Special Lecture (406)
Spoken Language Dialog Systems
Introduction to VoiceXML

Rolf Schwitter

schwitt@ics.mq.edu.au
Today’s Program

• Developing speech interfaces
• Brief history of VoiceXML
• VoiceXML
• GUI versus VUI
• VoiceXML implementations
• Grammars
Developing Speech Interfaces

- Speech interfaces can be developed using
  - general-purpose programming languages
  - special-purpose (programming) languages.
- A special-purpose language such as VoiceXML can
  - simplify application development
  - reduce network traffic
  - separate interaction code from application logic code
  - provide portability and simplicity
  - support prototyping and refinement.
Brief History of VoiceXML

- In 1999, AT&T, IBM, Lucent Technology and Motorola formed the VoiceXML Forum.
- The goal was to establish and promote VoiceXML for making Internet content available by phone and voice.
- Each company had previously developed its own markup language.
- Customers were reluctant to invest in proprietary technology.
- VoiceXML 1.0 was released in March 2000.
- VoiceXML 2.0 is a candidate recommendation (March 2004).
W3C Voice Browser Working Group

- Organisations participating in the Voice Browser Working Group:
  
  BeVocal, Canon, Comverse, France Telecom, Genesys, HeyAnita, Hitachi, HP, IBM, Intel, IWA/HWG, Loquendo, Microsoft, MITRE, Mitsubishi Electric, Motorola, Nokia, Nortel Networks, Nuance, PipeBeach, SAP, Scansoft, Snowshore Networks, SpeechWorks, Sun Microsystems, Syntellect, Tellme Networks, Unisys, Verascape, Vocalocity, VoiceGenie, Voxeo, and Voxpilot
<?xml version="1.0"?>
<vxml version="2.0">
  <form>
    <block>
      <prompt bargein="false">Welcome to Ajax Travel.
      <audio src="http://www.prerecorded.audiofile...."/>
      </prompt>
    </block>
  </form>
</vxml>
VoiceXML

- VoiceXML
  - is designed to describe the speech user interface
  - reduces the amount of speech expertise (but not design expertise).

- VoiceXML documents can
  - be static or dynamically generated by server side code
  - use the same business logic and databases as the visual Web.

- Note: The form interpretation algorithm of the voice browser drives the interaction between the VoiceXML documents and the user.
Missing Features

• VoiceXML 2.0 does not support
  – learn-to-speak applications
  – speaker identification and verification.
VoiceXML Architecture

- regular phone
- wireless phone
- soft phone

- telephony interface
- voice browser
- automated speech recognition
- text-to-speech synthesis
- touchtone
- audio play/record

- VoiceXML documents
- audio files
- service logic (CGI)
- transaction processing
- database interface
A VoiceXML Scenario

- A customer dials the phone number of a travel agent.
- The VoiceXML gateway receives the call along with information about the dialed number.
- The VoiceXML gateway searches a database.
- If successful, it maps the dialed number to a URL.
- This URL is the location of the agent’s main page (ajax.vxml).
- The gateway retrieves the ajax.vxml page together with associated files such as grammars and recorded audio from the HTTP server.
- These associated files may be cached on the VoiceXML gateway.
A VoiceXML Scenario

- The VoiceXML interpreter parses and executes the VoiceXML document.
- The interpreter steps through `ajax.vxml` playing prompts, hearing responses and passing them on to a speech recognition engine.
- If necessary, additional VoiceXML documents and associated files are retrieved from the HTTP server.
- Recorded audio is served by specifying the URL of the WAV file.
- Communications between the voice gateway and the HTTP server follow standard HTTP protocols.
More On VoiceXML: An Example Dialog

Computer: Welcome to Ajax Travel.
Computer: Please say your name.
Caller: Sam.
Computer: Do you want to travel by air, rail, or boat?
Caller: Rail.
Computer: You have selected to travel by rail.

