Design of an Integrated Development Environment for the Voice Browsing System

Liang-Teh Lee
Department of Computer Science and Engineering
Tatung University
ltlee@cse.ttu.edu.tw

Ying-Chieh Huang
Chunghwa Telecom Co., Ltd. Data Communication Business Group
jackyhua@cht.com.tw

Abstract

To the designer of the voice service applications, an appropriate integrated development environment (IDE) for testing, debugging and simulation is a very important tool. For instance, to the interactive voice response (IVR) related services, the IDE will make the development of the voice web pages as simple as the development of the Internet web pages. By applying the IDE of the VoiceViewer, either a veteran or a novice can develop the VoiceXML easily, the development efficiency can thus be improved significantly. The core of the IDE is a mechanism of the voice browsing, which is the pivot of the entire voice flow. In this paper, we propose a framework and mechanism of the IDE for constructing a voice web browsing system. Many operating systems are collocated with PC, however, most of them lack of an appropriate voice web page browsing system. In order to solve the problem, a voice web browsing system framework for operating system is proposed. The proposed framework is portable and expandable, it can be applied to various operation system platforms easily.

Keyword: Voice Service Environment, IDE, IVR

1. Introduction

The Internet is becoming more popular and convenient. Most people can acquire abundant information from the Internet through the browser of the Desktop PC (Personal Computer). Thus, to provide an IVR (Interactive Voice Response) mechanism of the telephone for user to browse the Internet is quite important. The purpose of the VoiceXML (Voice eXtensible Markup Language) is to bring the Internet architecture to the telephone. By using the VoiceXML, user can develop web pages of voice framework as simple as coding the HTML web pages[1]. Currently, however, most computer systems lack of on-line simulation tools for Chinese IVR. Though the voice browsing system can access web pages directly, the Chinese development tools with on-line simulation options have not been included yet [2]. Thus, it is worth to develop a Chinese voice browsing system and Debug/Test Tools for improving development and browsing interfaces of the VoiceXML.

The rest of the paper is organized as follows. We will introduce the properties and comparisons of the voice web browsing systems, and then probe into theory, foundation and development of voice web browsing systems in Section 2. Based on the observed and analyzed results in Section 2, Section 3 defines the required modules of the proposed system framework, and describes the properties of each module. In this Section, we also analyze each function of the proposed voice browsing system, and build the components of the browsing system according to the previous foundation, so as to construct the voice web browsing system framework. Using PC as the implement platform, in Section 4, a PC-Based voice web browsing system framework is presented. The comparisons of the proposed system with other system are also made in this Section. Section 5 addresses the conclusions of this paper and the possible future works in this area.

2. Background

2.1 The Core of Speech Application Technique

The Voice Application Technology is an application technique that enables the use of telephone or mobile phone, PC, PDA and other intellectual machines through the ASR, speech synpaper interactive techniques, voice browsing, and intellectual message processing techniques to visit the Internet for implementing personal services and commercial services. The voice application technique which consists of speech

(ASR and TTS), voice browsing, and intellectual text message processing techniques, has become a complete application technique, and is built in accordance with existing standards of related techniques. The voice application technique is a bridge that links Public Switched Telephone Network (PSTN), with speech as the principal core, and the Internet, with data as the principal core. The telephone and mobile have become information processing terminals for users to communicate in natural language. The speech application technique allows communication among machines and remote speech servers in the form of dialog for people to request machine services in voice.

Natural voice interaction has also other advantages. Users do not have to fill up a web form or use a keyboard to enter their names or e-mail addresses. Regarding keyword capture and natural voice processing technology, users can easily use their customized interface and say a word such as "台積電" and the computer can respond "買進 43 塊 5 毛…".

