RadioDoc: A VOICE-ACCESSIBLE DOCUMENT SYSTEM

Takuya NISHIMOTO, Masahiro ARAKI and Yasuhisa NIIMI
Department of Electronics and Information Science
Kyoto Institute of Technology, Japan
nishi@vox.dj.kit.ac.jp

ABSTRACT

We propose a voice-accessible document system and the document format called RadioDoc, which will be simple for both novice and expert users to use.

As novice users often have no knowledge of the available commands or hesitate to use them, the RadioDoc system is designed so that they need not use any speech input at all. Expert users, comfortable with voice command, may move back and forth within the document as with CD-player, or use the voice shortcut where the system jumps ahead to the specific information immediately without going through the entire program. We also designed the RadioDoc language and created a tool which converts RadioDoc documents to VoiceXML applications. As the results of our experiments, the success of our system is found in the wide range of the expertise of the users. The development process using RadioDoc language is also turned out to be productive.

1. INTRODUCTION

VoiceXML (Voice eXtensible Markup Language) [1] is expected to be a new standard used for making the Internet information system accessible via voice over the telephone. Telephony user interface which uses speech recognition is expected to be powerful because it is as natural as our talk to and with each other. It is, however, said that the telephony interface is especially sensitive to human factors and user interface design. Because the visual display does not exist, the poor dialog design in telephony interface cannot be compensated for [2].

Though VoiceXML has the capability of writing a variety of interaction styles using voice input and output, the usability of the applications which were built using the technology was not considered. Because of this, developers with VoiceXML have difficulties of making applications with good voice interfaces. Our approach, on the contrary, starts from the desired interaction styles using voice. Then we propose a new language so the developer can easily build applications with the preferable interaction styles.

Section 2 shows our classification of dialog tasks and some requirements for the interaction styles using voice, especially for helping novice users of the system. Section 3 introduces a new voice interface for the major applications at the voice portals, such as news or weather forecasts. Section 4 reveals the language design in detail and gives some examples. Section 5 shows the current implementations and some results of our preliminary evaluations. Section 6 includes a conclusion and future works.

2. DIALOG TASKS AND DESIGNS

It is important to have the capability of writing the dialog patterns without using any programming language, because the success of Voice-Web depends on the amount of voice-enabled contents. VoiceXML has an advantage, because it has the built-in logics for filling forms. It is, however, sometimes difficult to build VoiceXML applications which are easy to use. So our work starts with the investigations on interaction styles, which depend on the nature of tasks and users. The tasks for the dialog system may be classified according to the direction of information flow (Table 1) [3].

<table>
<thead>
<tr>
<th>flow of info.</th>
<th>class</th>
<th>example domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>user → system</td>
<td>slot-filling</td>
<td>hotel reservation</td>
</tr>
<tr>
<td>user ≤ system</td>
<td>DB search</td>
<td>bibliography search</td>
</tr>
<tr>
<td>user → system</td>
<td>explanation</td>
<td>route direction</td>
</tr>
</tbody>
</table>

Table 1: Classes of task domains.

Tasks may be divided into the following classes:

(1) Tasks in the slot-filling class, such as a hotel reservation, can be machine initiated dialogs. The system asks the questions and the user makes a response to each question.

(2) Tasks in the database search class, such as a bibliography search, can be mixed-initiative dialogs. The system asks questions to gain the keywords for searching, but all the questions need not be answered.

(3) Tasks in the explanation class, such as a route direction, can include the navigation of the contents initiated by the user, and the explanation is from the speech output.

The user’s input in class (1) and (2) must be processed at the application server which dynamically creates the VoiceXML document. The user is guided by the prompt in the VoiceXML and the user input is accepted by the grammar patterns of the VoiceXML. Therefore, for these task classes, the application logic and presentation are separated.

While class (1) and (2) may be grouped together, class (3) stands apart because the user needs to be assertive and initiate control to find the information he/she wishes to know. Such information consists of many menus and choices.
In the example shown in Figure 1, there may be seven dialog states, whose active commands may alter. The user must be aware of the structure of the contents, the current choices available, and the transition of the menus and information status. It is often difficult for the users to know how to repeat the output from the system or go back to the previous state, especially in the telephony user interface. Unlike the graphical user interface, the novice user has little or no knowledge to understand the available commands and judge the current status of the menus, and may be at a loss when confronted with the structure of the system. Because of this, most of the telephony Web services tend to be difficult to master.

If we carefully design the dialogs, we can, however, make a good telephony interface system that allows for intuitive operations or useful shortcut, but we need more knowledge of the use of VoiceXML and resulting notations would unfortunately be very complicated.

We must simplify the structure of the menu and keep the entire situation consistent. In all situations, a certain voice command will be expected to work as a unique function throughout the dialog. To create such an interface using the VoiceXML, the built-in menu function would need to be changed. Presently, it is not suitable as it limits the available grammar for each menu exclusively.

3. RADIODOC INTERFACE

3.1. CD-Player Metaphor

We introduce the RadioDoc system, an IVR (Interactive Voice Response) system designed for explanation tasks. The language designed for the RadioDoc system is called RadioDoc language. Figure 2 shows the concept of RadioDoc.

