Prototyping Best-Practice Speech User Interfaces with VoiceXML 2.0 and the IBM VoiceXML Toolkit

TR 29.3732
February 24, 2004

James R. Lewis

IBM Pervasive Computing

Boca Raton, Florida

Abstract

This report teaches methods for using VoiceXML 2.0 to create prototypes of speech user interfaces that are consistent with current best practices for directed dialogs. It does not necessarily teach professional coding practices, but does teach in a step-by-step fashion critical prototyping skills that we have found useful for evaluating speech user interface concepts and in conducting early usability studies.

Keywords

VXML 2.0
VoiceXML 2.0
Rapid prototyping
Speech user interfaces
Best practices
IBM VoiceXML Toolkit

Contents

Introduction

What is VoiceXML?

The primary purpose of VoiceXML is to enable the creation of a verbal (rather than a visual) interface to data. The design of VoiceXML makes it ideal for coding call flows (system prompts and help messages, speech recognition grammars, and directing the call flow based on user speech). The VoiceXML Forum (IBM[1], AT&T[2], Lucent[3], Motorola[4], and other companies) is the organization that, working with the W3C, has standardized VoiceXML (VoiceXML Forum, 2000).

Like HTML, VoiceXML is a computer language that uses tags (also called 'elements'). VoiceXML has comparatively few tags – fewer than fifty in Version 2.0. Because VoiceXML uses XML as its foundation, it requires a strict pairing of beginning and ending tags (for example, <tag> paired with </tag>, where the '/' before 'tag' indicates 'end'). In some cases, it might not be necessary to put any text between a beginning and ending tag, in which case you can combine the beginning and ending tags by putting the '/' at the end of the tag before the closing angle bracket (for example, <tag/>). It is common to forget to end a tag, so keep it in mind when you are debugging a program.

This report uses elements from Version 2.0 of VoiceXML to illustrate ways to create programs that are consistent with current best practices in the design of speech user interfaces (for example, see Balentine, Morgan, & Meisel, 2001; Gardner-Bonneau, 1999; IBM, 2003; Lewis, Simone, & Bogacz, 2000; Polkosky & Lewis, 2002; Sadowski & Lewis, 2000a, 2000b, 2001; Virzi & Huitema, 1997). The development environment described in this report is the IBM Voice Toolkit for WebSphere[5] Studio Version 4.2. Provided as a plugin for WebSphere Studio Site Developer (WSSD) or WebSphere Studio Application Developer (WSAD), the Toolkit provides an Integrated Development Environment (IDE) for the development of VoiceXML 2.0 applications.

Working with the IBM VoiceXML Toolkit

You can download the Toolkit free of charge from www.ibm.com (search for "IBM Voice Toolkit for WebSphere Studio" – if this doesn't work, try using the same search string in a search engine such as www.google.com).

If you already have a copy of WSSD or WSAD, then you can run the Toolkit installation to plug it into your copy of WebSphere Studio. If not, then you can get a trial download of WSAD at http://www7b.boulder.ibm.com/wsdd/downloads/WSsupport.html. Be sure you have installed WSSD or WSAD before installing the Toolkit.

---

[1] IBM is a registered trademark of International Business Machines Corp.
[2] AT&T is a registered trademark of AT&T Corp.
[3] Lucent is a registered trademark of Lucent Technologies Inc.
[4] Motorola is a registered trademark of Motorola, Inc.
[5] WebSphere is a registered trademark if International Business Machines Corp.

After download and installation, you need to perform the following steps to start a Voice Toolkit project and create a VoiceXML file for input:

1. Start your WSSD or WSAD.
2. Make sure you are in the Voice Toolkit perspective (Window > Open Perspective > Other > Voice)
3. Click File > New > Voice Project (The Voice Toolkit Project wizard appears.)
4. Type a name into the Project Name field in the Voice Toolkit Project wizard.
5. Click Finish.
6. Click File > New > VoiceXML File (The New VoiceXML File wizard appears.)
7. Type a name in the VoiceXML File Name field.
8. By default, the name of the project you just created should be in the Enter or Select the Folder field. If not, click on the project name from the project list that is below the field.
9. Click Finish.

The Toolkit contains an excellent Getting Started document, as well as extensive online help and access to important supplementary documents.

To view the Getting Started document, click Help > Welcome > Voice Toolkit for WebSphere Studio> Getting started developing voice applications (which appears on the Welcome panel under the first bullet).

To use contextual help, single-click the item for which you want information, then press F1.

To work with supplementary documents, click Help > Help Contents > Voice Tools > Related Documents. To get to a specific document, you might need to click the + box in the left panel to expand the topics. The available documents include this document and:

- IBM VoiceXML 2.0 Programmer's Guide (esp. see Chapter 2: Designing a Speech User Interface)

Another document that you might find useful is the VoiceXML 2.0 specification, available on the web at http://www.w3.org/TR/voicexml20/.

Sample VoiceXML Programs

The remainder of this report contains sample programs with explanations of how the various pieces work and how the samples illustrate current best practice in speech user interface design.

Sample 1: Hello World!
Following is a traditional 'Hello World' program, written in VoiceXML.

Figure 1. Hello World

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE vxml PUBLIC "-//W3C//DTD VOICEXML 2.0//EN"
"http://www.w3.org/TR/voicexml20/vxml.dtd">
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.w3.org/2001/vxml
 http://www.w3.org/TR/voicexml20/vxml.xsd">
<meta name="GENERATOR" content="IBM Voice Toolkit for WebSphere
Studio"/>

<form>
 <block>
  Hello, World!
 </block>
</form>

</vxml>
```

VoiceXML requires the initial lines (those preceding "<form>")and the last line ("</vxml>"), and inserts them lines automatically into new VoiceXML files.

Note the use of <form> and <block> tags, with text inside the block (later sections will include more information about these tags).

TRY IT! Type these lines into a new VoiceXML file named helloworld.vxml, then save the file. To run the program, click the drop-down control on the Run icon () that appears on the row of icons under the Menu bar[6], then select Run as > VoiceXML Application. Note that it will probably take a while for the program to begin running (especially the first program you run after starting the toolkit). All this program does is to play "Hello, World!", then stop.

[6] You should have set the appropriate input level for your microphone the first time you create a Voice project, but if you haven't done that step yet (or need to reset the level), click Run > Test Microphone. After the Test Microphone panel appears, click Display Script, then click Start and begin reading the script. The script tells you what to do and what to expect.

Some VoiceXML Concepts

A VoiceXML document forms a conversational finite state (dialog) machine in which the user is always in one dialog at a time. Each dialog determines the next transition. If a URI (dialog path pointing to a part of the current VoiceXML file, another VoiceXML file in the same directory, or a VoiceXML file on a server) does not refer to a document, the system assumes the current document. If it does not refer to a dialog, the system assumes the first dialog in the document. Execution ends when a dialog does not specify a next dialog (as in the Hello World! Sample), or if the dialog explicitly exits the conversation (using an exit tag -- <exit/>).

There are two types of VoiceXML dialogs: menus and forms. In this report, I will only use forms. Menus have some interesting and potentially useful properties, but these properties can make them more complicated to use. For more information about menus, see the IBM VoiceXML 2.0 Programmer's Guide.

Within a form, the key elements are blocks and fields:

- Use blocks to specify actions that do not require interaction with the user.
- Use fields to specify actions that do require interaction (dialog) with the user.

By default, the barge-in type for VoiceXML 2.0 is "hotword", which means that system speech will not stop unless the application receives valid input from the user. You can get better system response by changing the barge-in type to "speech", which means that system speech stops as soon as the application detects speech energy. As discussed in Chapter 2 of the VoiceXML 2.0 Programmer's Guide, both barge-in types have their strengths and weaknesses. The sample programs in this document will use the "speech" barge-in type, which is set using a property tag:

```
<property name="bargeintype" value="speech"/>
```

Sample 2: Hello Worlds

Here is the code for the second sample. It's a little more complex than the first because it involves interaction with a user. Note that it includes the <property> tag that sets the barge-in method to "speech".

Figure 2. Hello Worlds

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE vxml PUBLIC "-//W3C//DTD VOICEXML 2.0//EN" "vxml20-1115.dtd">
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
<meta name="GENERATOR" content="Voice Toolkit for WebSphere Studio"/>

<property name="bargeintype" value="speech"/>

<form id="helloworlds">
 <block name='introduction'>
  Thank you for calling Hello Worlds!
 </block>
 <field name='planet'>
  <prompt>
   Which planet? <break time="3s"/> Select Mercury,
   Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune,
   or Pluto.
  </prompt>

  <grammar version="1.0" mode="voice" root="planet">
   <rule id="planet" scope="public">
    <one-of>
     <item>Mercury</item>
     <item>Venus</item>
     <item>Earth</item>
     <item>Mars</item>
     <item>Jupiter</item>
     <item>Saturn</item>
     <item>Uranus</item>
     <item>Neptune</item>
     <item>Pluto</item>
    </one-of>
   </rule>
  </grammar>
  <filled>
      Hello, <value expr="planet"/>!  Goodbye.
  </filled>
 </field>
</form>

</vxml>
```

Let's take a closer look at the first five lines of the form. If you want to be able to specify transitions to different parts of your VoiceXML application, you need to provide labels for the

various parts. Forms have IDs-- blocks and fields have names. You can use single or double quotes around IDs and names. Within the first five lines of the form, we've provided labels for the form (“helloworlds”), block (‘introduction’), and field (‘planet’), and have played the introduction (“Thank you for calling Hello Worlds!”).

Now let’s examine the following three lines of the form (the prompt component in the field):

```
<field name='planet'>
 <prompt>
  Which planet? <break msecs="3000"/> Select Mercury,
  Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune,
  or Pluto.
 </prompt>
```

This example uses a short initial prompt (implicit question), followed by a 3 second pause (break tag), followed by an explicit listing of choices.

If the user knows the names of the nine planets of the solar system (as most users would), the short initial prompt provides a shortcut for getting to the next step of the program. If a user does not know what to say, there is only a three-second delay before the explicit presentation of the planet’s names. Had this been a menu for which most users would not know the choices, the more appropriate design would be to play the choices immediately (“Select Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune, or Pluto.”)

The inline grammar specifies the words/phrases that the recognizer will accept. One attribute of the grammar tag specifies the root (initial, or beginning) rule for the grammar. This grammar has only one rule, labeled with the id ‘planet’. Inside that rule, the choices are the items that appear between of <one-of> and </one-of> tags.

```
<grammar version="1.0" mode="voice" root="planet">
 <rule id="planet" scope="public">
  <one-of>
   <item>Mercury</item>
   <item>Venus</item>
   <item>Earth</item>
   <item>Mars</item>
   <item>Jupiter</item>
   <item>Saturn</item>
   <item>Uranus</item>
   <item>Neptune</item>
   <item>Pluto</item>
  </one-of>
 </rule>
</grammar>
```

The last part of the form is the ‘filled’ section. This is the part of the form in which you specify what to do once the user has filled a field with a valid response to a prompt.

```
<filled>
    Hello, <value expr="planet"/>!  Goodbye.
</filled>
```

When filled, the application plays "Hello, ", followed by whatever value filled the field with the name 'planet', followed by 'Goodbye." Note the use of the <value> element to feed back the planet name selected by the user. When you give a field a name, you are implicitly declaring a variable with that name to hold the value provided by the user.