2.2 The Network and Data

Voice browsing forms a bridge between the network and voice communications. The Internet explosion is basically built on the success of web browsing. The effective protocol communications between Client/Server architecture, HTML markup language, and http have given the Internet a strong distributed/concentrated query structure as well as a simplified application development mechanism. It can be said that browsing is the core of the Internet. Voice applications have been previously simplified with closed interaction. Data originally come from pre-recorded messages while operating procedures use simple keypad menus. The ability to distinguish and integrate voice that allows human-machine interaction is becoming more mature.

2.3 Voice Browsing, Debugging, and Testing Tools

Tools such as VOICEXML BUILDER [3] and V-Builder 1.2 [4] are not enough for browsing, debugging, and testing. Moreover, they do not support Chinese language. Currently, development problems voiced out in VoiceXML Forum are shown in Figure 2.1 [5]. The figure shows that in terms of software tools, those that help developers to improve coding efficiency and definition are enumerated as follows:

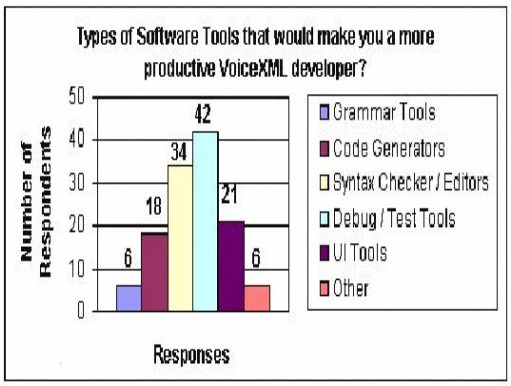

Figure 2.1: The VoiceXML-Related Tools Survey

Debug/Test Tools and Syntax Checkers/Editors are the most useful. Since current local industries do not have these two VoiceXML Chinese language development tools, the design in this paper is starting from these two so as to improve the problems involved in development and testing.

3. VoiceViewer: IDE for the Voice Browsing System

3.1 Integrated VoiceXML Voice Service Environment

The increasing of the people's outdoors activities and the advances of the technology have spurred development of various types of mobile devices. However, few are talking about voice browsing portals. Thus, there is a great need for designing an integrated VoiceXML voice operating environment. Voice services need an ideal operating environment so that all its functions can be used. We have designed an integrated VoiceXML voice service environment as shown in Figure 3.1. The environment includes a voice browsing system which consists of six main parts, Voice Server, TTS Server, ASR Server, Web Server, VXML Interpreter, and VXML Generator Module. The idea behind this operating environment is that, aside from providing plain old telephone service (POTS), it also offers IP-phone, smart phone, and tablet PC services.

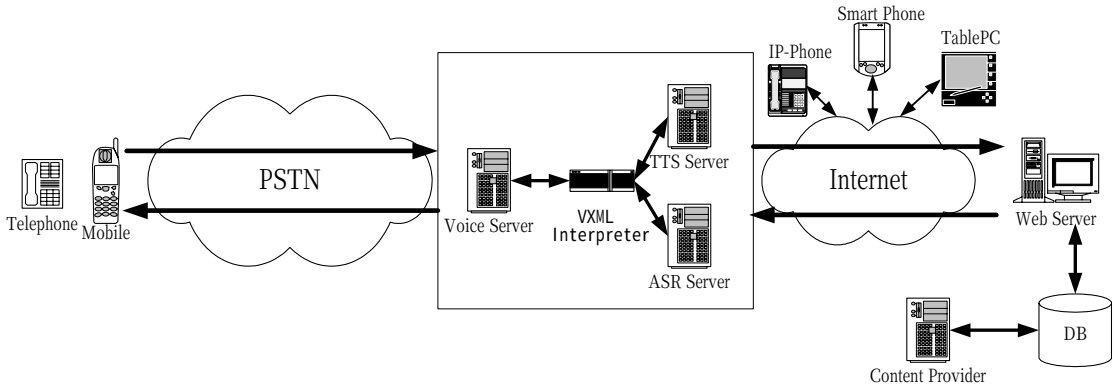

Figure 3.1: Integrated VoiceXML voice service

To the designer of the voice service applications, a development system with the above functions is required. Therefore, we implement the system by simulating the above functions using the functional diagram as shown in Figure 3.2. The system is an integrated development environment for the voice browsing system, called VoiceViewer. In the next section, the VoiceViewer Components and the VXI core will be discussed in detail.

