7. Voice Browser

WWW was introduced basically for sharing knowledge of all kinds. Now it has also become a communicating medium where as, telephone is still all time most popular medium of voice communication since very starting of 19th century. Now a great idea is to deploy web on telephone because accessing web on computer required computer literacy and understanding of English (90% of web-sites are in English) but only a few percentage of Indians are computer as well as English literate. So web is not very useful for those people. Voice browsing can do a good job in this particular case because it is in natural languages with no computer literacy. And another advantage is that it is very much user interactive so user can keep eyes and hand free for other things.It provide an affective medium for blind user of web.

A voice browser is a combination of hardware and software that takes voice and key pressed as input, interpret voice (voice markup languages) and generate voice as output. W3C s Speech Interface Framework provides four phases to complete, the task of voice browser. These are speech synthesis, speech recognition, telephony call control, voice dialogs (Voice XML).

Speech Recognition Grammars: Speech recognition grammar analyzes audio signals (voice) and recognizes the words uttered by a speaker or user. The Speech Recognition Grammar Specification (SRGS) also covers DTMF (touch tone key presses) input which provides an alternative Input method in noisy conditions or somehow uncomfortable condition to speak. Recognizers give almost confidence values. SRGS are specified in either an XML or an equivalent augmented BNF (ABNF) syntax. If there are several possible parses, the recognizer may report the most likely alternatives.

Speech Synthesis: It is just the reverse process of speech recognition. In Speech Recognition, user request is converted into web understandable format of request whereas Speech synthesis is the artificial production of human speech from text data. A text-to- speech system (engine) do perform this task in two parts- a front end and a back end. The front end takes input in the form of text and outputs a symbolic linguistic representation (symbols to represent linguistic information such as phonetics, phonology, morphology, syntax, or semantics). This further inputted to the back end. It results into synthesized speech waveform (actual sound output).

Call Control (CCXML): Call Control XML (CCXML) is a mark up language used to provide facilities like call transfer, call waiting, call ignorance, and call answering while user is in the process of accessing the web thru voice browser. Conferencing is also possible by CCXML. It is not going to build any network-based call processing applications in a telephone switching system or controlling an entire telecom network, CCXMLs scope is controlling resources in a platform on the network edge only.

VoiceXML (Voice dialogs): VoiceXML is a W3C recommendation for devloping advanced telephony applications.

VoiceXML allows the average web developer to write telephony applications with the ease and simplicity of writing the average HTML web page. As VXML is a tag-based markup language, its structure is very similar to HTML in many ways, but instead of being a primarily visual medium, VoiceXML is an auditory medium that allows the end user to navigate through his telephony page by using voice commands, rather than by clicking a button on a web page.

VoiceXML is designed for creating audio dialogs that feature synthesized speech, digitized audio, recognition of spoken and DTMF key input, recording of spoken input, telephony, and mixed initiative conversations. Its major goal is to bring the advantages of Web-based development and content delivery to interactive voice response applications.

HTML versus VXML:

Let's look at the following code samples from a simple HTML page and a VXML page.

```
<html>
 <body>
 <imgsrc= AngryGrandpa.jpg />
 </body>
</html>
<vxmlversion= 2.0">
 <form>
 <block>
 <prompt>Grandpa is angry! </prompt>
 </block>
</form>
</vxml>
```

In HTML page, there is a picture of an angry Grandpa. He does look rather upset, into it. But its VXML version, we will set up a document where callers can hear a prompt stating that Grandpa is yelling his head off. That is a example how we will try to change HTML into VXML.

How it Works

HTML page first written and then upload it to webserver so that it is publicly available for the whole world to fetch. When a user clicks a link to our angry grandpa HTML page, a request is sent to the web server hosting the document, and the HTML page will then load the page in the user s web browser with the accompanying picture of a really angry golden ager

VoiceXML follow same kind of rule, but with a slightly different spin, as we are using the telephone as the browser; VoiceXML content must reside on a public webserver so that our servers can see it. A major difference is that while web hosts commonly serve up HTML, there aren t that many folks in the telephony game, so you will have to specifcally associate a phone number from a voice gateway, and map it to the URL where VXML page resides. To access a VXML page, Instead of clicking on a link, a user will dial the number pointing to application, which is the equivalent of a hyperlink in HTML. When this number is dialed, it tells servers to fetch the document that has been associated with that particular phone number or PIN code. Assuming that the code is well formed XML, and that the mapping has no typos, then our AngryGrandpa.xml file will load and execute, thus outputting the text to speech message to the caller.