TRY IT! Type these lines into a new VoiceXML file named helloworld.vxml, then save the file.

To run the program, click the drop-down control on the Run icon () that appears on the row of icons under the Menu bar, then select Run as > VoiceXML Application. You can say a planet name either during or after the prompt. If the system recognizes the planet name you said, it will go to the filled section and play back the planet name (for example, “Hello, Mercury! Goodbye.”) Because there is no transfer of the call flow, the program stops automatically.

TRY IT YOURSELF! This is a good time to experiment with modifying an existing program. Using one of the following topics (or any other topic you wish), write and test an application with a single one-field form. Be sure to do the following (using Hello Worlds as a model):

- Write an introduction and put it in a block at the beginning of the form
- Write a prompt that has a short initial question or statement, followed by a pause (using the break element), followed by an explicit list of user choices
- Put the choices into a simple inline grammar
- In the <filled> element, write closing text that feeds back the value that the user selected

Some sample topics are:

- Ice cream flavors (vanilla, chocolate, strawberry, etc.)
- Types of books (nonfiction, mysteries, biographies, etc.)
- E-mail actions (reply, reply to all, forward, delete, etc.)

Sample 3. Adding More Complex Features to Hello Worlds

In this section, we'll add the following features to the existing Hello Worlds program:

- Use a variable and an <if> statement to determine whether to play the introduction
- Set up two always-active commands using <link>
- Use grammar tags to group planets into classes
- Use <assign> to re-assign values to variables
- Give variables document-level scope
- Use <if> and <elseif> to direct the call flow

Figure 3. Hello Worlds Version 3

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE vxml PUBLIC "vxml" "">
<vxml version="1.0">

<property name="bargeintype" value="speech"/>

<var name="skipintro" expr="'play'"/>

<link next="#helloworlds">
  <grammar version="1.0" mode="voice" root="returntomain">
   <rule id="returntomain" scope="public">
    <one-of>
     <item>main menu</item>
     <item>start over</item>
    </one-of>
   </rule>
  </grammar>
</link>

<link next="#exit">
  <grammar version="1.0" mode="voice" root="goodbye">
   <rule id="goodbye" scope="public">
    <one-of>
     <item>goodbye</item>
     <item>exit</item>
    </one-of>
   </rule>
  </grammar>
</link>

<form id="helloworlds">
 <block name='introduction'>
  <if cond="skipintro == 'skip'">
   <goto nextitem="planet"/>
  </if>
  Thank you for calling Hello Worlds!
 </block>
 <field name='planet'>
```

```
  <prompt>
   Which planet? <break time="3s"/> Select Mercury,
   Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune,
   or Pluto.
  </prompt>
  <grammar version="1.0" mode="voice" root="planet">
   <rule id="planet" scope="public">
    <one-of>
     <item>Mercury<tag>$="inner"</tag></item>
     <item>Venus<tag>$="inner"</tag></item>
     <item>Earth</item>
     <item>Mars<tag>$="inner"</tag></item>
     <item>Jupiter<tag>$="outer"</tag></item>
     <item>Saturn<tag>$="outer"</tag></item>
     <item>Uranus<tag>$="outer"</tag></item>
     <item>Neptune<tag>$="outer"</tag></item>
     <item>Pluto<tag>$="outer"</tag></item>
    </one-of>
   </rule>
  </grammar>
  <filled>
   <assign name="document.planettype" expr="planet"/>
   <assign name="document.planet" expr="planet$.utterance"/>
   <assign name="document.skipintro" expr="'skip'"/>
   <if cond="planettype == 'inner'">
      <goto next="#innerworld"/>
      <elseif cond="planettype == 'outer'"/>
      <goto next="#outerworld"/>
      <elseif cond="planettype == 'Earth'"/>
      <goto next="#earth"/>
   </if>
  </filled>
 </field>
</form>

<form id='earth'>
 <block>
  <value expr="planet"/> is home, sweet home.
  <goto next="#helloworlds"/>
 </block>
</form>

<form id='innerworld'>
 <block>
  <value expr="planet"/> is an <value expr="planettype"/>
   planet.
  <goto next="#helloworlds"/>
 </block>
</form>

<form id='outerworld'>
 <block>
  <value expr="planet"/> is an <value expr="planettype"/>
```

```
   planet.
  <goto next="#helloworlds"/>
 </block>
</form>

<form id='exit'>
 <block>
  Thanks for calling Hello Worlds.  Goodbye!
  <exit/>
 </block>
</form>

</vxml>
```

Playing an Introduction Once

First, let's look at the beginning of the program.

```
<var name="skipintro" expr="'play'"/>
```

This line declares a variable ('skipintro'), used to decide whether or not to play the introduction (see the 'if' in the introduction, located in the first block of the form). Note that the expr attribute gives the variable its initial value ('play'). (Expr is an abbreviation of the word 'expression'.)

In VoiceXML, if the value assigned to a variable is enclosed in single quotes inside a pair of double quotes, then it is a literal value (exactly, or literally, the character string enclosed in the single quotes). For example, in the <var> element used in this version of the program, the value given to skipintro is the character string 'play'. If you're assigning to a variable the value currently assigned to a different variable, then enclose the variable name in double quotes only. For example, suppose play was a variable that currently had the literal value 'yes', and you wanted to assign to the variable skipintro the current value of play. In that case, the code for declaring the variable skipintro would have looked like `<var name="skipintro" expr="play"/>`, and the current literal value of skipintro would be 'yes'. If a value is numeric rather than character (i.e., you plan to use it in numeric operations), then enclose it only in double quotes (for example, "100").

Next, let's go through the beginning of the first form:

```
<form id="helloworlds">
 <block name='introduction'>
  <if cond="skipintro == 'skip'">
   <goto nextitem="planet"/>
  </if>
  Thank you for calling Hello Worlds!
 </block>
 <field name='planet'>
```

The <if> statement in the introduction block skips the introduction if the value of skipintro is 'skip', and transfers the call flow to the item named 'planet' (which is the name of the field). Because the initial value of skipintro is 'play', the introduction will play the first time the program runs this form. Later, we'll change the value to 'skip' to prevent it from playing every time.

Note that the <if> statement has two key parts – the cond (short for condition) and <goto>. Pay careful attention to the use of quotation marks in the cond attribute. The interpretation of this cond attribute is, "If the variable named skipintro has a value of 'skip', then execute whatever code is between <if> and </if>. Otherwise, ignore the code between <if> and </if> and continue with the code that follows </if>." In this case, the code that is between <if> and </if> is a <goto> that transfers the call flow the item in the current form (indicated by the use of

'nextitem') that has the name 'planet'. The line of code that follows the </if> is the line that plays the introduction ("Thank you for calling Hello Worlds!" When the condition for the <if> statement is true, the program skips the introduction.

Defining Always-Active Commands with Links

The link elements define always-active commands for (1) returning to the main menu and (2) stopping the program. If a user says 'main menu' or 'start over', the call flow immediately goes to the form with the id 'helloworlds'. If a user says 'goodbye' or 'exit', the call flow goes immediately to a form with the id 'exit'. When the target for transfer defined with the 'next' attribute starts with "#", the target is the id of a form in the document. Otherwise, VoiceXML interprets the target as the name of a different document.

```
<link next="#helloworlds">
  <grammar version="1.0" mode="voice" root="returntomain">
   <rule id="returntomain" scope="public">
    <one-of>
     <item>main menu</item>
     <item>start over</item>
    </one-of>
   </rule>
  </grammar>
</link>

<link next="#exit">
  <grammar version="1.0" mode="voice" root="goodbye">
   <rule id="goodbye" scope="public">
    <one-of>
     <item>goodbye</item>
     <item>exit</item>
    </one-of>
   </rule>
  </grammar>
</link>
```

Using Grammar Tags to Classify Responses

The rest of the form is the same as the previous version of Hello Worlds, up to the grammar:

```
  <grammar version="1.0" mode="voice" root="planet">
   <rule id="planet" scope="public">
    <one-of>
     <item>Mercury<tag>$="inner"</tag></item>
     <item>Venus<tag>$="inner"</tag></item>
     <item>Earth</item>
     <item>Mars<tag>$="inner"</tag></item>
     <item>Jupiter<tag>$="outer"</tag></item>
     <item>Saturn<tag>$="outer"</tag></item>
     <item>Uranus<tag>$="outer"</tag></item>
     <item>Neptune<tag>$="outer"</tag></item>
     <item>Pluto<tag>$="outer"</tag></item>
```

```
    </one-of>
   </rule>
  </grammar>
```

The new part of the grammar is that the planets (except for 'Earth') now have tags (the character strings enclosed in structures like `<tag>$="string"</tag>`) that classify the planet as an inner or outer planet (inside or outside the asteroid belt). When grammar tags are present, they replace the recognized words as the value of the variable implicitly declared by naming the field (in this case, the variable named 'planet'). Grammar tags can be very useful for putting things into classes or for indicating that different words mean the same thing (which is a type of classification).

Setting Variables to Document-Level Scope

The first three lines in the <filled> section take care of some assignment of values to variables that we will use in other forms. For this reason, all of the variable names start with 'document.' The default scope of a variable in VoiceXML depends on the place in the program where the variable was declared. If declared in a form, then the variable will exist only while that form is active. Once the call flow leaves that form, the variable ceases to exist. To prevent this from happening, you can put 'document.' as the first part of the variable name during assignment, and the variable will have document-level scope instead of the default form-level scope. This means that the variable will continue to exist and be available for use anywhere in the document. Because skipintro was declared outside of a form, it has document-level scope by default (but it doesn't hurt to include 'document.' when referring to it).

```
<filled>
   <assign name="document.skipintro" expr="'skip'"/>
   <assign name="document.planettype" expr="planet"/>
   <assign name="document.planet" expr="planet$.utterance"/>
```

Specifically, the first line changes the value of skipintro to 'skip' and sets skipintro to document-level scope (so from now on, if the call flow transfers back to the first form 'helloworlds', the program will skip the introduction). The second line assigns to a new variable named planettype (set to document-level scope) the value of the variable named planet (which will be 'inner', 'outer', or 'Earth'). The third line assigns to the variable named planet (also set to document-level scope) whatever the grammar has recorded as what the user actually said rather than the value of the associated grammar tag (using the shadow variable "planet$.utterance"). In VoiceXML, 'shadow' variables are variables that the system generates automatically. To use a shadow variable, you must use the correct syntax. For this shadow variable, the required syntax is to type the variable name to which you are referring (in this example, 'planet'), followed by a dollar sign, followed by '.utterance'.

Directing the Call Flow with If Statements in the Filled Section

The <if> statement in the <filled> section is more complex than the first <if> statement because it has multiple conditions and transfers (defined with <elseif>).

