International Journal of Computer Applications (0975 – 8887)
Volume 74– No.14, July 2013

XML based Interactive Voice Response System

Sharad Kumar Singh
PT PureTesting Software P Ltd.
Noida, India

ABSTRACT

The paper presents the architecture of a web based interactive voice response system using Voice XML. The paper includes a discussion on the architecture of the IVR system, its components, and a detailed description of the functionality of VXML Interpreter and its use in IVR systems. It also describes the integration of VXML Interpreter, CCXML Interpreter and the related media & telephony resources. Finally it presents performance measurement techniques and technical proposal for increasing the performance of such a system.

General Terms

Interactive Voice Response System, Telephony, Voice XML, Call Control XML

Keywords

IVR, Voice XML, Web based IVR, VXML Interpreter, CCXML, FIA.

1. INTRODUCTION

Interactive voice response (IVR) is a phone technology that allows a computer to interact with humans through the use of voice and DTMF tones input via phone [1]. It has been an essential element in the customer support equation for more than a decade. The driving idea behind the development of VXML was to reduce cost because it would not require expensive call center agents but would be fully automated systems capable of quick modifications.

The advancement in technology from touch-tone to speech-enable and then integrating both has introduced new challenges including:

•Integrating disparate customer access points and back-end data sources.

•Scaling to ever-larger systems.

•Catering to larger number of customers.

•Quick development time for new customers and maintenance time for the existing ones.

It was difficult to provide solution to the above problems using the traditional interactive voice response systems. Hence, the modern tone-speech enabled solutions were developed to alleviate these problems. Voice XML is used to design and develop the backbone of these solutions which are known as voice portals.

The time required for the development of a voice xml based solution is almost three times lesser than what is required for traditional IVRs [2]. For the very same reasons, VoiceXML has been widely adopted and accepted as the platform for developing IVRs in the speech industry.

In order to understand the how a VoiceXML IVR application operates it is useful to compare it with a traditional web application. In a traditional web application, web browser presents an HTML based web page from a web server, populates data from a database server and performs specified actions based on the user inputs through the web browser. Similarly, in the case of a VoiceXML IVRs, the VXML engine is responsible for performing the tasks that were performed by the web browser and the web server.

2. A VOICE XML DOCUMENT

2.1 Sample VXML document

Following is a sample VXML document:

```
<?xml version="1.0"?>
<vxml application="sample.vxml" version="2.0">
  <form id="form1">
    <block>
      <prompt> Welcome World </prompt>
    </block>
  </form>
</vxml>
```

Each tag in the VXML document has its meaning and clearly described functionality as per the W3C standards. However to keep it short, this document plays “Welcome World” as soon as the user dials in the voice xml gateway.

2.2 HTML versus VXML

HTML pages and VXML pages are equally alike and distinct. Following is a code sample from a simple HTML page and a VXML page.

Table 1. HTML versus VXML document

HTML Page	VXML Page

`<html>`
`  <body>`
`    <img src="Coffee.jpg"/>`
`  </body>`
` </html>`
	`<vxml version="2.0">`
	`  <form>`
	`    <block>`
	`    <prompt>Coffee </prompt>`
	`    </block>`
	` </form>`
	`</vxml>`

In the above example of HTML, a page is set up where visitors can view a picture of a cup of coffee. However, in the VXML example, a document has been set up where callers can hear a prompt stating "coffee. The theories behind the VoiceXML example could be easily understood by digging a little deeper.

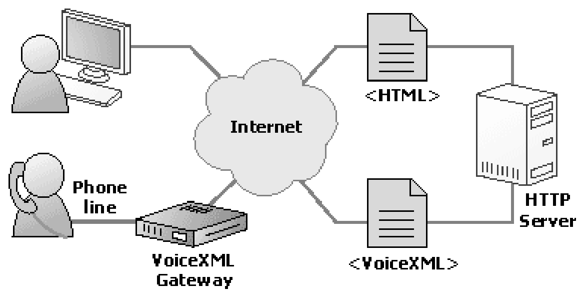

Fig 1: HTML v/s VXML

Using the HTML versus VXML analogy, it is important to compare how HTML pages are served and how VXML pages are executed when a caller calls. When designing an HTML page, a document like above is first created, and then uploaded it to some webserver so that it can be fetched whenever requested. Similarly in VoiceXML the content must be located on a webserver so that the servers can fetch it whenever required.

VoiceXML works with the same general principles but here the telephone is used as the browser.