...
A VoiceXML Code Fragment

```xml
<?xml version = "1.0"?>
<vxml version = "2.0">
<form>

  <block>
    <prompt>
      Welcome to Ajax Travel.
    </prompt>
  </block>

  <field name = "UserName">
    <prompt>
      Please say your name.
    </prompt>
  </field>

  <field name = "TravelMode">
    <prompt>
      Do you want to travel by air, rail, or boat?
    </prompt>
  </field>

  <field name = "PreferredTravel">
    <prompt>
      You have selected to travel by rail.
    </prompt>
  </field>

</form>
```
A VoiceXML Code Fragment

```xml
<grammar type = "application/srgs+xml" version = "1.0">
  <rule id = "aUser">
    <one-of>
      <item>fred</item>
      <item>sam</item>
    </one-of>
  </rule>
</grammar>
```

Computer: Welcome to Ajax Travel.
Computer: Please say your name.
Caller: Sam.
Computer: Do you want to travel by air, rail, or boat?
Caller: Rail.
Computer: You have selected to travel by rail.
A VoiceXML Code Fragment

```xml
<field>
  <filled>
    <goto next = "#travel"/>
  </filled>
</field>
```

Computer: Welcome to Ajax Travel.
Caller: Sam.
Computer: Please say your name.
Caller: Rail.
Computer: Do you want to travel by air, rail, or boat?
Computer: You have selected to travel by rail.

<!–- ... transition to another dialog in the current document -->
A VoiceXML Code Fragment

<menu id = "travel">
<prompt>
    Do you want to travel by air, rail, or boat?
</prompt>
</menu>

Computer: Welcome to Ajax Travel.
Computer: Please say your name.
Caller: Sam.
Computer: Do you want to travel by air, rail, or boat?
Caller: Rail.
Computer: You have selected to travel by rail.

<!-- choices follow on the next slides -->
A VoiceXML Code Fragment

<choice next = "#plane">
  <grammar type = "application/srgs+xml" version = "1.0">
    <rule id = "by_plane">
      <item> air </item>
    </rule>
  </grammar>
</choice>

| Computer: Welcome to Ajax Travel. |
| Computer: Please say your name.   |
| Caller: Sam.                      |
| Computer: Do you want to travel by air, rail, or boat? |
| Caller: Rail.                     |
| Computer: You have selected to travel by rail. |
A VoiceXML Code Fragment

```xml
<choice next = "#train">
  <grammar type = "application/srgs+xml" version = "1.0">
    <rule id = "by_train">
      <item> rail </item>
    </item>
  </rule>
</grammar>
</choice>
```

Computer: Welcome to Ajax Travel.
Computer: Please say your name.
Caller: Sam.
Computer: Do you want to travel by air, rail, or boat?
Caller: Rail.
Computer: You have selected to travel by rail.
A VoiceXML Code Fragment

```xml
<choice next="#boat">
  <grammar type="application/srgs+xml" version="1.0">
    <rule id="by_boat">
      <item> boat </item>
    </rule>
  </grammar>
</choice>
```

Computer: Welcome to Ajax Travel.
Computer: Please say your name.
Caller: Sam.
Computer: Do you want to travel by air, rail, or boat?
Caller: Rail.
Computer: You have selected to travel by rail.
A VoiceXML Code Fragment

```xml
<form id = "train">
  <block>
    <prompt>
      You have selected to travel by rail.
      Details for making travel arrangement would be here
      in a real application
    </prompt>
  </block>
</form>
```

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer: Welcome to Ajax Travel.</td>
<td></td>
</tr>
<tr>
<td>Computer: Please say your name.</td>
<td></td>
</tr>
<tr>
<td>Caller: Sam.</td>
<td></td>
</tr>
<tr>
<td>Computer: Do you want to travel by air, rail, or boat?</td>
<td></td>
</tr>
<tr>
<td>Caller: Rail.</td>
<td></td>
</tr>
<tr>
<td>Computer: You have selected to travel by rail.</td>
<td></td>
</tr>
</tbody>
</table>
VoiceXML Elements

- `<vxml>`: top-level element in each VoiceXML document
- `<form>`: a dialog for presenting information and collecting data
- `<block>`: a container of (non-interactive) executable code
- `<prompt>`: queue speech synthesis and audio output to the user
- `<field>`: declares an input field in a form
- `<filled>`: an action executed when fields are filled
VoiceXML Elements

- `<menu>` a dialog for choosing amongst alternative destinations
- `<choice>` define a menu item
- `<grammar>` specify a speech recognition or DTMF grammar
- `<goto>` go to another dialog in the same or different document
GUI versus VUI

Welcome To Ajax Travel

Please complete the following:

Name: [Textbox]
Telephone: [Textbox]

