

VoiceXML Absentee System

*Ping Gallivan, Qisheng Hong, Lisa Jordan, Elaine Li,
George Mathew, Yani Mulyani, Paul Visokey
Under the guidance of Dr. Charles Tappert
CSIS, Pace University*

ABSTRACT

VoiceXML is designed for creating audio dialogs that combine speech, audio digital, speech recognition, DTMF key input, recorded or synthetic speech, and telephony. The major goal is to bring web-based development and content delivery to interactive voice respond applications. A VoiceXML document, or a set of documents called an application, forms a conversational finite state machine. The user is always in one conversational state, or dialog, at a time. Each dialog determines the next dialog to transition to, and 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 and, 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. In this paper we present a VoiceXML absentee system that enables students to telephone in their class absence that is recorded in a university database. This application is suitable for any reasonably sized organization for a cost-effective and convenient way to record employee absences by having them interact directly with a computer with a telephone.

INTRODUCTION

Web applications deliver information and services exclusively through visual interfaces on computers with screens, keyboards, and mouse. The revolution of the web moves tremendously bypassed large market information and services represented by telephone, which provide interaction between voice input and audio output. VoiceXML is a Web-based markup language for representing human-computer dialogs, just like HTML. But while HTML assumes a graphical web browser, with display, keyboard, and mouse, VoiceXML assumes a voice browser with audio output (computer-synthesized and/or recorded), and audio input (voice and/or keypad tones). VoiceXML leverages the Internet for voice application development and delivery, greatly simplifying these difficult tasks and creating new opportunities. The ordinary phone has been very important in the development of VoiceXML, although VoiceXML's appeal is more general. The typical VoiceXML voice browser of today runs on a specialized voice gateway node that is connected both to the public switched telephone network and to the

Internet. These voice gateways extend the power of the web to the world's 1,300,000,000 phones.¹

SYSTEM ARCHITECTURE AND OVERVIEW

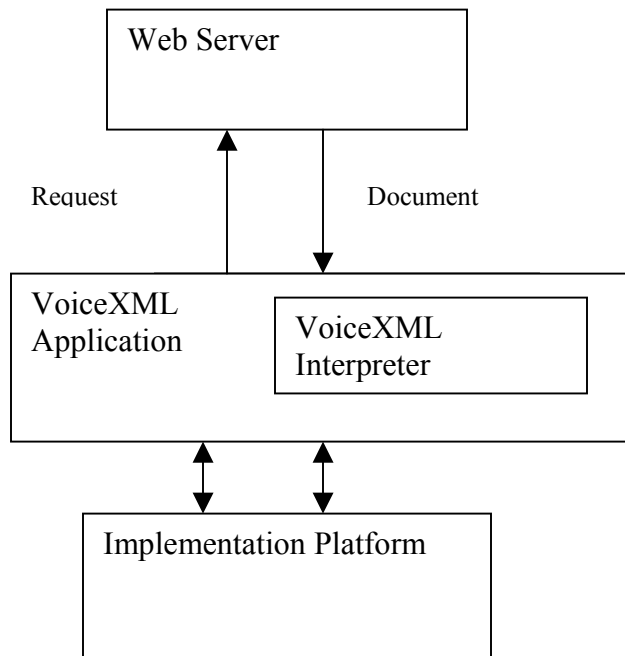


Figure 1: Architecture

A web server processes *requests* from a client application, *the VoiceXML Interpreter*, through the *VoiceXML application*. The server produces *VoiceXML documents* in reply, which are processed by the VoiceXML Interpreter. The VoiceXML interpreter monitors user inputs. The *VoiceXML application* and *the VoiceXML interpreter* control the *implementation platform*.

There are two kinds of dialogs: *forms* and *menus*. Forms define an interaction that collects values for a set of field item variables. Each field may specify a grammar that defines the allowable inputs for that field. If a form-level grammar is present, it can be used to fill several fields from one utterance. A menu presents the user with a choice of options and then transitions to another dialog based on that choice.