The next point of distinction is in how documents are fetched and executed. When a user clicks a link to the 'coffee' HTML page, a request is sent to the webserver hosting the document, and the HTML page will then load the page in the user's web browser with the accompanying picture of a cup of coffee. Again, VoiceXML works on the same principles, with the only difference that the telephone is used as the web browser. Instead of clicking on a link, a user will dial the number pointing to the VMXL gateway, which is the equivalent of a hyperlink in HTML. When this number is dialed, it tells the server/interpreter to fetch the document that has been associated with that particular phone number. Assuming that the code is well formed XML, and that the mapping has no typographical errors, the coffee.vxml file will be loaded and executed, thus outputting the text to speech message to the caller.

3. ARCHITECTURE OF VXML BASED IVR SYSTEM

A typical VoiceXML based system contains the four main components.

· Telephone Network: It could be a PSTN network or VoIP packet network.

· VoiceXML Gateway: It is the core of the voice xml based IVR systems. It comprises of an engine for interpreting VXML documents, speech synthesis, grammar recognition, audio playbacks and telephony resources.

· Application Server: It is typically a Web Server that hosts the VoiceXML documents.

· TCP/IP Network: LAN, WAN or public Internet.

VoiceXML connects to Telephone Network on one side and TCP/IP network and Application Server on the other side.

4. VOICE XML GATEWAY

The components included in the voice xml gateway revolve around the engine which is known as the VXML interpreter. This engine is responsible for Mediating, Controlling and Distributing data and logic flow in and around the other components of the gateway. VoiceXML is not only used to conceive and develop vocal but also multimodal solutions [3]. In such a case, VXML interpreter acts as a modality component in the multi modal architecture [4].

4.1 VXML Interpreter

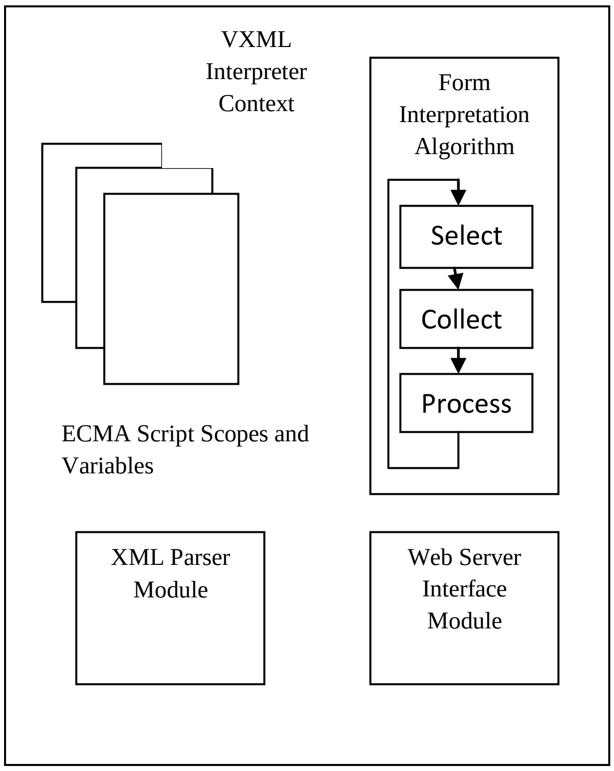
VXML Interpreter Context
Form Interpretation Algorithm
Select
Collect
Process
ECMA Script Scopes and Variables
XML Parser Module
Web Server Interface Module

Fig 2: VXML Interpreter

4.1.1 Design options

The VXML interpreter can be designed in any of the programming languages, the most common being C++ and JAVA [5]. However, there isn't much difference in the design perspective while using C++ or JAVA, both being an Object Oriented Language, the difference lays in the third party libraries and devices which would be required to communicate with the VXML Interpreter. Here, in this article the design and discussions are based on the C++ paradigm.

4.1.2 Components

VXML Interpreter engine comprises of the following major components:

4.1.2.1 Web Server Interface Module

The Web Server Interface Module uses various protocols, provided within the implementation to fetch documents and applications from a document server.

The common protocols which are used in order to request for and download documents from the document server via the HTTP client are through HTTP get, HTTP post, ftp and HTTPS. The protocol to be used and the document to be downloaded are specified during session initiation [6]. However, these could be specified again while traversing from one document to another.

A DNS lookup service is sometimes also required to resolve the named address. Multi-threading safe third party libraries such as libcurl etc. are used to achieve high performance document fetching in the engine.

4.1.2.2 Form Interpretation Module

The form Interpretation Module is responsible for the implementing the functional logic and traversing through the VXML document based on caller inputs [7]. The form interpretation algorithm (FIA) drives the interaction between the caller and a VoiceXML input items. Execution of the code in forms is handled by the form interpretation algorithm (FIA), which loops through the items in the form and processes (or reprocesses) them.