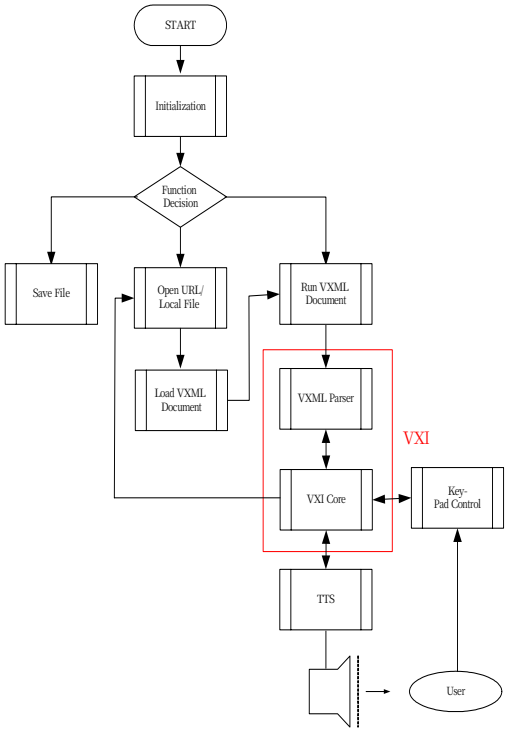

Figure 3.2: Functional diagram of the VoiceViewer

3.2 Design of the VoiceViewer

3.2.1 The Components

In this subsection seven components, Initialization, Function Decision, Save File, Open URL/Local File, Run VXML Document, Load VXML Document, and Key-Pad Control, are described first. Other components are VXML parser, VXI Core, and TTS.

3.2.2 VXML Interpreter

Similar to the function of most interpreters, instead of generating object code, the VXML interpreter will interpret and execute each original instruction immediately. There are usually two types of interpretation: iterative interpretation and recursive interpretation [6].

In our design, basic instructions may involve different formats, instructions may have commands, expressions, or declarations. Interpreting a command may prompt another action. In this case, recursive interpretation has to be used for the program. Below, we will give a simple example to illustrate the process of description, analysis, using a parse tree to build a basic computational procedure, and with this tree, assigning tasks to complete the work. At first, the VXI includs two parts, VXML parser and VXI core. The VXML parser will be discussed below:

1. VXML Parser:
As the “Run VXML Document” component receives the VXML document from the remote web server or disk, it gets a text content with text/vxml content type that is one of MIME (Multipurpose Internet Mail Extensions) [7] content types. It needs a VXML parser to parse the document. The VXML parser will get a document from the component and parse it. After finishing parsing the document, the VXML parser will construct a tree, called a parse tree, which represents the content of the document. Every node in the tree represents a tag or a text, and attributes of a node represent attributes of a tag. A child node is a node that maps the tag or text occurs under the tag that is represented as the parent node. VoiceXML documents are mainly made up of many

elements and grammatical rules; form especially plays an important role. This section will categorize VoiceXML elements.

During performing the lexical analysis, tokens must be defined. Based on the definition of different types, the syntactical analysis and debugging can be executed. Furthermore, based on the syntax format, composite command of the source program can be parsed and decided.

There are seven types of tokens as shown in Table 3.1.

Table 3.1 Token types

Token Type	Content
IDENTIFIER	Variable, element or attribute name
KEYWORD	Keyword
STRING	In string of " "
DELIMITER	Punctuation and operator
NUMBER	Value
BLOCK	< or >
Reserved	Reserved for portability and extensibility.

