the behavior of user initiated states of dialogue.

Another problem was the amount of the out-of-structure utterances (154) which deviate the states given in Figure 1. Even by well-trained operator and under restricted behavior, it was very difficult for extracting manageable user initiated utterances.

3. PROPOSAL OF EXTREME EXPERIMENT METHOD

3.1. Drawback of waterfall model

The experiment explained in section 2 presupposes the waterfall system development model. In case of using VoiceXML as a platform of a spoken dialogue system, possible waterfall system development model is shown in Table 2. In this table, usual software development stages are also listed.

Table 2 Waterfall development model.

Stage	Software system	Spoken dialogue system
1	Requirement analysis	Decide the functions of dialogue system
2	External specification	Design the interface to the back-end applications
3	Internal specification	Map dialogue state to each VoiceXML file
4	Implementation	Write VoiceXML files and grammar files
5	Test	Usability test

The dialogue data collection is located before the stage 1 (requirement analysis). Based on the dialogue data, system requirements, a set of dialogue states, system's prompts and the grammar of user's utterance are decided. In other words, although data collection affects almost all the stages of the waterfall process, the settings of dialogue collection have to be decided at the very early stage of the development.

In practice, this waterfall model development is suitable for well-designed system initiative dialogue system, for example, adding spoken interface to the ticket booking Web site. However, considering the difficulty in collecting user initiative utterance and the risk factors, the waterfall model cannot apply to the development of mixed initiative or user initiative dialogue systems.

3.2. Extreme Programming for software development

For the small scale, agile project, the extreme programming (XP) method [5] is proposed as an alternative of the waterfall model. The essences of XP are as follows.

1. Risk-driven development
   Risky part of the system should be developed first.
2. Short term and frequent release of the product
   One or two weeks are the term for the new release.

3. Test-centered programming
   Test case should be made before the programming and every unit should be built everyday.

These points are ported to our extreme experiment method explained in the next subsection. Other practices of XP are pair programming, using design pattern, 40 hours working time per week, etc.

3.3. Extreme Experiment for spoken dialogue system development

As a spoken dialogue development version of XP, we have developed extreme experiment (XE) method. The idea of XE is repeatedly short term development and WOZ testing. XE is suitable for developing mixed initiative dialogue systems on which usability test can affect almost all the stages of the development. The essences of XE are as follows.

1. Application based bootstrapping
   Most risky part of the spoken dialogue system is an interface to the backend application server. XE uses framework method, such as [6][7], for prototyping an initial product. Such framework can generate typical interaction patterns for slot-filling, database search and explanation type task.
2. Short term, frequent release of the product
   The initial product should be improved by the test-and-modify cycle. VoiceXML platform and standard server side Java resources enable such short term improvement.
3. Test and data collection by WOZ
   By adding a WOZ support add-on into the product, product test and data collection can be done simultaneously. Also, by the aid of log analyzing tools, the developer can easily modify the dialogue patterns and grammar files.

An outline of XE method is illustrated in Figure 2.

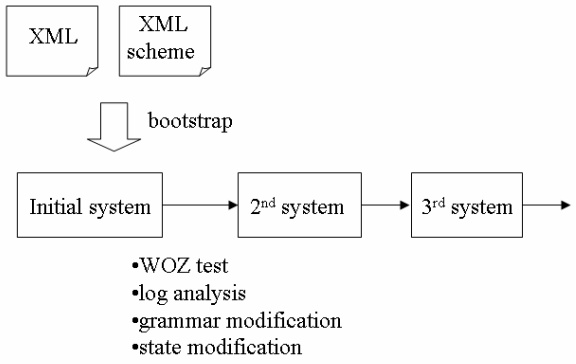

Figure 2 Outline of eXtreme Experiment (XE).

4. DETAILS OF EXTREME EXPERIMENT

4.1 Bootstrapping using Web framework

We have developed automatic spoken dialogue generation methods, i.e. XML-to-VoiceXML converter [6] and XML scheme to VoiceXML converter [7], which can be used for bootstrapping in XE. Currently, we are developing database search framework using Struts [8] (Figure 3). In Struts, the core control unit (action servelet) is provided by the Model-View-Controller framework. In our conversion method, the action form bean and action configuration file, which defines the system's reaction according to the return status of action class program, can be generated from XML scheme language of the given task.

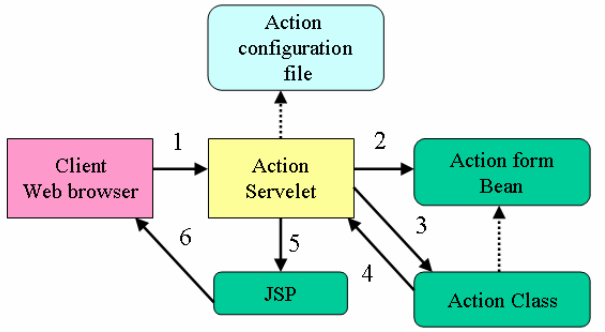

Figure 3 Struts framework for Web application.