According to the W3C recommendation, FIA must handle the following [7]

- Form initialization.
- Prompting or delivering audio to the caller.
- Grammar activation and deactivation
- Entering the form with an utterance that matched one of the form's document-scoped grammars while the user was visiting a different form or menu.
- Leaving the form because the user matched another form, menu, or link's document-scoped grammar.
- Processing multiple field fills from one utterance, including the execution of the relevant <filled> actions.
- Selecting the next form item to visit, and then processing that form item.
- Choosing the correct catch element to handle any events thrown while processing a form item.

The interpretation algorithm of FIA is broken into three different phases viz. select, collect and process phase [8]. In select phase the VXML interpreter selects the form item (input item) which needs to be processed. Then in the collect phase it collects the caller input against the selected item and validates with the active grammars which are basically validation rules for the input item [9]. Finally in the process phase, it processes the user input as per the execution logic defined in the VXML document.

4.1.2.3 XML Parser Module

The XML Parser module acts as the interface between the engine and the third party XML parsing and validation library. It is also responsible for handling the tags pertaining to the parsed DOM trees. It also contains functionality which allows the user to validate dynamically generated XML scripts against DTD and Schema. Its major functionality is to convert the parsed VXML file into a binary format to facilitate quick cycles for the Form Interpretation Module.

4.2 Media Resource

The media resource is responsible for speech recognition and grammar validations for the input received from the caller as an audio. It matches the input against the active grammars and converts the audio response to binary format which could be understood by the VXML Interpreter. It is also responsible for converting text to speech and audio playback. Whenever FIA encounters prompt element or block element with a text, it needs to be played as an audio to the caller [10]. The VXML interpreter sends this audio text or a .wav file location to the media resource which in turn plays it to the caller.

4.3 Telephony Resource

The telephony resources include DTMF inputs and Call control. DTMF inputs serve as caller responses to the Voice gateway. The call control features such as call disconnect, call transfer, conference etc. which are sent from the telephone are handled by the telephony resource. In order to achieve these functionalities Call Control Interpreter and VXML Interpreter should work in conjunction as described in the fig. 3.

5. CCXML GATEWAY

The CCXML (Call Control XML) gateway is required to handle call control requirements that are beyond the scope of the VoiceXML specification. Although the CCXML and VXML can be used in conjunction, both are mutually independent. VoiceXML does support certain call control features like transfer etc. but these would not be enough for providing complex solutions for call controls, which would be required in IVR systems. Hence, there are two approaches in designing an IVR system using the VXML. The first approach combines SIP as a control language and VoiceXML as the interaction language [11]. The second approach is to use CCXML as a control language. Former being more used with traditional IVR systems built with CCXML and the latter is used in more recent and advanced IVR systems. CCXML offers more controlled and advanced call control features with the advantage of short development time required to implement and modify the system. The following requirements are addressed by this w3c specification [12]:

- Support for multi-party conferencing, with advanced conference and audio control. A conferencing application involves multiple participants, and is dependent upon call control to establish relationships between those participants.
- The ability to give each active call leg its own dedicated VoiceXML interpreter. For example, in VoiceXML, the second leg of a transferred call lacks a VoiceXML interpreter of its own, limiting the scope of possible applications.
- Sophisticated multiple-call handling and control, including the ability to place outgoing calls.
- Handling for a richer class of asynchronous events. Advanced telephony operations involve substantial amounts of signals, status events, and message-passing. VoiceXML 2.0 does not integrate asynchronous "external" events into its event-processing model.
- VoiceXML lacks the external interfaces required to interact with an outside call queue, or place calls on behalf of an external document server.

The Voice XML when used with CCXML provides the benefits such as lower code complexity and higher extensibility [11]. In such a scenario, the engine and gateway once developed does not require repeated and frequent modifications. The only customization that is required is in creating or modifying the VXML/CCXML documents which are easier to code when compared to traditional code developed in a high level language. Hench the code complexity is reduced. Also XML based languages are far more extensible when compared with SIP based architecture.