Before coding the VXML parser, one must have a basic knowledge of VXML structure. This paper uses a modularized VXML parser based on the formal VXML structure (W3C standard) [8]. The structure has strict declaration and definition parts and uncommon statements are temporarily not considered. The study does not need the complete set, only the basic elements. The requirement for a basic browser and development system environment means that the interpreter structure and environment is also much simplified, using simplified functions. All VXML programs normally have more than one element and variable. An element has an element name, attribute list, and program code block. The program code block starts with a " < " followed by one or more statements which are terminated by a " > ". In VXML, statements are preceded by a Keyword (such as vxml or form) or it can be a simple expression (such as A = B + C). We use a simple production rule as defined in Table 3.2. The purpose of "Reserved" is reserved for portability and expansibility in the future.

A VXML program starts from a "<?xml" call mark. If there is no other call mark, open file, or goto call, then the document is terminated by a "</vxml>" mark. Based on the above production rule, the entire program is analyzed once. The "<?xml" mark starts the interpreter while the "<?xml"…"</vxml> " ends it.

Table 3.2 Production rules

Category	Production Rule
Program	Assembly of elements (Include variables)
Element	Element assigned word, attribute series, program blocks
Program block	"<" Statement ">"
Statement	Keywords, expressions, program blocks
Reserved	Reserved for portability and extensibility

The grammar rules describe each language's basic syntax and through syntactical rules, statements and expressions can be accurately made. Thus, for an interpreter, syntax decisions have to be based on parser rules. In this paper, the parser can recognize the source program. When a parse expression is executed, it will first run the scanner program. The scanner program will declare or search for the identifier.

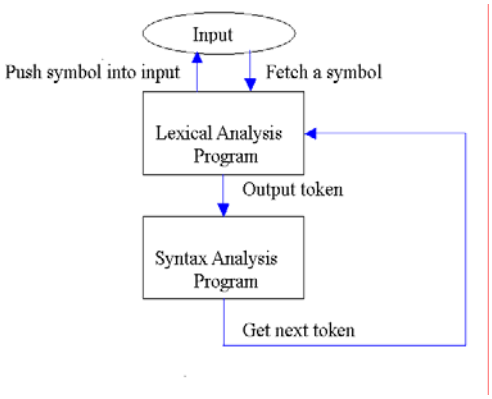

Figure 3.3: Token parsing

Figure 3.3 shows the interpreter which performs the lexical analysis and syntax analysis on the source program. Nevertheless, the process may not be a sequence of fetch, analyze, and execute steps. In this case, lexical and syntactical analyses are done recursively until the source program is cut into meaningful syntactical tokens. In addition, the parsing rule is verified to be error-free.

The lexical and syntactical analyses compose a producer/consumer relationship. Lexical analysis produces a token while syntactical analysis uses the token. A produced token can be stored in a token buffer until they are used up. When the buffer is full or empty, further processing cannot proceed. Normally, the buffer only has one token and the lexical

analysis program is replaced by a syntactical analysis program that calls the needed token.

The current study undertakes three levels of analysis on source code syntax:

- Linear Analysis: Source code is read from left to right as a string of symbols. It is then organized into tokens which are symbols that have special meanings.
- Hierarchical Analysis: Each symbol or syntax token is organized together in a hierarchy, forming a hierarchical assembly with its own special meaning. This is also called syntax analysis.
- Semantic Analysis: Checks are made to ensure that each part of the program can be meaningfully combined.

2. VXI core:

To design the interpreter, the current study uses a recursive interpretation and the basic structure is illustrated as follows:

Begin (CC is a parse tree, shown in Figure 3.11)
fetch and analyze CC ;
execute CC ;
end

By parsing the lexical analysis and grammatical rules, we can complete the core program using the interpreter as in the Figure 3.4.