An *application* is a set of documents sharing the same *application root document*. Whenever the user interacts with a document in an application, its application root document is also loaded. The application root document remains loaded while the user is

transitioning between other documents in the same application, and it is unloaded when the user transitions to a document that is not in the application. While it is loaded, the application root document's variables are available to the other documents as *application variables*, and its grammars can also be set to remain active for the duration of the application.²

Figure 2. Below shows the transition of documents (D) in an application that share a common application root document (root).

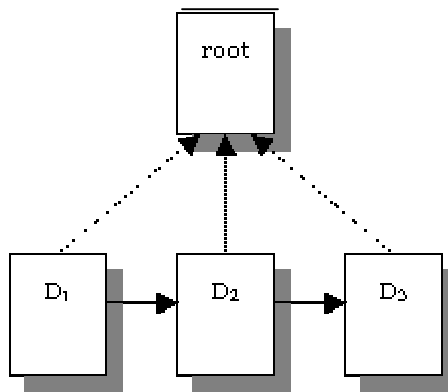


Figure 2: Transition of root document

In cases where you want multiple documents to work together as one application, you select one document to be the *application root document*, and refer to it in the other document's `<vxml>` elements. When this is done, every time the interpreter is told to load a document in this application, it also loads the application root document if it is not already loaded. The application root document remains loaded until the interpreter is told to load a document that belongs to a different application. Data can be protected from the XML parser in a CDATA section. Use a CDATA section to escape text that would otherwise be recognized as markup.

VoiceXML builds upon the basic concepts and rules set forth by XML to define a logical vocabulary for describing interactive voice applications. The `<vxml>` element is always the top-level element of a vxml document. There are two kinds of vxml applications, single document and multi-document. Document execution begins at the first dialog by default. As each dialog executes, it determines the next dialog. When a dialog doesn't specify a successor dialog, document execution stops.

Each dialog has one or more speech and/or DTMF *grammars* associated with it. In *machine directed* applications, each dialog's grammars are active only when the user is in that dialog. In *mixed initiative* applications, where the user and the machine alternate in determining what to do next, some of the dialogs are flagged to make their grammars *active* (i.e., listened for) even when the user is in another dialog in the same document, or

on another loaded document in the same application. In this situation, if the user says something matching another dialog's active grammars, execution transitions to that other dialog, with the user's utterance treated as if it were said in that dialog. Mixed initiative adds flexibility and power to voice applications.

If an implementation platform supports barge-in, the service author can specify whether a user can interrupt, or "barge-in" on, a prompt. This speeds up conversations, but is not always desired. If the user must hear all of a warning, legal notice, or advertisement, barge-in should be disabled.

The timeout attribute specifies the interval of silence allowed while waiting for user input after the end of the last prompt. If this interval is exceeded, the platform will throw a `noinput` event. For example, the user may be given five seconds for the first input attempt, and ten seconds on the next.

The various timing properties for speech and DTMF recognition work together to define the user experience. DTMF grammars use `timeout`, `interdigittimeout`, `termtimeout` and `termchar` to tailor the user experience. The `timeout` parameter determines when the `<noinput>` event is thrown because the user has failed to enter any DTMF. The `interdigittimeout` determines when the `nomatch` event is thrown because a DTMF grammar is not yet recognized, and the user has failed to enter additional DTMF. Speech grammars use `timeout`, `completetimeout`, and `incompletetimeout` to tailor the user experience.

It's important to note that VoiceXML can be used *without* Automated speech recognition (ASR) systems or text-to-speech (TTS): users can listen to recorded audio and press keys in response. Speech technology makes applications more powerful and pleasant to use.

Absentee System application³

The Absentee System application was developed basically for Pace University students to report class absences. This application is suitable to be modified to work for any department or organization. Currently the application has been developed using PHP on MySQL database at Pace University.