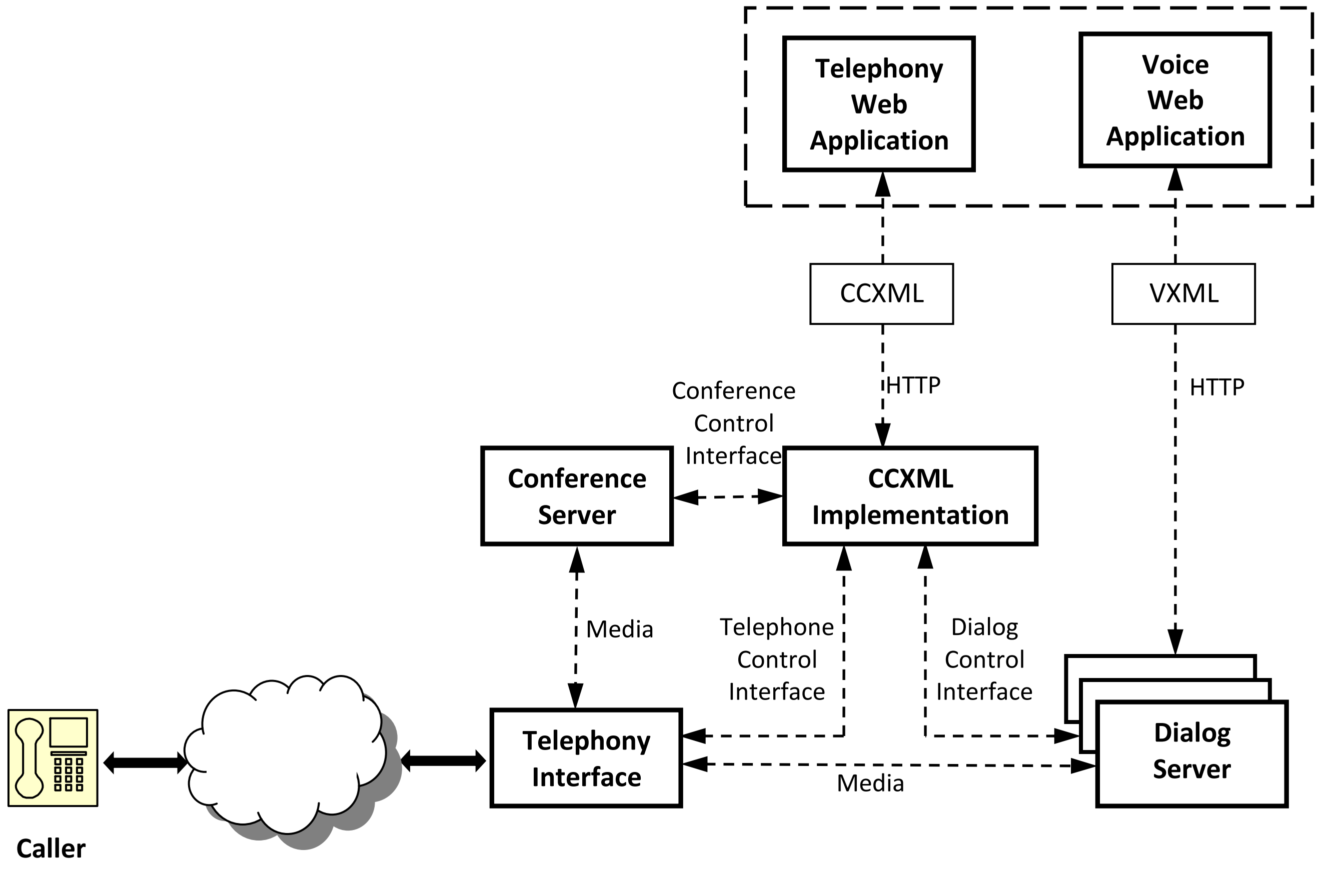

Fig 3: CCXML – VXML Integration Architecture

6. DEVELOPMENT TOOLS

There are numerous resources which can be used to quickly create or modify an error free VXML document. Following are few of them:

6.1 Plain text IDE

These are integrated development environments for creating and modifying a VXML/CCXML document. The IDEs usually should have any or all of the following features:

- Color Coding – This enables the developer to quickly identify the reserve keywords, matching tags, custom strings for prompt and audio playbacks etc.
- Inline syntax checking – The IDEs usually act like a mini interpreter; in which code can be continuously parsed while it is being edited, providing instant feedback when syntax errors are introduced.
- Document traversal – The VXML documents may contain certain transition tags which require a new document to be loaded. These IDEs are also able to immediately load the next document as soon as the user clicks on the hyperlink or the URL.