A RadioDoc file consists of the sections. Each section consists of paragraphs. A paragraph provides the same function for output as the VoiceXML prompts, such as synthesized voice (Text-to-Speech) and recorded audio files. The novice users may use the system like the radio, only listening without any commands. After the last section ends, the output will be started from the first section again. While listening to the contents, the user will gradually understand the system structure and the available voice commands. To cope with the difficulty of the use, we introduce the CD-player metaphor, in which the novice users can easily understand the concept of “back” or “forward”. The users of RadioDoc easily know the behavior of moving back and forth as with the CD-player. The expert users may also use the voice shortcut for the specific information immediately.

3.2. Sound Effects and Audio Skins

Sound effects and music are important for improving the usability of telephony user interface. Audio files can be inserted to the prompts in VoiceXML, but it makes it difficult to separate the contents and the presentations. Such documents will be hard to reuse, or to adapt to the users’ preferences.

In RadioDoc, we introduce the concept of Audio Skins, which indicates the usage of sound effects and music for the roles defined in the RadioDoc. Audio Skins are separated from the RadioDoc content itself. The relation of RadioDoc and Audio Skins is similar to that of the HTML (Hyper Text Markup Language) and CSS (Cascading Style Sheets) for the Web design.

Audio Skins may be selected by the creator of the contents, to suit for the category of the contents, such as entertainment, news, route guides, stock prices, and so on. The listeners of the document may also select and apply their own favorite Audio Skins to use with any RadioDoc contents.

The following are the roles of sound effects (shown as SE) and music we may use:

- The sound for emphasizing the active voice command, such as “You may use the voice shortcuts, including (SE) art, (SE) entertainment, (SE) weather report.” This concept is similar to the indication of the linked words in HTML using the difference of color or typeface.
- The sound which shows the end of the prompt, in other words, a short break begins where the users may use the voice command.
- The sound for notifying the acceptance of the voice input.
- The music which shows the beginning of a section. The user may barge-in during the music for moving forwards or backwards.

Our preliminary evaluation of the sound effects showed the effectiveness for reducing the mistaken use of the vocabulary [4].

3.3. Search Subdialogs

The voice commands within a RadioDoc file are always active. So, it may happen that the same active grammars appear more than once within the single document. The destination of the voice shortcut may be uncertain. In such cases, the RadioDoc system provides the subdialogs to confirm by Yes-No
questions. The following dialog shows an example based on the document which contains the grammar “Reservation” in both “Hotel Reservation” and “Rent-a-Car Reservation” sections.

[User] Reservation.
[System] Do you want a hotel reservation?
[User] No.
[System] Do you want a rent-a-car reservation?
[User] Yes.
[System] (Jump to rent-a-car reservation section)

Users may cancel the subdialog and return to the RadioDoc sections where the voice command is activated.

4. RADIODOC LANGUAGE

The RadioDoc language is based on XML (eXtensible Markup Language). Elements related to the document structure, such as <section> or <p> are derived from SmartDoc developed by ASAMI Tomoharu [5]. Elements related to the voice input and output, such as <grammar> or <audio>, are borrowed from VoiceXML.

4.1. Basic Syntax and Examples

In RadioDoc, each <section> element contains paragraphs (<p>). A paragraph consists of <grammar> and <prompt>. Section is the unit for “back” or “skip” voice commands. At the beginning of each section, there is opening music. The user may say “back” for moving to the previous section. The contents of the section follow the opening music and the user may use the “back” or “repeat” command for moving to the opening of a section. This is the same behavior as the CD-player.

An <a> (anchor) element shows an attribute for inserting sound effects which indicates the voice command.

The example below has three sections:

```xml
<rdoc>
<head> <title>RadioDoc Example</title>
<link rel="audioskin" href="audioskin.xml" />
</head>
<body>
<section type="once" title="start menu" id="start">
<p>
<grammar>start menu</grammar>
<prompt> <audio src="welcome.wav"/>
Welcome to the KIT campus information. <break/>
You may use voice commands at any time, such as
<a>skip</a>,<a>repeat</a>,<a>route guide</a> or
<a>event guide</a>. </prompt>
</p>
</section>
<section title="route guide" id="route">
<p>
<grammar>route guide</grammar>
<prompt>This is the route guide. You may use
<a>by car</a> or <a>by train</a> for shortcut.</prompt>
</p>
<p id="car">
<grammar>by car</grammar>
<prompt>You can go by car on route XX.</prompt>
</p>
<p id="train">
<grammar>by train</grammar>
</p>
</section>
<section title="event guide" id="event">
<p>
<grammar>event guide</grammar>
<prompt>This is the event guide......(omitted) </prompt>
</p>
</section>
</body> </rdoc>
```

4.2. Speech Recognition Grammar

In explanation tasks, most of the voice commands will be used for search within the document. RadioDoc uses keywords, such as the name of the sections, as the command words. The shortcut for jumping to any area of the contents, via voice command is available at any time.