Goals of VoiceXML

VoiceXMLs main goal is to bring the full power of Web development and content delivery to voice response applications, and to free the authors of such applications from low-level programming and resource management. It enables integration of voice services with data services using the familiar client-server paradigm. A voice service is viewed as a sequence of interaction dialogs between a user and an implementation platform. The dialogs are provided by document servers, which may be external to the implementation platform. Document servers maintain overall service logic, perform database and legacy system operations, and produce dialogs. A VoiceXML document specifies each interaction dialog to be conducted by a VoiceXML interpreter. User input affects dialog interpretation and is collected into requests submitted to a document server. The document server replies with another VoiceXML document to continue the user s session with other dialogs.

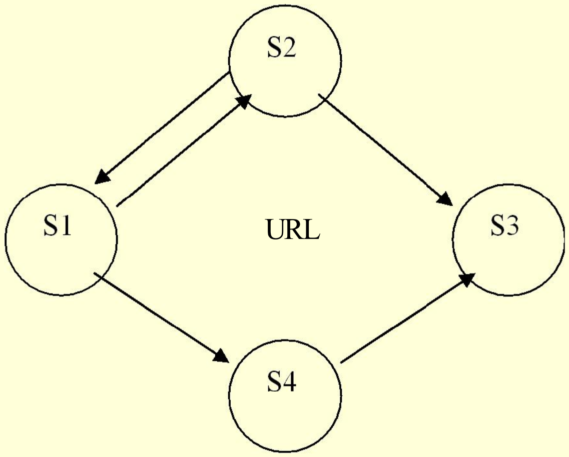
Scope of VoiceXML

The language describes the human-machine interaction provided by voice response systems, which includes:

- Output of synthesized speech (text-to-speech).
- Output of audio files.
- Recognition of spoken input.
- Recognition of DTMF input.
- Recording of spoken input.
- Control of dialog flow.
- Telephony features such as call transfer and disconnect.

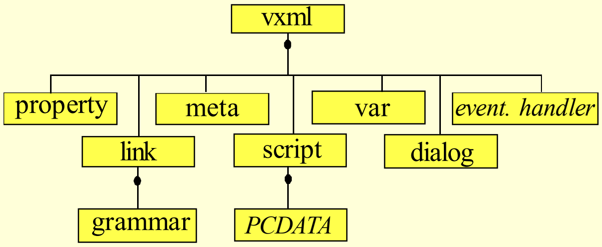
VoiceXML document structure:

A VoiceXML *document* (or an *application*) forms a conversational finite state machine. The user is always in one conversational node/state/ *dialog*, at a time. Each dialog/node determines the next dialog/node to transition to. *Transitions* are specified using URIs, which define the next document and dialog to use. If a URI does not refer to a document, the current document is assumed. If it does not refer to a dialog, the first dialog in the document is assumed. Execution is terminated when a dialog does not specify a



successor, or if it has an element that explicitly exits the conversation.

Figure 1 – Dialog state



Each VoiceXML document must have a vxml element as its root node. Figure 2 lists the various units of the vxml element type.

Figure 2 - VXML element type's various units

All of these children are other VoiceXML element types except two: event.handler and dialog. These are abstract element types, defined as entities in the DTD. Each of them can represent a union of element types.

Events are thrown by the platform under a variety of circumstances, such as when the user does not respond, doesn t respond intelligibly, requests help, etc. The interpreter also throws events if it finds a semantic error in a VoiceXML document. Events are caught by event.handler elements.

Dialog element type which includes form and menu. These are the two types of dialogs defined in VoiceXML. Forms define an interaction that collects values for a set of form item variables. A menu presents the user with a choice of options and then transitions to another dialog based on that choice. Each dialog has one or more speech and/or DTMF grammars associated with it.

The audible element types represent audio that can be played back. PCDATA (parsed character data) can be interpreted as audible content, as is the case with plain text processed by a text-to-speech processor.

References:

www.w3.org