4.2 WOZ support add-on

For WOZ testing, a WOZ support add-on element is attached to every user initiative part of VoiceXML file (i.e. <initial> element). Every <initial> element has additional DTMF (Dual Tone Multiple Frequency) grammar which can select system's behavior. In menu selection phase of dialogue, each DTMF number corresponds to the each menu item. In multiple slot-filling phase of dialogue, possible subsets of test scenario (i.e. the values of variables) are embedded into the DTMF selection. If the user inputs an out-of-grammar utterance and the speech recognition part rejects the input, a short sound let the wizard know this situation and the wizard inputs the corresponding DTMF input. By such operation, the user does not realize his/her utterance is rejected. The system continues the dialogue as if the utterance is normally accepted.

Therefore, because the wizard assists the dialogue system in difficult dialogue state, we can easily collect long dialogue for each user.

The key point of this test setting is the accuracy of the detection of OOG utterances. In our preliminary experiment, using the confidence measure of Voice Web Server provided by Nuance Corp., we can distinguish large part of OOG utterances if we customize the reject threshold value in a preparatory experiment. We compare

the confidence score of in-grammar utterances and out-of-grammar utterances using statistical language model of Voice Web Server, SayAnything[tm]. The threshold value is determined by the difference of the average of five types of utterances (2 to 3 content words) for each subject. “In” means in-grammar word sequence and “near” means that we substitute one content word by the word which has same vowels. The results are shown in Table 3.

Table 3 Adapted confidence score and its error rate.

	1	2	3	4	5
in	75.8	72.7	63.3	73.5	70.4
near	57.6	58.3	52.8	64.9	54.0
threshold	62	69	56	67	62
Error rate	8%	8%	12%	16%	8%

4.3 Log analyzing tools

In order to feedback the results of WOZ experiment, we have developed log analyzing tools for the Voice Web Server. The log analyzer outputs the dialogue log in the form of dialogue corpus with the confidence score, the link to the recorded audio file and the system status (accept or reject). The example of dialogue log is shown in Figure 4.

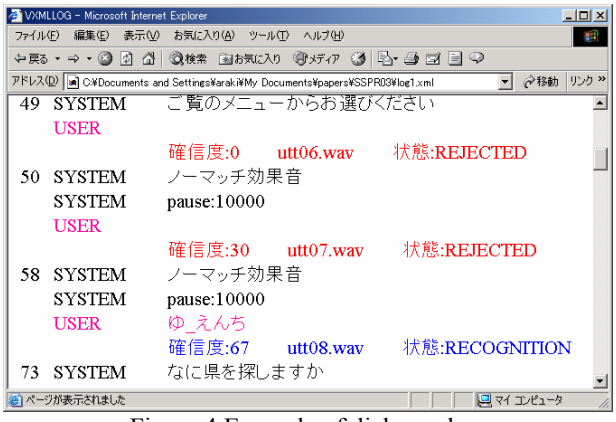

Figure 4 Example of dialogue log.

We can transcribe rejected user’s utterance in the log file (XML format). Using this transcription and the type of next system’s utterance, we can use these logs for statistical or probabilistic grammar learning.
The other log analyzing tool is to make VoiceXML file which reproduce the dialogue. The VoiceXML file is made by time stamp, pause information, system’s utterance by TTS and user’s utterance wave file. If we reflect the modification of dialogue system to this revive data, we can evaluate the effect of the modification by simulation.

5. CONCLUSION

In this paper, we proposed the XE method for spontaneous speech collection and the development of mixed initiative spoken dialogue systems.
We plan to extend XE as follows.

1 Incorporating a statistical utterance classification module with parameter modification according to the support input by the wizard.
2 Incorporating test-first practice of XP. The test means dialogue examples made by user. We are trying to realize automatic dialogue system generator form dialogue examples [9].

6. REFERENCES

[1] F.M. Norman and G. N. Gilbert, Simulating speech systems. Computer Speech and Language 5: 81-99, 1991.
[2] L. Degerstedt and A. Jonsson, A method for Iterative Implementation of Dialogue Management, In Proc of IJCAI workshop on Knowledge and Reasoning in Practical Dialogue Systems, 2001.
[3] VoiceXML Forum, Voice eXtensible Markup Language (VoiceXML™) version 1.0, http://www.w3.org/TR/voicexml/ 2000.
[4] The Japanese Discourse Resource Initiative, Japanese Dialogue Corpus of Multi-level Annotation, In Proc. of 2nd SIGDial Workshop 2000.
[5] K. Beck, eXtreme Programming Explained – Embrace Change, Addison-Wesley, 1999.
[6] M. Araki, T. Ono, K. Ueda, T. Nishimoto, and Y. Niimi, An Automatic Dialogue System Generator from the Internet Information Contents, Proc, EUROSPEECH2001, pp1150-1156, 2001.
[7] M. Araki, Automatic Generation Method of Spoken Dialog Systems from XML Scheme Language (in Japanese), Information Technology letters, Vol.1, pp.97-98, 2002.
[8] The Apache Jakarta Project, Struts, http://jakarta.apache.org/struts/, 2002.
[9] K. Otani, M. Araki, T. Nishimoto, and Y. Niimi Development of mixed initiative spoken dialogue systems from dialogue corpus (in Japanese), Technical report of SIG-SLUD-A103-04, JSAI, pp.21-26, 2002.