```
   <if cond="planettype == 'inner'">
      <goto next="#innerworld"/>
      <elseif cond="planettype == 'outer'"/>
      <goto next="#outerworld"/>
      <elseif cond="planettype == 'Earth'"/>
      <goto next="#earth"/>
   </if>
  </filled>
```

Depending on the value of planettype (which can be 'Earth', 'inner' or 'outer'), the call flow goes to forms with the ids 'earth', 'innerworld', or 'outerworld', respectively.

Using the Document-Level Variables and Unconditional Return to Main Menu

Next, let's take a look at those three forms:

```
<form id='earth'>
 <block>
  <value expr="planet"/> is home, sweet home.
  <goto next="#helloworlds"/>
 </block>
</form>

<form id='innerworld'>
 <block>
  <value expr="planet"/> is an <value expr="planettype"/>
   planet.
  <goto next="#helloworlds"/>
 </block>
</form>

<form id='outerworld'>
 <block>
  <value expr="planet"/> is an <value expr="planettype"/>
   planet.
  <goto next="#helloworlds"/>
 </block>
</form>
```

All three forms have the same structure, containing a single block that contains a statement followed by a <goto> that returns the call flow to the form named 'helloworlds'. All of the statements use `<value expr="planet"/>` to play the name of the selected planet. The statements in the innerworld and outerworld forms also use `<value expr="planettype"/>` to play the type of planet (inner or outer).

Exiting without Confirmation

The last part of the program is the exit form, shown below:

```
<form id='exit'>
 <block>
  Thanks for calling Hello Worlds.  Goodbye!
  <exit/>
 </block>
</form>
```

This is a simple form that plays a goodbye sentence ("Thanks for calling Hello Worlds. Goodbye!"), followed by an <exit/> element, which stops the program. Note that this form does not give users an opportunity to stop the exit and return to the application – a practice that we would not normally advocate (and which we take care of in Version 4).

TRY IT! Type this version of Hello Worlds into new VoiceXML file and run it. (If you're viewing this as an electronic document, you can copy and paste the lines into the new document.) You can say a planet name either during or after the prompt. If the system recognizes the planet name you said, it transfers to one of the three planet-type forms shown above, then returns to the first form (and will continue doing that until you say 'exit' or 'goodbye').

Sample 4. Even More Features

In this section, we'll add the following new features to Hello Worlds:

- Self-revealing help
- Go back (unconditional)
- Exit with confirmation
- Adjusted pronunciation
- Audio tones
- Simulated data acquisition from a backend server
- Playing the data
- Use of breaks to fine-tune timing

Figure 5 (which appears at the end of this section) contains the entire program. Before trying to work your way through the whole thing, let's go over the key new features, starting with self-revealing help.

Creating Self-Revealing Help

Self-revealing help refers to a help strategy in which you reveal more contextually relevant information at a dialog turn each time the user says 'Help', says something the system can't understand (called, in VoiceXML, a nomatch event), or doesn't say anything for a defined silence timeout period (usually 7 seconds – called, in VoiceXML, a noinput event). For applications that can automatically transfer to a call center staffed with human agents, we recommend two levels of help, followed by a transfer to an agent. For applications that cannot transfer to a human agent, one strategy is to cycle through the two help levels[7].

[7] Note that this is a mostly untested strategy for agentless systems, and some early usability testing we've done with it suggests that it can lead to usability problems when users don't realize that they're cycling between two help levels. It might be necessary to modify this strategy in a real system, possibly by labeling the help levels (which is, again, an untested strategy). Please be aware of this if you plan to build a system using this method, and be sure to conduct usability studies to make sure that it works for your application.

Here is an example of self-revealing help from the main menu of the Hello Worlds application, Version 4:

```
<var name="helpcounter" expr="0"/>

      … first lines of helloworlds form, through the end of the prompt …
  <catch event="help noinput nomatch">
   <assign name="helpcounter" expr="helpcounter+1"/>
   <if cond="helpcounter == 1">
    <prompt>
       <break time="150ms"/>Please say the name of a planet.
      <break time="2s"/> Select Mercury, Venus, Earth,
      Mars, Jupiter, Saturn, Uranus, Neptune, or Pluto.
    </prompt>
   </if>
   <if cond="helpcounter == 2">
    <prompt>
       <break time="150ms"/>At any time you can say Help,
      Repeat, Go Back, Start Over, or Exit.  To continue,
      say Mercury, Venus, Earth, Mars, Jupiter, Saturn,
      Uranus, Neptune, or Pluto.
    </prompt>
      <assign name="helpcounter" expr="0"/>
   </if>
  </catch>
```

When declared, the variable named helpcounter gets an initial value of 0. The <catch> element is set to respond to help, noinput, and nomatch events. The <assign> statement under <catch> increments helpcounter to play the next level of help. The line at the end of the second help level resets helpcounter to 0. The effect is to cycle between the two levels of help. (To prototype transfer, you could use a third level of help that transfers the call flow to a fake transfer form.) To ensure that the help messages in a form will play in the correct order, it is important to reset the value of helpcounter by including a line of code at the beginning of the form that assigns it the value 0, as shown below:

```
<form id='gettopic'>
 <block>
  <assign name="helpcounter" expr="0"/>
 </block>
```

Defining an Unconditional Go Back

It's very handy for users to be able to work their way back through a call flow using an always-active 'go back' command. It's pretty easy to do this for simple call flows, but it requires the following types of code:

- The declaration of a variable named goback.
- A link that defines the always-active 'go back' command.

- Code in every form containing a field that defines the place to which the program should transfer the call flow if a user says ‘go back’.
- A form that actually performs the ‘go back’ function.

Here’s the implementation of ‘go back’ in Hello Worlds Version 4, starting with the declared variable (with an initial value of ‘undefined’) and the link (which programs the application to go to a form with the id ‘goback’ if the user says ‘go back’):

```
<var name="goback"/>

<link next="#goback">
  <grammar version="1.0" mode="voice" root="return">
   <rule id="return" scope="public">
    <one-of>
     <item>go back</item>
    </one-of>
   </rule>
  </grammar>
</link>
```

Hello Worlds Version 4 prompts users for two pieces of information – the planet (like the previous versions) and a planetary attribute, such as temperature, number of moons, distance from the sun, etc. If a user has provided a planet for the first prompt, then, on hearing the second prompt decides to change the planet, he or she can say ‘go back’, as long as the transfer target for ‘go back’ has been defined. One way to take care of this is to assign the ‘go back’ target in a block placed at the beginning of each form. Here’s the code that does this at the beginning of the Version 4 form that prompts the user for a planetary attribute (the ‘gettopic’ form):

```
<form id='gettopic'>
 <block>
  <assign name="document.goback" expr="'helloworlds'"/>
 </block>
```

So, if a user is in the ‘gettopic’ form and says ‘go back’, the call flow will transfer back to the ‘helloworlds’ form, after the processing defined in the ‘goback’ form:

```
<form id='goback'>
 <block>
  <goto expr="'#'+document.goback"/>
 </block>
</form>
```

All the ‘goback’ form does is to use an expr attribute to concatenate a ‘#’ with the current value of the goback variable inside of a <goto> statement, which has the effect of going back to the previous dialog.

Defining a Conditional Go Back

This 'go back' strategy works well as long as the structure of the application is purely hierarchical – in other words, there's only one place to go back to from any other place. If it's possible to get to a place in the user interface from more than one path, then the 'go back' strategy for that place will need to be conditional.

For example, imagine an application in which a user can check on the current balance of a bill by making a selection directly from the main menu, or can first select Billing from the main menu, then Current Balance. The conditional goback in the form that plays the current balance could have the following structure:

```
<form id='currentbalance'>
  <block>
   <if cond="mainChoice == 'current balance'">
    <assign name="document.goback" expr="'mainMenu'"/>
    <elseif cond="mainChoice == 'billing'"/>
    <assign name="document.goback" expr="'billing'"/>
   </if>
  </block>
```

This code specifies that if the user's most recent choice from the main menu (mainChoice) was 'current balance', then the value of goback is 'mainMenu' (returning the user to the Main Menu). If the user's most recent choice from the main menu was 'billing', then the value of goback is 'billing' (returning the user to the Billing Menu).

Exiting with Confirmation

If the system activates the exit command, you might want to have the user provide confirmation in case:

- The user actually said something else
- The user didn't realize that it would end the call

If the user rejects the confirmation, then the application should return the user to the form in use when the system detected the exit command. One way to do this is to use a strategy similar to the one described for 'go back'. The components needed to accomplish this are:

- The declaration of a variable named currentform.
- A link that defines the always-active 'exit' command.
- Code in every form containing a field that assigns the form's id to a variable.
- An exit confirmation form.

Here is the declaration of currentform in Hello Worlds Version 4, followed by the link that establishes the always-active exit command:

```
<var name="currentform"/>

<link next="#confirmexit">
  <grammar version="1.0" mode="voice" root="goodbye">
   <rule id="goodbye" scope="public">
    <one-of>
     <item>goodbye</item>
     <item>exit</item>
    </one-of>
   </rule>
  </grammar>
</link>
```

In this version of the program, users can say ‘goodbye’ or ‘exit’ to jump to the exit confirmation form.

Next is an example of the assignment of a form’s id to a variable:

```
<form id='gettopic'>
 <block>
  <assign name="currentform" expr="'gettopic'"/>
 </block>
```

Finally, here is the exit confirmation form from Hello Worlds Version 4:

```
<form id='confirmexit'>
 <field name="exitChoice" type="boolean">
  <prompt>
   <break time="150ms"/>
   Do you want to end this call?
  </prompt>
  <catch event="help noinput nomatch">
  <assign name="helpcounter" expr="helpcounter+1"/>
  <if cond="helpcounter == 1">
   <break time="150ms"/>Please say Yes, No, or Repeat.
  </if>
  <if cond="helpcounter == 2">
   <break time="150ms"/>At any time you can say Help, Repeat,
   Go Back, Start Over, or Exit.  To end the call, say Yes.
   To return to Hello Worlds, say No.
   <assign name="helpcounter" expr="0"/>
  </if>
  </catch>
  <filled>
   <if cond="exitChoice">
      <goto next="#exit"/>
      <else/>
      <audio src="triple.au"/>
      <break time="150ms"/>
      Returning.
      <goto expr="'#'+document.currentform"/>
```

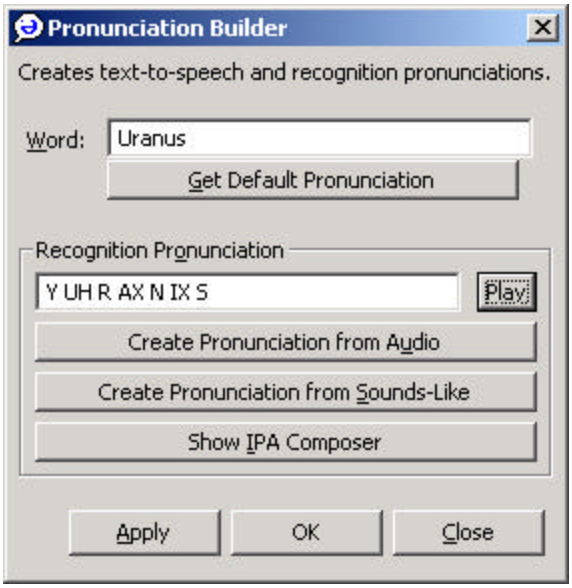
```
   </if>
  </filled>
 </field>
</form>
```

Note the use of the Boolean (yes/no) grammar type (specified as an attribute in the <field> tag) and the special form of the comparison for a Boolean value (<if cond="exitChoice">). This has the same meaning as <if cond="exitChoice == 'yes'">. If the caller has said 'yes' (or any synonym for 'yes' available in the Boolean grammar), the program goes to the form with the id of 'exit'. Otherwise, the program uses the same technique as that described for 'go back' to return to the form that was current when the exit procedure started (<goto expr="'#'+document.currentform"/>).

Adjusting Pronunciations with the Pronunciation Builder

One way to change a pronunciation is to use the IBM VoiceXML Toolkit's Pronunciation Builder, shown below in Figure 4. The various controls let you get the default pronunciation for the word, which you can then change by saying the word into a microphone, typing a sounds-like spelling, or editing with an IPA (International Phonetics Association) Composer tool. The toolkit stores pronunciations in a special set of 'pool' files and folders.

Figure 4. Pronunciation Builder



Of course, another way to change pronunciations is to spell the word you want to change in a way that causes the system to pronounce (and recognize) it correctly. This works because no one calling into a speech application can see the way that you have spelled (or misspelled) words.

Integrating Audio Tones

You can use the <audio> tag to put tones in an application. This example shows the use of a tone named triple.au to provide audio highlighting (both before and after playing the planet name) for a planet landmark:

```
<form id="playlandmark">
 <block>
  <audio src="triple.au"/>
  <break time="150ms"/>
  <value expr="document.planet"/>
  <break time="150ms"/>
  <audio src="triple.au"/>
  <goto next="#gettopic"/>
 </block>
</form>
```

Simulating Data Acquisition from a Backend Server

In a prototype, it might be necessary to simulate the acquisition of data from a database on a backend server. The following lines illustrate an example of this:

```
<form id='lookupdata'>
 <block>
  <if cond="document.planet == 'mercury'">
   <assign name="document.distance" expr="'58 million
   kilometers'"/>
   <assign name="document.position" expr="'the closest
   planet to the sun'"/>
   <assign name="document.orbit" expr="'88 days'"/>
   <assign name="document.temperature" expr="'440 degrees
   Celsius'"/>
   <assign name="document.size" expr="'next to smallest'"/>
   <assign name="document.atmosphere" expr="'98% helium, 2%
   hydrogen'"/>
   <assign name="document.moons" expr="'no moons'"/>
  </if>
  …
<if cond="document.planet == 'pluto'">
   <assign name="document.distance" expr="'5.9 billion
   kilometers'"/>
   <assign name="document.position" expr="'usually the
   farthest planet from the sun'"/>
   <assign name="document.orbit" expr="'248 and a half
   years'"/>
   <assign name="document.temperature" expr="'negative 233
   degrees Celsius, only 40 degrees above absolute zero'"/>
   <assign name="document.size" expr="'smallest'"/>
   <assign name="document.atmosphere" expr="'mostly
   nitrogen, with some carbon monoxide and methane'"/>
   <assign name="document.moons" expr="'one moon'"/>
  </if>
```

Playing the Data

After getting the data, you need to play it, as shown in the following lines of code (note the use of document-level variables):

```
<form id='playit'>
 <block>
  <if cond="document.topic == 'distance'">
   At an average distance of <value
   expr="document.distance"/>,
   <value expr="document.planet"/> is <value
   expr="document.position"/>.
  </if>
  <if cond="document.topic == 'orbit'">
   It takes <value expr="document.planet"/> <value
   expr="document.orbit"/> to orbit the sun.
  </if>
  <if cond="document.topic == 'temperature'">
   The average surface temperature of <value
   expr="document.planet"/> is
   <value expr="document.temperature"/>.
  </if>
  <if cond="document.topic == 'size'">
   <value expr="document.planet"/> is the <value
   expr="document.size"/> planet.
  </if>
  <if cond="document.topic == 'atmosphere'">
   The atmosphere of <value expr="document.planet"/>
   contains <value expr="document.atmosphere"/>.
  </if>
  <if cond="document.topic == 'moons'">
   <value expr="document.planet"/> has <value
   expr="document.moons"/>.
  </if>
  <if cond="document.topic == 'all'">
   At an average distance of <value
   expr="document.distance"/>,
   <value expr="document.planet"/> is <value
   expr="document.position"/>,
   taking <value expr="document.orbit"/> to complete its
   orbit.  It has an average surface temperature of <value
   expr="document.temperature"/>.  The atmosphere is <value
   expr="document.atmosphere"/>.  <value
   expr="document.planet"/> is the <value
   expr="document.size"/> planet, and has <value
   expr="document.moons"/>.
   <goto next="#helloworlds"/>
  </if>
  <goto next="#whatnext"/>
 </block>
</form>
```

Using Breaks to Fine-Tune Timing

You can use <break> tags to fine-tune the timing of pauses in a user interface. For example, if a menu has more than three options, it's advantageous to separate the options with 750-msec pauses. These pauses are long enough to invite users to barge-in to make a selection, but are short enough that they do not adversely affect the time required to present all of the options. Another use for <break> is to define the length of the pause that follows a nondirective prompt before presenting a set of specific options (or, if there are too many options, playing an example help). We typically set these breaks to 3 seconds (3000 milliseconds). If a menu occurs at a task-terminal point (a place where the user might want to escape from the current dialog turn), then recent studies of videotaped usability sessions suggest that it's a good idea to present a set of always-active navigation commands 1500 milliseconds after the end of the primary menu. The following lines of code illustrate all three of these uses for <break>.

```
Select a planet. <break time="3000ms"/>

Say Mercury, <break time="750ms"/>
    Venus, <break time="750ms"/>
    Earth, <break time="750ms"/>
    Mars, <break time="750ms"/>
    Jupiter, <break time="750ms"/>
    Saturn, <break time="750ms"/>
    Uranus, <break time="750ms"/>
    Neptune, <break time="750ms"/>
    Or Pluto.

<break time="1500ms"/> You can always say Repeat, Go Back,
Start Over, or Exit.
```

Another important use of the <break> tag in prototypes using TTS is to place a small pause (150 to 250 msec) at the beginning of system prompts and messages. The purpose of this small pause is to help users who have barged into a prompt or message to detect the change between the message they have interrupted and the next message. Without this break, the audio for the messages runs together, making it difficult to tell when one has stopped and the next has started. For example:

```
    <break time="150ms"/>At any time you can say Help,
    Repeat, Go Back, Start Over, or Exit.  To continue,
    say Mercury, Venus, Earth, Mars, Jupiter, Saturn,
    Uranus, Neptune, or Pluto.
```

Figure 5. Hello Worlds Version 4

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE vxml PUBLIC "-//W3C//DTD VOICEXML 2.0//EN" "vxml20-1115.dtd">
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
<meta name="GENERATOR" content="Voice Toolkit for WebSphere Studio"/>

<property name="bargeintype" value="speech"/>

<var name="skipintro" expr="'play'"/>
<var name="helpcounter" expr="0"/>
<var name="distance"/>
<var name="position"/>
<var name="orbit"/>
<var name="temperature"/>
<var name="size"/>
<var name="atmosphere"/>
<var name="moons"/>
<var name="goback"/>
<var name="currentform"/>

<link next="#helloworlds">
  <grammar version="1.0" mode="voice" root="returntomain">
   <rule id="returntomain" scope="public">
    <one-of>
     <item>main menu</item>
     <item>start over</item>
    </one-of>
   </rule>
  </grammar>
</link>

<link next="#confirmexit">
  <grammar version="1.0" mode="voice" root="goodbye">
   <rule id="goodbye" scope="public">
    <one-of>
     <item>goodbye</item>
     <item>exit</item>
    </one-of>
   </rule>
  </grammar>
</link>

<link next="#goback">
  <grammar version="1.0" mode="voice" root="return">
   <rule id="return" scope="public">
    <one-of>
     <item>go back</item>
    </one-of>
   </rule>
  </grammar>
</link>
```

```
<form id="helloworlds">
 <block name='introduction'>
  <assign name="helpcounter" expr="0"/>
  <assign name="currentform" expr="'helloworlds'"/>
  <if cond="skipintro == 'skip'">
   <goto nextitem="planet"/>
  </if>
   Welcome to Hello Worlds!  Your voice site for information
   about the planets of the solar system.  You can say help
   or repeat at any time.
 </block>
 <field name='planet'>
  <prompt>
   <break time="150ms"/>
    Select a planet. <break time="3s"/>
    Say Mercury, <break time="750ms"/>
    Venus, <break time="750ms"/>
    Earth, <break time="750ms"/>
    Mars, <break time="750ms"/>
    Jupiter, <break time="750ms"/>
    Saturn, <break time="750ms"/>
    Uranus, <break time="750ms"/>
    Neptune, <break time="750ms"/>
    Pluto, <break time="750ms"/>
    or Exit.
  </prompt>
  <catch event="help noinput nomatch">
   <assign name="helpcounter" expr="helpcounter+1"/>
   <if cond="helpcounter == 1">
    <prompt>
       <break time="150ms"/>Please say the name of a planet.
      <break time="2s"/> Select Mercury, Venus, Earth,
      Mars, Jupiter, Saturn, Uranus, Neptune, or Pluto.
    </prompt>
   </if>
   <if cond="helpcounter == 2">
    <prompt>
       <break time="150ms"/>At any time you can say Help,
      Repeat, Go Back, Start Over, or Exit.  To continue,
      say Mercury, Venus, Earth, Mars, Jupiter, Saturn,
      Uranus, Neptune, or Pluto.
    </prompt>
      <assign name="helpcounter" expr="0"/>
   </if>
  </catch>
  <grammar version="1.0" mode="voice" root="planet">
   <rule id="planet" scope="public">
    <one-of>
     <item>mercury</item>
     <item>venus</item>
     <item>earth</item>
     <item>mars</item>
     <item>jupiter</item>
```

```
     <item>saturn</item>
     <item>uranus</item>
     <item>neptune</item>
     <item>pluto</item>
    </one-of>
   </rule>
  </grammar>
  <filled>
   <assign name="document.planet" expr="planet"/>
   <assign name="document.skipintro" expr="'skip'"/>
   <goto next="#playlandmark"/>
  </filled>
 </field>
</form>

<form id="playlandmark">
 <block>
  <audio src="triple.au"/>
  <prompt>
   <break time="150ms"/>
   <value expr="document.planet"/>
   <break time="150ms"/>
   <audio src="triple.au"/>
  </prompt>
  <goto next="#gettopic"/>
 </block>
</form>

<form id='gettopic'>
 <block>
  <assign name="helpcounter" expr="0"/>
  <assign name="document.goback" expr="'helloworlds'"/>
  <assign name="currentform" expr="'gettopic'"/>
 </block>
 <field name="topic">
  <prompt>
   <break time="150ms"/>Select a topic.  <break time="3s"/>
   Say Distance from Sun, <break time="750ms"/>
   Orbital Period, <break time="750ms"/>
   Temperature, <break time="750ms"/>
   Size, <break time="750ms"/>
   Atmospheric Composition, <break time="750ms"/>
   Number of Moons, <break time="750ms"/>
   or Tell Me Everything.
  </prompt>
  <catch event="help noinput nomatch">
   <assign name="helpcounter" expr="helpcounter+1"/>
   <if cond="helpcounter == 1">
    <prompt>
      <break time="150ms"/>Please say the topic you're
     interested in.  Select Distance from Sun, Orbital
     Period, Temperature, Size, Atmospheric Composition,
      Number of Moons, or Tell Me Everything.
     </prompt>
```

```
   </if>
   <if cond="helpcounter == 2">
    <prompt>
     <break time="150ms"/>At any time you can say Help,
     Repeat, Go Back, Start Over, or Exit.  To continue, say
     Distance from Sun, Orbital Period, Temperature, Size,
     Atmospheric Composition, Number of Moons, or Tell Me
     Everything.
    </prompt>
    <assign name="helpcounter" expr="0"/>
   </if>
  </catch>
  <grammar version="1.0" mode="voice" root="planetproperty">
   <rule id="planetproperty" scope="public">
    <one-of>
     <item>distance from sun<tag>$="distance"</tag></item>
     <item>orbital period<tag>$="orbit"</tag></item>
     <item>temperature</item>
     <item>size</item>
     <item>atmospheric composition<tag>$="atmosphere"</tag></item>
     <item>number of moons<tag>$="moons"</tag></item>
     <item>tell me everything<tag>$="all"</tag></item>
     <item>no<tag>$="goback"</tag></item>
     <item>thats not right<tag>$="goback"</tag></item>
    </one-of>
   </rule>
  </grammar>
  <filled>
   <assign name="document.topic" expr="topic"/>
   <if cond="topic=='goback'">
    <goto next="#helloworlds"/>
   </if>
   <goto next="#lookupdata"/>
  </filled>
 </field>
</form>

<form id='lookupdata'>
 <block>
 <prompt>
 </prompt>
  <if cond="document.planet == 'mercury'">
   <assign name="document.distance" expr="'58 million
   kilometers'"/>
   <assign name="document.position" expr="'the closest
   planet to the sun'"/>
   <assign name="document.orbit" expr="'88 days'"/>
   <assign name="document.temperature" expr="'440 degrees
   Celsius'"/>
   <assign name="document.size" expr="'next to smallest'"/>
   <assign name="document.atmosphere" expr="'98% helium, 2%
   hydrogen'"/>
   <assign name="document.moons" expr="'no moons'"/>
  </if>
```

```
<if cond="document.planet == 'venus'">
 <assign name="document.distance" expr="'108 million
 kilometers'"/>
 <assign name="document.position" expr="'the second planet
 from the sun'"/>
 <assign name="document.orbit" expr="'224 days'"/>
 <assign name="document.temperature" expr="'457 degrees
 Celsius, the hottest in the solar system due to a runaway
 greenhouse effect'"/>
 <assign name="document.size" expr="'sixth largest'"/>
 <assign name="document.atmosphere" expr="'97% carbon
 dioxide, 3% nitrogen'"/>
 <assign name="document.moons" expr="'no moons'"/>
</if>
<if cond="document.planet == 'earth'">
 <assign name="document.distance" expr="'150 million
 kilometers'"/>
 <assign name="document.position" expr="'the third planet
 from the sun'"/>
 <assign name="document.orbit" expr="'365 days'"/>
 <assign name="document.temperature" expr="'15 degrees
 Celsius'"/>
 <assign name="document.size" expr="'fifth largest'"/>
 <assign name="document.atmosphere" expr="'79% nitrogen,
 21% oxygen'"/>
 <assign name="document.moons" expr="'one moon'"/>
</if>
<if cond="document.planet == 'mars'">
 <assign name="document.distance" expr="'228 million
 kilometers'"/>
 <assign name="document.position" expr="'the fourth planet
 from the sun'"/>
 <assign name="document.orbit" expr="'687 days'"/>
 <assign name="document.temperature" expr="'negative 55
 degrees Celsius'"/>
 <assign name="document.size" expr="'seventh largest'"/>
 <assign name="document.atmosphere" expr="'96% carbon
 dioxide, 3% nitrogen, 1% argon'"/>
 <assign name="document.moons" expr="'two moons'"/>
</if>
<if cond="document.planet == 'jupiter'">
 <assign name="document.distance" expr="'778 million
 kilometers'"/>
 <assign name="document.position" expr="'the fifth planet
 from the sun'"/>
 <assign name="document.orbit" expr="'12 years'"/>
 <assign name="document.temperature" expr="'negative 153
 degrees Celsius'"/>
 <assign name="document.size" expr="'largest'"/>
 <assign name="document.atmosphere" expr="'90% hydrogen,
 10% helium'"/>
 <assign name="document.moons" expr="'39 moons'"/>
</if>
<if cond="document.planet == 'saturn'">
```

```
 <assign name="document.distance" expr="'1.4 billion
 kilometers'"/>
 <assign name="document.position" expr="'the sixth planet
 from the sun'"/>
 <assign name="document.orbit" expr="'29 and a half
 years'"/>
 <assign name="document.temperature" expr="'negative 185
 degrees Celsius'"/>
 <assign name="document.size" expr="'next to largest'"/>
 <assign name="document.atmosphere" expr="'75% hydrogen,
 25% helium'"/>
 <assign name="document.moons" expr="'30 moons'"/>
</if>
<if cond="document.planet == 'uranus'">
 <assign name="document.distance" expr="'2.9 billion
 kilometers'"/>
 <assign name="document.position" expr="'the seventh
 planet from the sun'"/>
 <assign name="document.orbit" expr="'84 years'"/>
 <assign name="document.temperature" expr="'negative 215
 degrees Celsius'"/>
 <assign name="document.size" expr="'third largest'"/>
 <assign name="document.atmosphere" expr="'83% hydrogen,
 15% helium, 2% methane'"/>
 <assign name="document.moons" expr="'21 moons'"/>
</if>
<if cond="document.planet == 'neptune'">
 <assign name="document.distance" expr="'4.5 billion
 kilometers'"/>
 <assign name="document.position" expr="'usually the
 eighth planet from the sun'"/>
 <assign name="document.orbit" expr="'165 years'"/>
 <assign name="document.temperature" expr="'negative 225
 degrees Celsius'"/>
 <assign name="document.size" expr="'fourth largest'"/>
 <assign name="document.atmosphere" expr="'85% hydrogen,
 13% helium, 2% methane'"/>
 <assign name="document.moons" expr="'eight moons'"/>
</if>
<if cond="document.planet == 'pluto'">
 <assign name="document.distance" expr="'5.9 billion
 kilometers'"/>
 <assign name="document.position" expr="'usually the
 farthest planet from the sun'"/>
 <assign name="document.orbit" expr="'248 and a half
 years'"/>
 <assign name="document.temperature" expr="'negative 233
 degrees Celsius, only 40 degrees above absolute zero'"/>
 <assign name="document.size" expr="'smallest'"/>
 <assign name="document.atmosphere" expr="'mostly
 nitrogen, with some carbon monoxide and methane'"/>
 <assign name="document.moons" expr="'one moon'"/>
</if>
<goto next="#playit"/>
```

```
 </block>
</form>

<form id='playit'>
 <block>
  <if cond="document.topic == 'distance'">
   At an average distance of
   <value expr="document.distance+', '"/>
   <value expr="document.planet"/> is
   <value expr="document.position+'.'"/>
  </if>
  <if cond="document.topic == 'orbit'">
   It takes <value expr="document.planet"/>
   <value expr="document.orbit"/> to orbit the sun.
  </if>
  <if cond="document.topic == 'temperature'">
   The average surface temperature of
   <value expr="document.planet"/> is
   <value expr="document.temperature+'.'"/>
  </if>
  <if cond="document.topic == 'size'">
   <value expr="document.planet"/> is the
   <value expr="document.size"/> planet.
  </if>
  <if cond="document.topic == 'atmosphere'">
   The atmosphere of <value expr="document.planet"/>
   contains <value expr="document.atmosphere+'.'"/>
  </if>
  <if cond="document.topic == 'moons'">
   <value expr="document.planet"/> has
   <value expr="document.moons+'.'"/>
  </if>
  <if cond="document.topic == 'all'">
   At an average distance of
   <value expr="document.distance+', '"/>
   <value expr="document.planet"/> is
   <value expr="document.position+', '"/>
   taking <value expr="document.orbit"/>
   to complete its orbit.
   It has an average surface temperature of
   <value expr="document.temperature+'.'"/>
   The atmosphere is <value expr="document.atmosphere+'.'"/>
   <value expr="document.planet"/> is the
   <value expr="document.size"/> planet, and has
   <value expr="document.moons+'.'"/>
   <goto next="#helloworlds"/>
  </if>
  <goto next="#whatnext"/>
 </block>
</form>

<form id='whatnext'>
 <block>
  <assign name="helpcounter" expr="0"/>
```

```
 <assign name="document.goback" expr="'gettopic'"/>
 <assign name="currentform" expr="'whatnext'"/>
</block>
<block>
 <prompt>
  <break time="150ms"/>
  Select another topic or another planet.
  <break time="2s"/>
  For more information about <value
  expr="document.planet"/>,
 </prompt>
</block>
<field name="topic2">
 <prompt>
  select Distance from Sun, <break time="750ms"/>
  Orbital Period, <break time="750ms"/>
  Temperature, <break time="750ms"/>
  Size, <break time="750ms"/>
  Atmospheric Composition, <break time="750ms"/>
  Number of Moons, <break time="750ms"/>
  or Tell Me Everything.
 </prompt>
 <catch event="help noinput nomatch">
 <assign name="helpcounter" expr="helpcounter+1"/>
 <if cond="helpcounter == 1">
  <prompt>
   <break time="150ms"/>Please say the topic you're
   interested in.  Select Distance from Sun, Orbital Period,
   Temperature, Size, Atmospheric Composition, Number of
   Moons, or Tell Me Everything.
  </prompt>
 </if>
 <if cond="helpcounter == 2">
  <prompt>
   <break time="150ms"/>At any time you can say Help, Repeat,
   Go Back, Start Over, or Exit.  To continue, say Distance
   from Sun, Orbital Period, Temperature, Size, Atmospheric
   Composition, Number of Moons, or Tell Me Everything.
  </prompt>
  <assign name="helpcounter" expr="0"/>
 </if>
 </catch>
 <grammar version="1.0" mode="voice" root="planetprop2">
  <rule id="planetprop2" scope="public">
   <one-of>
    <item>distance from sun<tag>$="distance"</tag></item>
    <item>orbital period<tag>$="orbit"</tag></item>
    <item>temperature</item>
    <item>size</item>
    <item>atmospheric composition<tag>$="atmosphere"</tag></item>
    <item>number of moons<tag>$="moons"</tag></item>
    <item>tell me everything<tag>$="all"</tag></item>
    <item>repeat</item>
    <item>mercury</item>
```

```
   <item>venus</item>
   <item>earth</item>
   <item>mars</item>
   <item>jupiter</item>
   <item>saturn</item>
   <item>uranus</item>
   <item>neptune</item>
   <item>pluto</item>
   <item>select another topic<tag>$="newtopic"</tag></item>
   <item>select another planet<tag>$="newplanet"</tag></item>
  </one-of>
 </rule>
</grammar>
<filled>
 <if cond="topic2 == 'newplanet'">
  <goto next="#helloworlds"/>
 </if>
 <if cond="topic2 == 'newtopic'">
  <goto nextitem="topic2"/>
 </if>
 <if cond="topic2 == 'mercury'">
  <assign name="document.planet" expr="'mercury'"/>
  <goto next="#playlandmark"/>
 </if>
 <if cond="topic2 == 'venus'">
  <assign name="document.planet" expr="'venus'"/>
  <goto next="#playlandmark"/>
 </if>
 <if cond="topic2 == 'earth'">
  <assign name="document.planet" expr="'earth'"/>
  <goto next="#playlandmark"/>
 </if>
 <if cond="topic2 == 'mars'">
  <assign name="document.planet" expr="'mars'"/>
  <goto next="#playlandmark"/>
 </if>
 <if cond="topic2 == 'jupiter'">
  <assign name="document.planet" expr="'jupiter'"/>
  <goto next="#playlandmark"/>
 </if>
 <if cond="topic2 == 'saturn'">
  <assign name="document.planet" expr="'saturn'"/>
  <goto next="#playlandmark"/>
 </if>
 <if cond="topic2 == 'uranus'">
  <assign name="document.planet" expr="'uranus'"/>
  <goto next="#playlandmark"/>
 </if>
 <if cond="topic2 == 'neptune'">
  <assign name="document.planet" expr="'neptune'"/>
  <goto next="#playlandmark"/>
 </if>
 <if cond="topic2 == 'pluto'">
  <assign name="document.planet" expr="'pluto'"/>
```

```
    <goto next="#playlandmark"/>
   </if>
   <if cond="topic2 == 'repeat'">
    <goto next="#playit"/>
   </if>
   <assign name="document.topic" expr="topic2"/>
   <goto next="#lookupdata"/>
  </filled>
 </field>
</form>

<form id='goback'>
 <block>
  <goto expr="'#'+document.goback"/>
 </block>
</form>

<form id='confirmexit'>
 <field name="exitChoice" type="boolean">
  <prompt>
   <break time="150ms"/>
   Do you want to end this call?
  </prompt>
  <catch event="help noinput nomatch">
  <assign name="helpcounter" expr="helpcounter+1"/>
  <if cond="helpcounter == 1">
   <prompt><break time="150ms"/>Please say Yes, No, or Repeat.</prompt>
  </if>
  <if cond="helpcounter == 2">
   <prompt>
    <break time="150ms"/>At any time you can say Help, Repeat,
    Go Back, Start Over, or Exit.  To end the call, say Yes.
    To return to Hello Worlds, say No.
   </prompt>
   <assign name="helpcounter" expr="0"/>
  </if>
  </catch>
  <filled>
   <if cond="exitChoice">
      <goto next="#exit"/>
      <else/>
      <audio src="triple.au"/>
      <prompt>
       <break time="150ms"/>
       Returning.
      </prompt>
      <goto expr="'#'+document.currentform"/>
   </if>
  </filled>
 </field>
</form>

<form id='exit'>
 <block>
```

```
  <prompt>
   <break time="150ms"/>
   Thanks for calling Hello Worlds.  Goodbye!
  </prompt>
  <exit/>
 </block>
</form>

</vxml>
```