The VoiceXML Absentee System has been designed to receive and keep records of absentee calls from students, faculty, and university staff. Two interfaces were created for this application, 1) which is a web interface that will provide enrollment to this service and access to information of the absences and 2) a phone interface (VXML) where users call in to record their absence. The user must first enroll via the web prior to using the phone service. The user will enter some pertinent information and create a unique `userId` and password to subsequently enter the Absentee System via a telephone. All users will provide the following information when enrolling on the web: a name, email address, a unique `userId` and a password. The faculty may be asked to enter the course `Ids` and the semester they are teaching those courses, which means that the faculty may update this information every semester. Staff members may be asked to enter the campus they are working at, department they are working in and whether or not they are

managers. The faculty and staff provide this additional information so that they are able to view their student's or their staff members' absentee records, rather than viewing all students' or staff member's absentee records.

Each user will be categorized into one of four login types, which are as follows: student, staff, faculty or administrator. The administrator will have access to all information provided by the system, as well as creating an additional administrator user id and password. Access to some information will be granted to some users, such as instructors and employers, who will be able to view the absence records for their courses or departments, respectively.

When the user calls the system s/he will be asked to enter the user id and password. Upon successfully entering into the system, the user will go through a series of questions. The login information will determine which category the user is in and the appropriate questions will be asked. If the user is a student or instructor then the system will ask for the user's course Id for the class that will be missed and date the class will be missed. If the user is a member of the university staff then the system will ask for the day that the user will be absent. All the information obtained by the system will be stored into a database, which the authorized users can access via the web. The information can be viewed on the web.

The absentee application capturing the course and date of absence are shown in the appendix-A.

Voice Portals

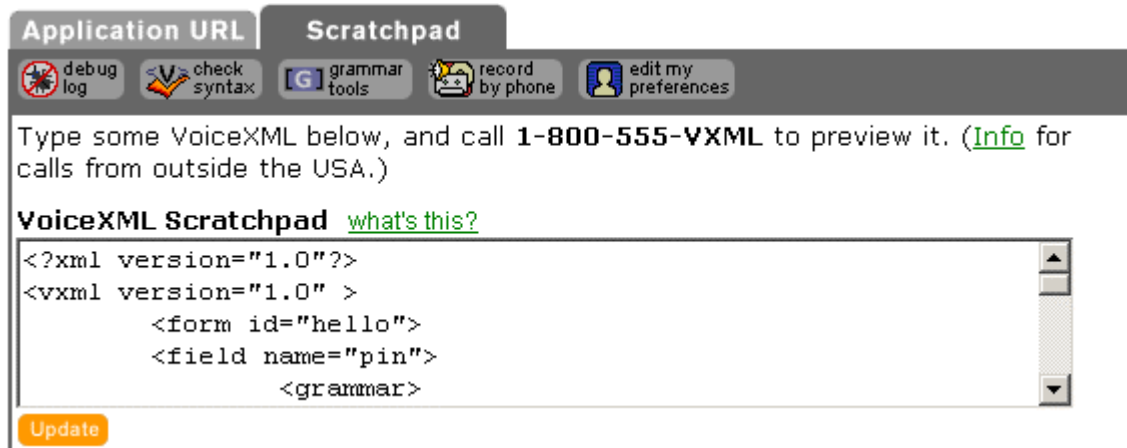
A voice portal is a system that lets customers access information on the Internet through a telephone interface. It uses technologies such as speech recognition and text to speech (TTS) conversion to create a user interface that allows users to navigate through the 'voice web pages' using a phone and voice commands. Existing voice portals show that it is possible to create voice applications with today's technology. A recent study by Cahners In-Stat Group predicts that the voice portal and voice services market is expected to exceed \$6 billion within 5 years in the U.S. The voice portal technology providers include Speechworks, Nuance, IBM, Philips, Vocal SpeechWare, Audiopoint, BeVocal, Pipebeach, Tellme Networks, Tellsurf Networks, Voicegenie, Periphonics (Nortel), Cisco, Dialogic (Intel).

TellMe:

Currently our application is run via a voice portal created by TellMe Networks Inc., though our vxml application is on Pace University server. The voice portal acts as a browser that runs the application for a user. TellMe Studio is a Web-based development environment that provides access to their complete Platform SDK, including tools that enable you to build, test, and deploy VXML applications without downloading or installing any hardware or software. There are two parts in the TellMe Studio,

MyExtensions and MyStudio, that become available after you create an account in studio.tellme.com. MyExtensions is for developers to deploy their Extension on 1-800-555-TELL, and promote it through the Extensions Web and phone directories.