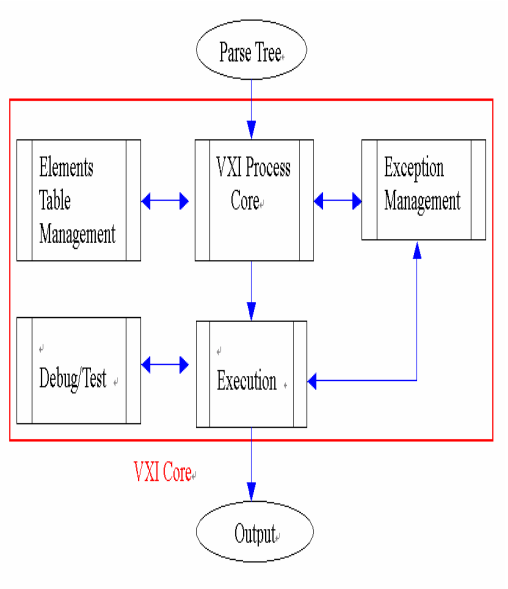

Figure 3.4 Basic interpreter framework

The VXI core program shown in Fig. 3.4 is the interpreter module proposed by this paper. The program was written using Visual Basic and Visual C++ objects, object encapsulation module, and enabling this part of the program to have a basic encapsulation. In our design, we avoid integrating interpreter functions that may produce undesired results.

Based on the loop structure in the VXML language, the loop structure in our interpreter is divided into a syntactical structure:
Define if … else as:

```
<if ( expression )>
 statement
<else/>
 statement2
</if>
```

From the above definition, by using a basic loop function, we can perform the loop through the program control. The VXML Interpreter is the core the proposed IDE, and the if-else structure is the basic syntax for handling recursive processing. According to the parsing rule, we can build a parse tree as shown in Figure 3.5

After constructing the parse tree as shown in Figure 3.5, the interpreter will travel the tree to generate the required result.

3.3 TTS design

The flow of TTS is shown in the Fig. 3.6. A text is derived after the processing of the interpreter. Then the BIG5 code of text is converted, and an audio file is obtained based on the index tree. The audio file is sent to a voice synthesizer, to generate the natural voice output. The TTS is expandable for other languages, such as the simplified Chinese character, Japanese character, etc.

3.4 Input and Output

3.4.1 Console Mode

In the simple console, we used MS-DOS mode for standard input/output while standard error is written in standard output. The interpreter module reads the source program through the parameter. It allocates a memory block for the source code, and then handed on to the interpreter for processing. Results of error messages are written in the output environment. The standard output is the console dialogue screen. The interpreter module is coded by Visual C++.