Sample 4b. Using Recorded Speech

The main difference between Hello Worlds Version 4 and Version 4b is the introduction of recorded speech. The complete code for Version 4b appears in the Appendix.

Replacing TTS with Recorded Speech

Replacing static TTS with recorded speech is reasonably easy for a prototype, as long as you're willing to record the audio yourself or you have someone who will do it for you. For a commercially deployed system, you will want to have the audio segments produced professionally, following guidelines such as those available in Balentine and Morgan (2001).

To do the conversion for text initially programmed for production via TTS, you enclose the text in <audio> and </audio> tags, and specify the name (and, if necessary, the required directory path) using the 'src' attribute in the <audio> tag[8], as shown below. Because any large project will have many audio files, it's a good idea to put the audio files in their own folder in the project (the example shows the path to a folder named 'audio').

```
<audio src="audio/intro.au">
 Welcome to Hello Worlds!  Your voice site for information
 about the planets of the solar system.
</audio>
```

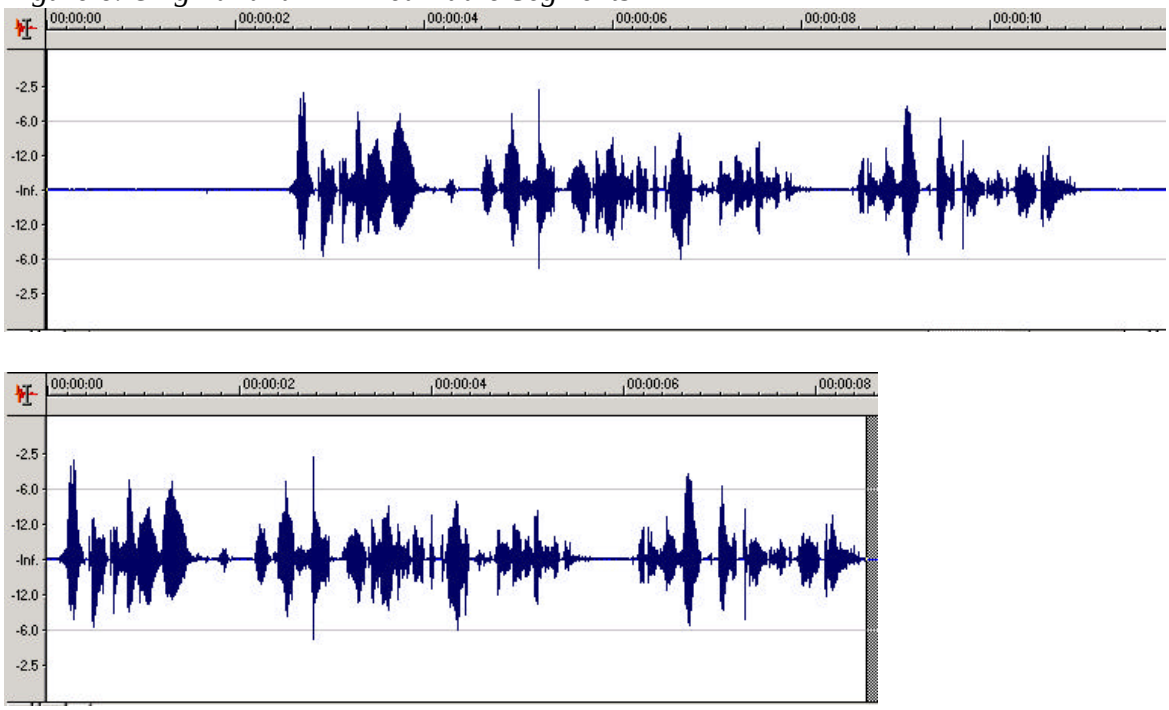
To create the audio file, select File > New > Audio File from the Voice Toolkit menu bar, then use the audio tool that appears to create the initial recording. This ensures that the recording will have the appropriate attributes for the browser to be able to play it. If you are using a separate directory for the audio files, be sure to save it in the audio directory. If the system can't find the specified audio file, it will use play the text with TTS.

Editing Recorded Audio Segments

After making your recordings, you should use an audio editor if you have one to equalize the volume of your recordings, to trim beginning and ending silences, and to take care of any required editing of segment-internal pauses (either lengthening or shortening). Figure 6 shows the waveforms of an original and trimmed audio segment.

[8] The only reason that the name of the audio file starts with 'z-' is to position it at the end of the alphabetically-organized file list in the Navigator panel of the Voice Toolkit. This is nothing more than a convenient way to group the audio files together.

Figure 6. Original and Trimmed Audio Segments



As shown in Figure 6, almost all of the silence has been trimmed (deleted) from the beginning and end of the segment. Unless there is a specific reason to do otherwise, you should trim recorded speech aggressively to avoid unwanted delays in the playing of audio segments (caused by silence at the beginning of the segment) and unwanted extensions of the silence timeout period (caused by silence at the end of a segment).

Dynamic Selection of Audio Segments

In VoiceXML 2.0, you dynamically select an audio segment to play (like the dynamic methods available for controlling the data played by TTS). This feature is available by using 'expr' in place of 'src' inside an <audio> tag, as shown in the fourth line of the following form:

```
<form id="playlandmark">
 <block>
  <audio src="audio/triple.au"/>
    <audio expr="'audio/'+document.planet+'.au'">
     <break time="150ms"/>
     <value expr="document.planet"/>
     <break time="150ms"/>
    </audio>
  <audio src="audio/triple.au"/>
  <goto next="#gettopic"/>
 </block>
</form>
```

Note: If you plan to play two audio segments together to make a larger segment, try to end the first segment at a natural pause point (for example, a period or comma) to help make the concatenation of the segments sound as natural as possible.

Using an Application Root Document

All examples in this report have coded the entire application in one document, with variable and grammar scope set to 'document' level as required. An alternative structure uses multiple documents, with an 'application root document' containing always-active commands and other common code. As long as a set of VoiceXML documents refers to the application root document, it stays loaded, and its variables and grammars, if set to document scope on the application root document, are available for use on any page of the application. References to an application root document appear in the <vxml> element:

```
<vxml version="2.0" application="app-root.vxml">
```

Very complex prototypes might be easier to manage if broken up into several documents that refer to a common application root document.

References

Balentine, B., Morgan, D. M., & Meisel, W. S. (2001). *How to build a speech recognition application: A style guide for telephony dialogs* (2nd ed). San Ramon, CA: Enterprise Integration Group.

Gardner-Bonneau, D. (1999). *Human factors and voice interactive systems.* New York, NY: Kluwer.

International Business Machines Corp. (2003). *VoiceXML programmer's guide (Version 4.2)*. Boca Raton, FL: Author.

Lewis, J. R., Simone, J. E., & Bogacz, M. (2000). *Designing common functions for speech-only user interfaces: Rationales, sample dialogs, potential uses for event counting, and sample grammars* (Tech. Report 29.3287). West Palm Beach, FL: International Business Machines Corp.

Polkosky, M. D., & Lewis, J. R. (2002). Effect of ticking rate on user estimation of system response time. *International Journal of Human-Computer Interaction, 14*, 423-446.

Sadowski, W. J., & Lewis, J. R. (2000a). *Usability evaluation of speech user interfaces for three currency conversion prototypes* (Tech. Report 29.3308). West Palm Beach, FL: International Business Machines Corp.

Sadowski, W. J., & Lewis, J. R. (2000b). *Wizard of Oz usability evaluation of the IBM WebSphere WebVoice demo* (Tech. Report 29.3321). West Palm Beach, FL: International Business Machines Corp.