6.2 GUI based IDE

Using the GUI based integrated development environments developer can use the drag and drop feature to create or modify any VXML document. These IDEs serve the same purpose as Plain Text IDEs but differ in the following:

- Drag and Drop – If a developer wants to create or modify any tag in the VXML document, it is not required to type the tag and its parameters. Instead, the developer drags the required tag from the available tag tools and it is automatically populated at the desired location.
- Proactive rather than reactive – The IDE does not allow developer to insert any incorrect tags or correct tags at incorrect locations. Hence, it acts proactively rather than throwing an error after the developer finishes typing.
- Large binary size –GUI based IDEs are complex to design and have a very large binary or source code size.
- Low availability – Since these types of IDEs require a lot of monetary and development effort, these are not easily available commercially as well as in the open source domain.

7. PERFORMANCE MEASUREMENT AND OPTIMIZATION

Performance is one of the key features of any VXML or CCXML interpreter. It is measured as how the integrated system performs in terms of responsiveness and stability under a particular workload. The approach suggested here treats the complete gateway as a black box. Artificial load is generated from a different program which is known as a workload generator. The workload generator calls the APIs exposed by the telephony interface. As soon as a call is received on the telephony interface through the workload generator, it follows the normal execution path to complete the call which would typically involve session initiation, fetching documents, parsing, traversal and dialog interpretations etc. When it completes the call it returns to the workload generator with the status of the call. This status would comprise of time taken, success/failure etc. A number of calls are generated from the workload generated in order to perform a load test on the system. The through put of the system varies depending on many reasons but the following two are the critical ones to look at:

$$Throuhput \propto \frac{1}{\text{number of calls to the interface}}$$

$$Throuhput \propto \frac{1}{\text{size of the VXML or CCXML document}}$$

The first one is more of a business requirement that drives the load testing and cannot be regulated or improved through technical improvements in the system. The second one is more relevant for making improvements to the system. It is observed that the performance degradation for huge documents is due to two major factors:

- Time taken in downloading the document.
- Parsing the document.

Caching could be used in order to reduce the time taken to download the document. The document should be cached and checked before downloading, for any modifications that have been done on the previously downloaded document. If the document has not been modified from the last download it can be fetched from the cache. Similarly, a caching mechanism can be developed for the parsed binary tree which is created after the parsing of the document as described in section 4.1.2.3 of this article. These two techniques could be used in conjunction with each other in order to make major improvements in the performance of the VXML Engine.

8. PLATFORM CERTIFICATION

In order to establish a VXML based IVR system, the platform has to go through a series of validation tests which are provided by the VXML forum [13]. These tests are based on W3C VXML specifications and standards. These are basically an exhaustive set of tests which act as a benchmark for the platform based on the W3C standards. The VoiceXML Forum's Platform Certification Program usually provides vendor-independent, industry-standard certification that supports all parts of the VoiceXML ecosystem.

9. ACKNOWLEDGMENTS

The author would like to express his sincere thanks to all the people in the VXML Interpreter development team for their help and support without which it would not have been possible to get the deep insights in the Voice XML.

10. REFERENCES

[1] From Wikipedia, description about interactive voice response system
http://en.wikipedia.org/wiki/Interactive_voice_response
[2] VoiceXML Forum is a global industry organization that works to accelerate the adoption of VoiceXML and adjacent technologies. The reference is taken from the frequently asked questions of the forum.
http://www.voicexml.org/about/frequently-asked-questions
[3] Anderson, E. A., Breitenbach, S., Burd, T., Chidambaram, N., Houle, P., D. Newsome, D, Tang, X., Zhu, X., Early Adopter VoiceXML, Wrox, 2001 p 24.
[4] W3C Recommendation for VoiceXML 3.0.
http://www.w3.org/TR/voicexml30/
[5] Adam Hocek, David Cuddihy, Prentice Hall Professional Published: January 2003, Definitive VoiceXML
[6] The Staff of DreamTech Inc, McGraw-Hill Companies 2002, VoiceXML 2.0 Developer's Guide
[7] W3C Recommendation for VoiceXML 2.0.
http://www.w3.org/TR/voicexml20/
[8] A. Larson, Prentice Hall Professional Technical Reference 2002, VoiceXML: Introduction to Developing Speech Applications
[9] W3C Recommendation for SRGS grammar.
http://www.w3.org/TR/speech-grammar/
[10] Bob C. Edgar, C M P Books, 2001, the VoiceXML Handbook.
[11] Daniel Amyot and Renato Simoes 2007 "Combining Voice XML with CCXML: A Comparative Study", Consumer Communications and Networking Conference, 2007.
[12] W3C Recommendation for CCXML 1.0.
http://www.w3.org/TR/ccxml/
[13] VoiceXML Forum is a global industry organization that works to accelerate the adoption of VoiceXML and adjacent technologies. The reference is taken from the platform certification section of the forum.
http://www.voicexml.org/platform-certification