In the VoiceXML, speech recognition can be used at the menu with choices, the field which the user must give some values, and the link to the other dialog. Only the grammars in the application root document are globally activated.

In the RadioDoc, on the contrary, all the voice commands are treated as global. This is because the telephony interface has the difficulty of showing the status of the system, and the user occasionally confuses the available command words. To avoid such difficulties, the destination of link by a certain voice command must be consistent within the document.

To write such interface efficiently, the destination of link by a certain voice command must be consistent within the document. To write such interface efficiently, the destination of link by a certain voice command must be consistent within the document. To write such interface efficiently, the destination of link by a certain voice command must be consistent within the document. To write such interface efficiently, the destination of link by a certain voice command must be consistent within the document.

4.3. Interoperability with VoiceXML

RadioDoc is designed for the explanation tasks. To build a dialog system which includes the slot-filling or database search tasks, RadioDoc provides the link to the outer VoiceXML document server as shown:

```xml
<section title="hotel reservation">
<p>
<prompt timeout="5s">For reservation, please say <a>reserve</a> or <a>yes</a>.</prompt>
</p>
</section>
```

4.3. Interoperability with VoiceXML

RadioDoc is designed for the explanation tasks. To build a dialog system which includes the slot-filling or database search tasks, RadioDoc provides the link to the outer VoiceXML document server as shown:

```xml
<section title="hotel reservation">
<p>
<prompt timeout="5s">For reservation, please say <a>reserve</a> or <a>yes</a>.</prompt>
</p>
</section>
```

Figure 3: Concepts of grammar in RadioDoc

4.3. Interoperability with VoiceXML

RadioDoc is designed for the explanation tasks. To build a dialog system which includes the slot-filling or database search tasks, RadioDoc provides the link to the outer VoiceXML document server as shown:

```xml
<section title="hotel reservation">
<p>
<prompt timeout="5s">For reservation, please say <a>reserve</a> or <a>yes</a>.</prompt>
</p>
</section>
```
Grammars in the <link> element are active only within the <link> prompts. If the utterance doesn’t match the grammar or no input was given, the system goes to the next item of the contents. It may also be useful to call RadioDoc within the VoiceXML. In such case, RadioDoc may be invoked as the external object in VoiceXML applications as shown:

```xml
<object classid="radiodoc">
  <param name="href" expr="http://server/main.rdoc#event"/>
</object>
```

5. EVALUATIONS

We implemented a RadioDoc-to-VoiceXML converter to test the proposed interaction style and the language design. Some functions such as <link> element, audio skin selection, and search subdialogs have not been implemented yet. As the VoiceXML interpreter, we used the Nuance Voice Web Server. Our converter generates an application root grammar which includes all grammars within the original document. Each paragraph is converted to the <field> element, which has its own grammar for navigation, such as “back” or “skip”. The <noinput> element of the field contains the command to move to the next part. The beginning of the section is an additional field where the opening music is played.

We performed a preliminary experiment to compare the usability of the menu-driven IVR system and that of the RadioDoc. The application we built is the campus event information for Japanese language. The menu-driven system was built using Pronexus VBVoice and Nuance 7. Eight students tested both systems which provided the same information. Most of them were not novice users of the speech recognition. The same information retrieval task was given to all subjects. The results shown in Table 2 indicates that the RadioDoc has the same usability as the conventional system which uses the menus, and the RadioDoc can reduce the time to achieve the task. RadioDoc was reported especially useful by the users who had much trouble with the menu-driven system.

<table>
<thead>
<tr>
<th>interaction style</th>
<th>subjective test (0-10)</th>
<th>average time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>menu-driven</td>
<td>5.1</td>
<td>8.6</td>
</tr>
<tr>
<td>RadioDoc</td>
<td>5.0</td>
<td>5.5</td>
</tr>
</tbody>
</table>

Table 2: The results of evaluation

The size of the RadioDoc file we created is 160 lines. Our converter generated 19 files of VoiceXML and grammar files, with 661 lines in total. The total time of our contents is 4'10".

6. CONCLUSIONS

We proposed the RadioDoc, a voice-accessible document system. We also described the current status of our works. The result of our preliminary experiment showed the efficiency of the user interface, both for novice and expert users. The language we designed also reduced the total size of the contents, compared to the VoiceXML documents for writing the same application. The remaining work includes the evaluation with the novice users of the speech recognition, improvement of the usability for navigations and search subdialogs, and the use of more effective sound effects and music. Aural Style Sheets [6] was designed for very similar purposes to our works. Our future work is the integration of visual document and audio document. SmartDoc enables the production of HTML, JavaHelp, LaTeX, PDF (Portable Document Format) from the single document source. As our plan, shown in Figure 4, the SmartDoc normalizer and the post-processor we are creating generate the <grammar> and <prompt> elements to assist the audio-based publication. These tools may help the accessibility of the Web, especially for the visually impaired users.

![Figure 4: Overview of the RadioDoc system.](http://www.xmlsmartdoc.org/)

7. REFERENCES