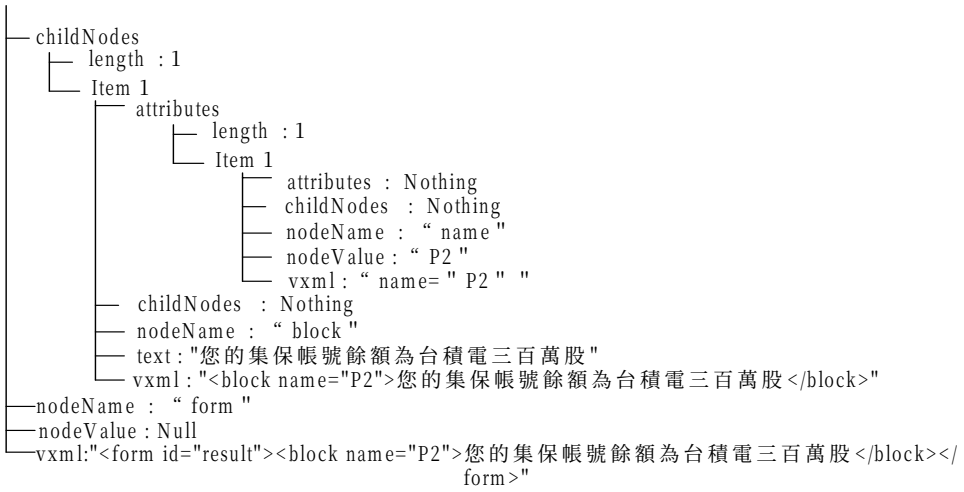

Figure 3.5 Constructed parse tree framework

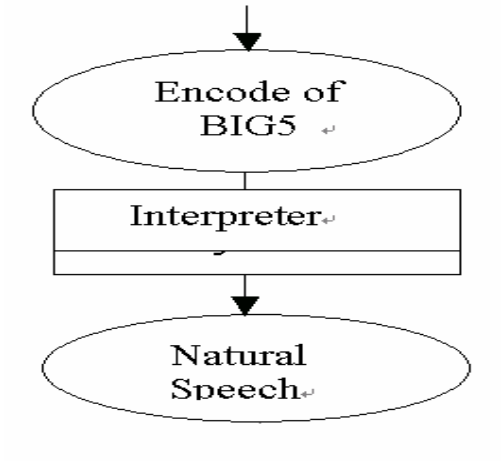

Figure 3.6 TTS Flow

3.4.2 Window System

The browser system and editor use the basic frame of Visual Basic. This helps us to use the main frame and table for the window interface. There is not necessary to set the windows appearance. We already have a basic editor area with complete editing functions. Moreover, we can construct a basic browsing and editing environment (Figure 3.7 shows Browser and editor).

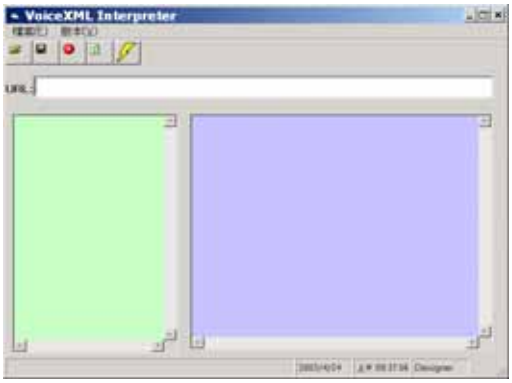

Figure 3.7 Browser and editor environment

4. Operation and Discussion

4.1 VoiceViewer Platform

There are modes in the system, console mode and windows mode.

1. Console mode: Execution of the VXI Core The input method is set to use the parameter method by opening a file (See Figure 4.1).

Figure 4.1: Parameter addition method to input the file

We used the interpreter, called testVXI, to interpret query_gipo.vxml in a console mode. We also used the command line to input query_gipo.vxml contents. The shell shows the prompt when the program is executed.

Figure 4.2 shows the starting information for interpreting the program query_gipo.vxml after issuing the command:
C:\VXICore>Exe\testVXI.exe query_gipo.vxml -numChannels 1.

The output of the interpreted result for interpreting the program query_gipo.vxml is shown in Figure 4.3.

In Figure 4.3 the simulation process are shown and detailed procedures can be viewed. This allows developers to see everything at one glance and quickly evaluate if the voice browser procedure is proceeding as planned.

2. Windows mode: Interpreter Execution

The current section uses images to explain interpreter functions. Figure 4.4 shows the voice browser system executed in Windows. The testing platform operating system is Windows 2000.

Figure 4.5 shows a query for numerical input from the user when the voice browser system is running.