MyStudio Platform looks as in the following picture:



To build and test more complex phone sites, just publish your VoiceXML and audio files on a publicly accessible Web server, point MyStudio at the URL for your application's "home page", and once again call the MyStudio phone number to preview your application.

The key to developing a speech recognition based VXML application is a "grammar". A grammar defines the set of valid expressions that a user can say when interacting with a voice application. Each interactive dialog in an application references one or more grammars using one or more grammar elements. The TellMe Studio Platform provides you with three choices when integrating grammars into your voice application:

- a) Reference a grammar in the TellMe Studio Grammar Library
- b) Define your own grammar using the Nuance Grammar Specification Language
- c) Define your own grammar using the Speech Grammar Markup Language (GRXML)

TellMe Studio⁴ can validate a grammar from scratchpad or a URL. It uses "VoiceXML Syntax Checker" which is implemented using a modified version of the perlSGML library. "Grammar Phrase Generator" displays phrases your grammar is capable of recognizing. Through it, you can view all phrases your grammar can recognize, or just generate a random sampling. The grammar is fundamental to the success of the application. If the user says words and phrases not included in the grammar the application will fail (although there are graceful recovery techniques). If you try to include every possible word and phrase the probability of successful speech recognition will decrease because the search space increases and the potential for acoustic confusability (sound alikes / homonyms) increases. A "DTMF Generator" will generate DTMF (touchtone) equivalents for a list of words and check for conflicts.

Portal Limitations:

Once the standard VoiceXML scripts are written, there is difficulty getting the scripts to work successfully using only the vendor's developer resources and the VoiceXML spec document. The main problem was differences in the way the vendors handle grammars. Though the concept and structure of the language is fairly easy to use, for someone creating a real-world app, with a live database behind it and more complex functions, coding the app to VoiceXML standards may be time consuming.

FUTURE WORK

We are in the process of porting the absentee application to the Ascent VXML hardware at Pace University from the TellMe portal. Enhancing the application with more functionality and reliability is under consideration.

CONCLUSION

VXML will have profound impacts, changing the way we use the phone - and perhaps the design of phones themselves - as well as changing the nature and evolution of the Web. By making it easier to program Web applications for voice access, VXML can bring high efficiency to call center and intranet development. Applications similar to the absentee system may be considered cost-effective and convenient to interact with a software application without human interaction or any expensive computing devices.⁵

REFERENCES:

1. <http://www.voicexml.org/>, Voice XML forum
2. <http://www.wdvl.com/Authoring/Languages/XML/Specifications.html>, Voice XML specifications
3. <http://www.csis.pace.edu/~ctappert/cs616-02.default.htm>, team #3 and team #4
4. <http://www.tellMe.com>
5. <http://www.w3.org/TR/2001/WD-voicexml20-20011023/>

APPENDIX A:

(partial VXML Code from absentee application)

```
<?xml version="1.0" ?>
<!DOCTYPE vxml (View Source for full doctype...)>
<vxml application="http://resources.tellme.com/lib/universals.vxml">
  <form id="crsdate" anchor="false">
    <var name="username" />
    <block>
      <assign name="username" expr="" />
    </block>
    <block>
      <prompt>
        Hello
      </prompt>
    </block>
    <field name="courseid" timeoutondtmf="false" confirm="no" bargein="true"
      magicword="false" phoneticpruning="false">
      <prompt>Please say the course i d</prompt>
      <grammar type="application/x-gsl" mode="voice">
        <![CDATA[
          [
            (Char_speak:d1 Char_speak:d2 Dig_speak:d3 Dig_speak:d4
              Dig_speak:d5) {<option strcat($d1 strcat($d2 strcat($d3 strcat($d4
                $d5)))>}
            (cancel) {<option "cancel">}
          ]
          Char_speak
          [
            c{return(c)}
            s{return(s)}
          ]
          Dig_speak
          [
            one {return(1)}
            two {return(2)}
            three {return(3)}
            four {return(4)}
            five {return(5)}
            six {return(6)}
            seven {return(7)}
            eight {return(8)}
            nine {return(9)}
            [zero oh] {return(0)}
          ]
        ]]>
      </grammar>
    </field>
    <filled>
      <prompt>
        You said
        <value expr="courseid" />
      </prompt>
    </filled>
  </form>
</vxml>
```

```

    <catch event="nomatch" count="10">The course i d is invalid Please say the
    course i d.</catch>
  <noinput>
    I did not understand the course i d.
    <reprompt order="curr" />
  </noinput>
</field>
<field name="date_absent" timeoutondtmf="false" confirm="no" bargein="true"
  magicword="false" phoneticpruning="false">
  <prompt>What is the date you will be absent?</prompt>
  <grammar type="application/x-gsl" mode="voice">
    <![CDATA[
      TELLME_DATE
    ]]>
  </grammar>
  <catch event="nomatch" count="10">I am sorry I can not understand. Please
  repeat the date you will be absent.</catch>
  <noinput>
    I did not understand the date.
    <reprompt order="curr" />
  </noinput>
  <filled>
    <goto next="#finddate" method="get" />
  </filled>
</field>
</form>
<form id="finddate" anchor="false">
  <block>
    <script>
      <![CDATA[
        var date1 = vxmldata.get("date_absent");
        var myDate="";
        var myYear="";
        var myMonth="";
        var myDaten="";
        var mySpecial="";
        var myMonthc = "";
        function ParseGrammar2(sGramResult) {
          myDate = "";
          var cMonth = new
          Array('january','february','march','april','may','june','july','august','septe
          mber','october','november','december');
          var arrNames = [];
          var arrValues = [];
          var arrNamesValues = GramResult.split('^');
          for (var i = 0; i < arrNamesValues.length; i++) {
            var arrNameValuePair =
              arrNamesValues[i].split('=');
            arrNames[i] = arrNameValuePair[0];
            arrValues[i] = arrNameValuePair[1];
            if (arrNames[i] == 'month')
              {myMonthc = arrValues[i] };
            if (arrNames[i] == 'date')
              {myDaten = arrValues[i] };
            if (arrNames[i] == 'year')
              {myYear = arrValues[i] };
          }
        }
      ]]>
    </script>
  </block>
</form>

```

```

        if (arrNames[i] == 'special_date')
            {mySpecial = arrValues[i] };
        }
        var i;
        for (i=0;i < cMonth.length;i++){
            if (cMonth[i] == myMonthc) break;
        }

        myMonth = i + 1;
        myDate = myYear + '-' + myMonth + '-'
                + myDaten ;

    } // eo function

    ParseGrammar2(date1);
]]>
</script>
</block>
<block>
    You said,
    <value expr="myMonthc" />
    ,
    <value expr="myDaten" />
</block>
<field name="yesno" timeoutndtmf="false" confirm="no" bargein="true"
    magicword="false" phoneticpruning="false">
    <prompt>Is this the date you said?. say, 'yes' or 'no'.</prompt>
    <grammar type="application/x-gsl" mode="voice">
        <![CDATA[
            YES_NO
        ]]>
    </grammar>
    <catch event="nomatch" count="3">I am sorry I can not understand. Please
        repeat your response. yes or no.</catch>
    <filled>
        <prompt>
            you said
            <value expr="yesno" />
        </prompt>
        <if cond="yesno != 'yes'">
            <prompt>Please re-enter Course and Date of Absent</prompt>
            <goto next="#crsdate" method="get" />
            <else />
            <submit next="saveAbsenteeRecord.php" method="post"
                namelist="courseid myYear myMonth myDaten username"
            />
        </if>
    </filled>
</field>
<block>
    <submit next="saveAbsenteeRecord.php" method="post" namelist="courseid
        myYear myMonth myDaten username" />
</block>
</form>
</vxml>

```