Sadowski, W. J., & Lewis, J. R. (2001). *Usability evaluation of the IBM WebSphere WebVoice demo* (Tech. Report 29.3387). West Palm Beach, FL: International Business Machines Corp.

Virzi, R. A., & Huitema, J. S. (1997). Telephone based menus: Evidence that broader is better than deeper. In *Proceedings of the Human Factors and Ergonomics Society 41st Annual Meeting* (pp. 315-319). Albuquerque, NM: Human Factors and Ergonomics Society.

Appendix: Hello Worlds Version 4b

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE vxml PUBLIC "-//W3C//DTD VOICEXML 2.0//EN" "vxml20-1115.dtd">
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
<meta name="GENERATOR" content="Voice Toolkit for WebSphere Studio"/>

<var name="skipintro" expr="'play'"/>
<var name="helpcounter" expr="0"/>
<var name="distance" expr="'undefined'"/>
<var name="position" expr="'undefined'"/>
<var name="orbit" expr="'undefined'"/>
<var name="temperature" expr="'undefined'"/>
<var name="size" expr="'undefined'"/>
<var name="atmosphere" expr="'undefined'"/>
<var name="moons" expr="'undefined'"/>
<var name="goback" expr="'undefined'"/>
<var name="currentform" expr="'undefined'"/>

<property name="bargeintype" value="speech"/>

<link next="#helloworlds">
  <grammar version="1.0" mode="voice" root="returntomain">
   <rule id="returntomain" scope="public">
    <one-of>
     <item>main menu</item>
     <item>start over</item>
    </one-of>
   </rule>
  </grammar>
</link>

<link next="#confirmexit">
  <grammar version="1.0" mode="voice" root="goodbye">
   <rule id="goodbye" scope="public">
    <one-of>
     <item>goodbye</item>
     <item>exit</item>
    </one-of>
   </rule>
  </grammar>
</link>

<link next="#goback">
  <grammar version="1.0" mode="voice" root="return">
   <rule id="return" scope="public">
    <one-of>
     <item>go back</item>
    </one-of>
   </rule>
  </grammar>
</link>
```

```
<form id="helloworlds">
 <block name='introduction'>
  <assign name="helpcounter" expr="0"/>
  <assign name="currentform" expr="'helloworlds'"/>
  <if cond="skipintro == 'skip'">
   <goto nextitem="planet"/>
  </if>
  <audio src="audio/intro.au">
   Welcome to Hello Worlds!  Your voice site for information
   about the planets of the solar system.  You can say help
   or repeat at any time.
  </audio>
 </block>
 <field name='planet'>
  <prompt>
   <break time="150ms"/>
   <audio src="audio/main.au">
    Select a planet. <break time="3000ms"/>
    Say Mercury, <break time="750ms"/>
    Venus, <break time="750ms"/>
    Earth, <break time="750ms"/>
    Mars, <break time="750ms"/>
    Jupiter, <break time="750ms"/>
    Saturn, <break time="750ms"/>
    Uranus, <break time="750ms"/>
    Neptune, <break time="750ms"/>
    Pluto, <break time="750ms"/>
    or Exit.
   </audio>
  </prompt>
  <catch event="help noinput nomatch">
   <assign name="helpcounter" expr="helpcounter+1"/>
   <if cond="helpcounter == 1">
    <audio src="audio/mainhelp1.au">
       <break time="150ms"/>Please say the name of a planet.
      <break time="2000ms"/>
       Select Mercury, Venus, Earth, Mars, Jupiter, Saturn,
      Uranus, Neptune, or Pluto.
      </audio>
   </if>
   <if cond="helpcounter == 2">
      <audio src="audio/anytime.au">
       <break time="150ms"/>At any time you can say Help,
      Repeat, Go Back, Start Over, or Exit.
    </audio>
    <audio src="audio/mainhelp2.au">
       To continue, say Mercury, Venus, Earth, Mars, Jupiter,
      Saturn, Uranus, Neptune, or Pluto.
      </audio>
      <assign name="helpcounter" expr="0"/>
   </if>
  </catch>
  <grammar version="1.0" mode="voice" root="planet">
   <rule id="planet" scope="public">
```

```
    <one-of>
     <item>mercury</item>
     <item>venus</item>
     <item>earth</item>
     <item>mars</item>
     <item>jupiter</item>
     <item>saturn</item>
     <item>uranus</item>
     <item>neptune</item>
     <item>pluto</item>
    </one-of>
   </rule>
  </grammar>
  <filled>
   <assign name="document.planet" expr="planet"/>
   <assign name="document.skipintro" expr="'skip'"/>
   <goto next="#playlandmark"/>
  </filled>
 </field>
</form>

<form id="playlandmark">
 <block>
  <audio src="audio/triple.au"/>
    <audio expr="'audio/'+document.planet+'.au'">
     <break time="150ms"/>
     <value expr="document.planet"/>
     <break time="150ms"/>
    </audio>
  <audio src="audio/triple.au"/>
  <goto next="#gettopic"/>
 </block>
</form>

<form id='gettopic'>
 <block>
  <assign name="helpcounter" expr="0"/>
  <assign name="document.goback" expr="'helloworlds'"/>
  <assign name="currentform" expr="'gettopic'"/>
 </block>
 <field name="topic">
  <prompt>
   <break time="150ms"/>
   <audio src="audio/selectatopic.au">
    Select a topic.
   </audio>
   <audio src="audio/list.au">
    <break time="3000ms"/>
    Say Distance from Sun, <break time="750ms"/>
    Orbital Period, <break time="750ms"/>
    Temperature, <break time="750ms"/>
    Size, <break time="750ms"/>
    Atmospheric Composition, <break time="750ms"/>
    Number of Moons, <break time="750ms"/>
```

```
    or Tell Me Everything.
   </audio>
  </prompt>
    <catch event="help noinput nomatch">
   <assign name="helpcounter" expr="helpcounter+1"/>
   <if cond="helpcounter == 1">
    <audio src="audio/topichelp1.au">
       <break time="150ms"/>Please say the topic you're
      interested in.  Select Distance from Sun, Orbital
      Period, Temperature, Size, Atmospheric Composition,
       Number of Moons, or Tell Me Everything.
      </audio>
   </if>
   <if cond="helpcounter == 2">
      <audio src="audio/anytime.au">
       <break time="150ms"/>At any time you can say Help,
      Repeat, Go Back, Start Over, or Exit.
      </audio>
    <audio src="audio/topichelp2.au">
       To continue, say Distance from Sun, Orbital Period,
      Temperature, Size, Atmospheric Composition, Number of
      Moons, or Tell Me Everything.
      </audio>
      <assign name="helpcounter" expr="0"/>
   </if>
  </catch>
  <grammar version="1.0" mode="voice" root="planetproperty">
   <rule id="planetproperty" scope="public">
    <one-of>
     <item>distance from sun<tag>$="distance"</tag></item>
     <item>orbital period<tag>$="orbit"</tag></item>
     <item>temperature</item>
     <item>size</item>
     <item>atmospheric composition<tag>$="atmosphere"</tag></item>
     <item>number of moons<tag>$="moons"</tag></item>
     <item>tell me everything<tag>$="all"</tag></item>
     <item>no<tag>$="goback"</tag></item>
     <item>thats not right<tag>$="goback"</tag></item>
    </one-of>
   </rule>
  </grammar>
  <filled>
   <assign name="document.topic" expr="topic"/>
   <if cond="topic=='goback'">
    <goto next="#helloworlds"/>
   </if>
   <goto next="#lookupdata"/>
  </filled>
 </field>
</form>

<form id='lookupdata'>
 <block>
  <if cond="document.planet == 'mercury'">
```

```
 <assign name="document.distance" expr="'58 million
 kilometers'"/>
 <assign name="document.position" expr="'the closest
 planet to the sun'"/>
 <assign name="document.orbit" expr="'88 days'"/>
 <assign name="document.temperature" expr="'440 degrees
 Celsius'"/>
 <assign name="document.size" expr="'next to smallest'"/>
 <assign name="document.atmosphere" expr="'98% helium, 2%
 hydrogen'"/>
 <assign name="document.moons" expr="'no moons'"/>
</if>
<if cond="document.planet == 'venus'">
 <assign name="document.distance" expr="'108 million
 kilometers'"/>
 <assign name="document.position" expr="'the second planet
 from the sun'"/>
 <assign name="document.orbit" expr="'224 days'"/>
 <assign name="document.temperature" expr="'457 degrees
 Celsius, the hottest in the solar
 system due to a runaway greenhouse effect'"/>
 <assign name="document.size" expr="'sixth largest'"/>
 <assign name="document.atmosphere" expr="'97% carbon
 dioxide, 3% nitrogen'"/>
 <assign name="document.moons" expr="'no moons'"/>
</if>
<if cond="document.planet == 'earth'">
 <assign name="document.distance" expr="'150 million
 kilometers'"/>
 <assign name="document.position" expr="'the third planet
 from the sun'"/>
 <assign name="document.orbit" expr="'365 days'"/>
 <assign name="document.temperature" expr="'15 degrees
 Celsius'"/>
 <assign name="document.size" expr="'fifth largest'"/>
 <assign name="document.atmosphere" expr="'79% nitrogen,
 21% oxygen'"/>
 <assign name="document.moons" expr="'one moon'"/>
</if>
<if cond="document.planet == 'mars'">
 <assign name="document.distance" expr="'228 million
 kilometers'"/>
 <assign name="document.position" expr="'the fourth planet
 from the sun'"/>
 <assign name="document.orbit" expr="'687 days'"/>
 <assign name="document.temperature" expr="'negative 55
 degrees Celsius'"/>
 <assign name="document.size" expr="'seventh largest'"/>
 <assign name="document.atmosphere" expr="'96% carbon
 dioxide, 3% nitrogen, 1% argon'"/>
 <assign name="document.moons" expr="'two moons'"/>
</if>
<if cond="document.planet == 'jupiter'">
 <assign name="document.distance" expr="'778 million
```