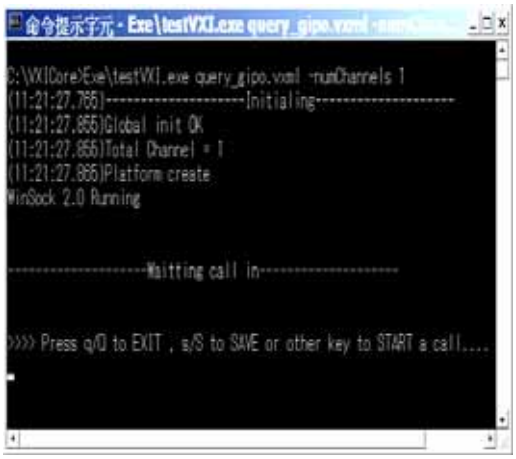

Figure 4.2 Pre query_gipo.vxml wait screen

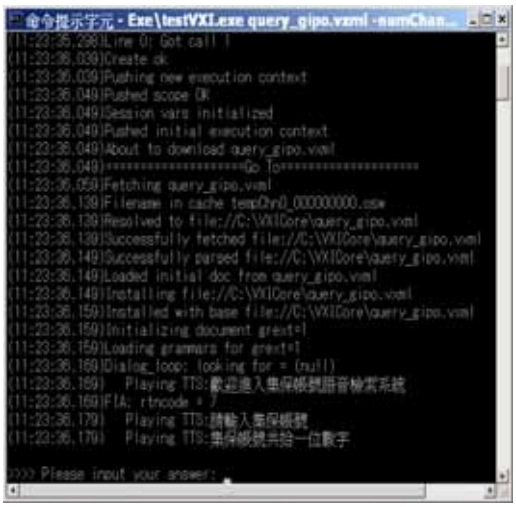

Figure 4.3 query_gipo.vxml contents and output

4.2 Evaluation and Discussion

Table 4.1 shows the comparisons of four IDEs, OpenVXI, VoiceGenie IDE, XML Spy, and the proposed VoiceViewer. SpeechWorks [9] and VoiceGenie [10] are two of the market leaders in voice browsing system development, and have begun efforts to simplify development of voice browsing systems for their customers.

Figure 4.4 Executing the voice browser system in Windows

Figure 4.5 Phone keypad simulation interface

Table .4.1: Comparison of IDE

Product / Function	OpenVXI	VoiceGenie Genie IDE	XML Spy	VoiceViewer
Chinese TTS package	X	X	X	○
Voice Interaction	○	○	X	○
Console mode	○	X	X	○
Windows mode	X	○	○	○
Online	○	○	X	○
Portability	X	X	X	○
Extensibility	○	X	X	○

5. Conclusion

The VoiceViewer voice browser system described in this paper can be run in either MS-DOS or windows mode. We developed a VoiceXML interpreter, including the editing, debugging, simulation, and testing, that facilitates developers in debugging and testing of voice web. In addition, the Chinese TTS (Text To Speech) mechanism has been modified

in our system to solve the mixed pronunciation problem of the Chinese character. As a result, through the interface of the browser, developers or users can develop or browse the VoiceXML web pages easily. Furthermore, for browsing the VoiceXML web pages, the PC speaker can be used to browse and present the flow of VoiceXML document, and the flow of IVR can be processed by way of the simulated interface of the telephone keypad input. Thus, the developers and general users can develop or browse voice web pages simply and quickly on the PC through a centralized VoiceXML Test Portal.

6. References

[1] D. Raggett, A.L. Hors, and I. Jacobs, "HTML 4.01 Specification", W3C Recommendation. WWW Consortium, Dec. 1999.

[2] Ying Chi Huang, "Research on Web Accessing with Aural Rendering Model", http://datas.ncl.edu.tw/theabs/1/, Xerox PARC, 4-6 Nov. 1996, pp. 57-103

[3] Janet D. Hartman and Joaquin A. Vila," VOICEXML BUILDER: A WORKBENCH FOR INVESTIGATING VOICED-BASED APPLICATIONS", ASEE/IEEE Frontiers in Education Conference, October 10 - 13, 2001 Reno, NV6

[4] Nuance communication, V-Builder 1.2, Nuance, 2001

[5] http://www.voicexmlreview.org

[6] David A. Watt, "Programming Language Processors", Prentice Hall, 1993

[7] N. Freed, et al., Multipurpose Internet Mail Extensions (MIME), November 1996. http://www.ietf.org/rfc/rfc2045.txt.

[8] http://www.w3.org/TR/2003/CR-voicexml20-20030128/#dmlAFIA

[9] SpeechWorks International, Inc. http://www.speechworks.com/

[10] VoiceGenie Technologies Inc, http://www.voicegenie.com/