```
 kilometers'"/>
 <assign name="document.position" expr="'the fifth planet
 from the sun'"/>
 <assign name="document.orbit" expr="'12 years'"/>
 <assign name="document.temperature" expr="'negative 153
 degrees Celsius'"/>
 <assign name="document.size" expr="'largest'"/>
 <assign name="document.atmosphere" expr="'90% hydrogen,
 10% helium'"/>
 <assign name="document.moons" expr="'39 moons'"/>
</if>
<if cond="document.planet == 'saturn'">
 <assign name="document.distance" expr="'1.4 billion
 kilometers'"/>
 <assign name="document.position" expr="'the sixth planet
 from the sun'"/>
 <assign name="document.orbit" expr="'29 and a half
 years'"/>
 <assign name="document.temperature" expr="'negative 185
 degrees Celsius'"/>
 <assign name="document.size" expr="'next to largest'"/>
 <assign name="document.atmosphere" expr="'75% hydrogen,
 25% helium'"/>
 <assign name="document.moons" expr="'30 moons'"/>
</if>
<if cond="document.planet == 'uranus'">
 <assign name="document.distance" expr="'2.9 billion
 kilometers'"/>
 <assign name="document.position" expr="'the seventh
 planet from the sun'"/>
 <assign name="document.orbit" expr="'84 years'"/>
 <assign name="document.temperature" expr="'negative 215
 degrees Celsius'"/>
 <assign name="document.size" expr="'third largest'"/>
 <assign name="document.atmosphere" expr="'83% hydrogen,
 15% helium, 2% methane'"/>
 <assign name="document.moons" expr="'21 moons'"/>
</if>
<if cond="document.planet == 'neptune'">
 <assign name="document.distance" expr="'4.5 billion
 kilometers'"/>
 <assign name="document.position" expr="'usually the
 eighth planet from the sun'"/>
 <assign name="document.orbit" expr="'165 years'"/>
 <assign name="document.temperature" expr="'negative 225
 degrees Celsius'"/>
 <assign name="document.size" expr="'fourth largest'"/>
 <assign name="document.atmosphere" expr="'85% hydrogen,
 13% helium, 2% methane'"/>
 <assign name="document.moons" expr="'eight moons'"/>
</if>
<if cond="document.planet == 'pluto'">
 <assign name="document.distance" expr="'5.9 billion
 kilometers'"/>
```

```
   <assign name="document.position" expr="'usually the
   farthest planet from the sun'"/>
   <assign name="document.orbit" expr="'248 and a half
   years'"/>
   <assign name="document.temperature" expr="'negative 233
   degrees Celsius, only 40
   degrees above absolute zero'"/>
   <assign name="document.size" expr="'smallest'"/>
   <assign name="document.atmosphere" expr="'mostly
   nitrogen, with some carbon monoxide and methane'"/>
   <assign name="document.moons" expr="'one moon'"/>
  </if>
  <goto next="#playit"/>
 </block>
</form>

<form id='playit'>
 <block>
  <if cond="document.topic == 'distance'">
   At an average distance of
   <value expr="document.distance+', '"/>
   <value expr="document.planet"/> is
   <value expr="document.position+'.'"/>
  </if>
  <if cond="document.topic == 'orbit'">
   It takes <value expr="document.planet"/>
   <value expr="document.orbit"/> to orbit the sun.
  </if>
  <if cond="document.topic == 'temperature'">
   The average surface temperature of
   <value expr="document.planet"/> is
   <value expr="document.temperature+'.'"/>
  </if>
  <if cond="document.topic == 'size'">
   <value expr="document.planet"/> is the
   <value expr="document.size"/> planet.
  </if>
  <if cond="document.topic == 'atmosphere'">
   The atmosphere of <value expr="document.planet"/>
   contains <value expr="document.atmosphere+'.'"/>
  </if>
  <if cond="document.topic == 'moons'">
   <value expr="document.planet"/> has
   <value expr="document.moons+'.'"/>
  </if>
  <if cond="document.topic == 'all'">
   At an average distance of
   <value expr="document.distance+', '"/>
   <value expr="document.planet"/> is
   <value expr="document.position+', '"/>
   taking <value expr="document.orbit"/>
   to complete its orbit.
   It has an average surface temperature of
   <value expr="document.temperature+'.'"/>
```

```
   The atmosphere is <value expr="document.atmosphere+'.'"/>
   <value expr="document.planet"/> is the
   <value expr="document.size"/> planet, and has
   <value expr="document.moons+'.'"/>
   <goto next="#helloworlds"/>
  </if>
  <goto next="#whatnext"/>
 </block>
</form>

<form id='whatnext'>
 <block>
  <assign name="helpcounter" expr="0"/>
  <assign name="document.goback" expr="'helloworlds'"/>
  <assign name="currentform" expr="'whatnext'"/>
 </block>
 <block>
   <audio src="audio/newplanet.au">
    <break time="150ms"/>Select another topic or another planet.
   </audio>
   <audio src="audio/pause2000.au">
    <break time="2000ms"/>
   </audio>
   <if cond="document.planet=='mercury'">
    <audio src="audio/moremercury.au">
     For more information about <value
     expr="document.planet"/>,
    </audio>
   </if>
    <if cond="document.planet=='venus'">
    <audio src="audio/morevenus.au">
     For more information about <value
     expr="document.planet"/>,
    </audio>
   </if>
   <if cond="document.planet=='earth'">
    <audio src="audio/moreearth.au">
     For more information about <value
     expr="document.planet"/>,
    </audio>
   </if>
   <if cond="document.planet=='mars'">
    <audio src="audio/moremars.au">
     For more information about <value
     expr="document.planet"/>,
    </audio>
   </if>
   <if cond="document.planet=='jupiter'">
    <audio src="audio/morejupiter.au">
     For more information about <value
     expr="document.planet"/>,
    </audio>
   </if>
   <if cond="document.planet=='saturn'">
```

```
   <audio src="audio/moresaturn.au">
    For more information about <value
    expr="document.planet"/>,
   </audio>
  </if>
  <if cond="document.planet=='uranus'">
   <audio src="audio/moreuranus.au">
    For more information about <value
    expr="document.planet"/>,
   </audio>
  </if>
  <if cond="document.planet=='neptune'">
   <audio src="audio/moreneptune.au">
    For more information about <value
    expr="document.planet"/>,
   </audio>
  </if>
  <if cond="document.planet=='pluto'">
   <audio src="audio/morepluto.au">
    For more information about <value
    expr="document.planet"/>,
   </audio>
  </if>
</block>
<field name="topic2">
 <prompt>
  <audio src="audio/list.au">
   select Distance from Sun, <break time="750ms"/> Orbital
   Period, <break time="750ms"/> Temperature,
   <break time="750ms"/> Size, <break time="750ms"/>
   Atmospheric Composition, <break time="750ms"/> Number of
   Moons, <break time="750ms"/> or Tell Me Everything.
  </audio>
 </prompt>
 <catch event="help noinput nomatch">
 <assign name="helpcounter" expr="helpcounter+1"/>
 <if cond="helpcounter == 1">
  <audio src="audio/topichelp1.au">
   <break time="150ms"/>Please say the topic you're
   interested in.  Select Distance from Sun, Orbital
   Period, Temperature, Size, Atmospheric Composition,
   Number of Moons, or Tell Me Everything.
  </audio>
 </if>
 <if cond="helpcounter == 2">
  <audio src="audio/anytime.au">
   <break time="150ms"/>At any time you can say Help,
   Repeat, Go Back, Start Over, or Exit.
  </audio>
  <audio src="audio/topichelp2.au">
   To continue, say Distance from Sun, Orbital Period,
   Temperature, Size, Atmospheric Composition, Number of
   Moons, or Tell Me Everything.
  </audio>
```

```
 <assign name="helpcounter" expr="0"/>
</if>
</catch>
<grammar version="1.0" mode="voice" root="planetprop2">
 <rule id="planetprop2" scope="public">
  <one-of>
   <item>distance from sun<tag>$="distance"</tag></item>
   <item>orbital period<tag>$="orbit"</tag></item>
   <item>temperature</item>
   <item>size</item>
   <item>atmospheric composition<tag>$="atmosphere"</tag></item>
   <item>number of moons<tag>$="moons"</tag></item>
   <item>tell me everything<tag>$="all"</tag></item>
   <item>repeat</item>
   <item>mercury</item>
   <item>venus</item>
   <item>earth</item>
   <item>mars</item>
   <item>jupiter</item>
   <item>saturn</item>
   <item>uranus</item>
   <item>neptune</item>
   <item>pluto</item>
   <item>select another topic<tag>$="newtopic"</tag></item>
   <item>select another planet<tag>$="newplanet"</tag></item>
  </one-of>
 </rule>
</grammar>
<filled>
 <if cond="topic2 == 'newplanet'">
  <goto next="#helloworlds"/>
 </if>
 <if cond="topic2 == 'newtopic'">
  <goto nextitem="topic2"/>
 </if>
 <if cond="topic2 == 'mercury'">
  <assign name="document.planet" expr="'mercury'"/>
  <goto next="#playlandmark"/>
 </if>
 <if cond="topic2 == 'venus'">
  <assign name="document.planet" expr="'venus'"/>
  <goto next="#playlandmark"/>
 </if>
 <if cond="topic2 == 'earth'">
  <assign name="document.planet" expr="'earth'"/>
  <goto next="#playlandmark"/>
 </if>
 <if cond="topic2 == 'mars'">
  <assign name="document.planet" expr="'mars'"/>
  <goto next="#playlandmark"/>
 </if>
 <if cond="topic2 == 'jupiter'">
  <assign name="document.planet" expr="'jupiter'"/>
  <goto next="#playlandmark"/>
```

```
   </if>
   <if cond="topic2 == 'saturn'">
    <assign name="document.planet" expr="'saturn'"/>
    <goto next="#playlandmark"/>
   </if>
   <if cond="topic2 == 'uranus'">
    <assign name="document.planet" expr="'uranus'"/>
    <goto next="#playlandmark"/>
   </if>
   <if cond="topic2 == 'neptune'">
    <assign name="document.planet" expr="'neptune'"/>
    <goto next="#playlandmark"/>
   </if>
   <if cond="topic2 == 'pluto'">
    <assign name="document.planet" expr="'pluto'"/>
    <goto next="#playlandmark"/>
   </if>
   <if cond="topic2 == 'repeat'">
    <goto next="#playit"/>
   </if>
   <assign name="document.topic" expr="topic2"/>
   <goto next="#lookupdata"/>
  </filled>
 </field>
</form>

<form id='goback'>
 <block>
  <goto expr="'#'+document.goback"/>
 </block>
</form>

<form id='confirmexit'>
 <field name="exitChoice" type="boolean">
  <prompt>
   <break time="150ms"/>
   <audio src="audio/end.au">
    Do you want to end this call?
   </audio>
  </prompt>
  <catch event="help noinput nomatch">
  <assign name="helpcounter" expr="helpcounter+1"/>
  <if cond="helpcounter == 1">
   <audio src="audio/yesnorepeat.au">
    <break time="150ms"/>Please say Yes, No, or Repeat.
   </audio>
  </if>
  <if cond="helpcounter == 2">
   <audio src="audio/anytime.au">
    <break time="150ms"/>At any time you can say Help,
    Repeat, Go Back, Start Over, or Exit.
   </audio>
   <audio src="audio/exithelp2.au">
    To end the call, say Yes.  To return to Hello Worlds,
```

```
    say No.
   </audio>
   <assign name="helpcounter" expr="0"/>
  </if>
  </catch>
  <filled>
   <if cond="exitChoice">
      <goto next="#exit"/>
      <else/>
      <audio src="triple.au"/>
      <audio src="audio/returning.au">
       <break time="150ms"/>Returning.
      </audio>
      <goto expr="'#'+document.currentform"/>
   </if>
  </filled>
 </field>
</form>

<form id='exit'>
 <block>
  <audio src="audio/bye.au">
   <break time="150ms"/> Thanks for calling Hello Worlds.  Goodbye!
  </audio>
  <exit/>
 </block>
</form>

</vxml>
```