How would you like to travel?
- [ ] Airplane
- [ ] Train
- [ ] Boat

[Submit] [Clear]
# GUI versus VUI

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>GUI (HTML)</th>
<th>VUI (VoiceXML)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fonts</td>
<td>Choice of multiple sizes, colours, and types</td>
<td>Choice of multiple voices</td>
</tr>
<tr>
<td>Pictures</td>
<td>Present the picture to the user</td>
<td>Describe the picture with words (caption)</td>
</tr>
<tr>
<td>Backgrounds</td>
<td>Choice of multiple colours and patterns</td>
<td>Choice of background music or sounds</td>
</tr>
<tr>
<td>Menus</td>
<td>Large number of choices</td>
<td>Usually 7 ± 2 choices</td>
</tr>
<tr>
<td>Forms</td>
<td>Users may enter values in any order</td>
<td>Users enter values usually in predefined order</td>
</tr>
<tr>
<td>Prompting the user</td>
<td>Large menu of information and links</td>
<td>Limited number of prompts and options</td>
</tr>
<tr>
<td>User’s response</td>
<td>Click a field and enter a value</td>
<td>Speak the menu choice or field value</td>
</tr>
</tbody>
</table>
## GUI versus VUI

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>GUI (HTML)</th>
<th>VUI (VoiceXML)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Navigation</td>
<td>Click the link or use the keyboard to enter URL</td>
<td>Speak the option choice</td>
</tr>
<tr>
<td>Human memory</td>
<td>Screen acts as an extension to human memory</td>
<td>Caller is forced to use own mental memory</td>
</tr>
<tr>
<td>Dialog style</td>
<td>Mostly user-directed</td>
<td>Mostly application-directed</td>
</tr>
<tr>
<td>Input control</td>
<td>Apply the type constraints</td>
<td>Apply the grammar rules</td>
</tr>
<tr>
<td>Accuracy of input</td>
<td>Input is always recognized</td>
<td>Input may not be recognised correctly</td>
</tr>
<tr>
<td>Event handling</td>
<td>Display the event message</td>
<td>Present help as a prompt</td>
</tr>
<tr>
<td>Global commands</td>
<td>Available as a menu in a separate frame</td>
<td>Announced at the beginning of a session</td>
</tr>
<tr>
<td>Dates</td>
<td>Absolute dates are used (28.7.2003)</td>
<td>Relative dates are used (“next Monday”)</td>
</tr>
</tbody>
</table>
VoiceXML Implementations

- Web-based VoiceXML development tools:
  - Tellme at http://studio.tellme.com
  - BeVocal at http://café.bevocal.com
  - HeyAnita at http://www.heyanita.com
- VoiceXML platforms (and graphical development tools):
  - Nuance at http://www.nuance.com
  - OptimTalk at http://www.optimtalk.cz
  - Motorola’s Mobile ADK for Voice (old beta version only)
Tellme Studio

• Tellme studio is a suite of Web-based VoiceXML development tools.
• Tellme studio enables you
  – to build and test, and publish VoiceXML applications
  – without buying or installing any hardware or software.
• By registering, you can develop your application for free.
• But check out first the VoiceXML elements supported by the Tellme voice interpreter.
MyStudio

- VoiceXML scratchpad
  - You can write a phone application using the VoiceXML scratchpad.
- Application URL
  - Alternatively, you can write a phone application using a text editor and store the result on a Web server.
  - The application URL points to the initial VoiceXML document.
- VoiceXML terminal
  - You can test the application logic and flow using the VoiceXML terminal.
MyStudio

You're signed in as 28759.

Home
MyStudio

Documentation
VoiceXML 2.0 Elements
VoiceXML Tutorials
Advanced Topics
Grammars
Advanced Network Services

Tools
Scratchpad
Debug Log
Syntax Checker
Grammar Tools
Voice/XML Terminal
Record by Phone

Resources
Code Library
Audio Library
Utilities

MyStudio 2.0 (It's here.)

Application URL
Scratchpad

VoiceXML Scratchpad name: My Scratchpad

<?xml version = "1.0"?>
<?xml version = "2.0"?>
<form>
  <block>
    <prompt>
      Welcome to Ajax Travel
    </prompt>
  </block>
  <field name = "UserName">
    <prompt>
      Say your user name
    </prompt>
  </field>
  <grammar type = "application/srgs+xml"
    version = "1.0">

© Macquarie University 2004
VoiceXML Scratchpad

Type some VoiceXML below, and call 1-800-555-VXML to preview it. [International | Yelp]

✅ Your change was successful. The syntax checker was run. (No errors detected in your VoiceXML.)

**VoiceXML Scratchpad name:** My Scratchpad

```
<?xml version = "1.0"?>
<vxml version = "2.0">
<form>
  <block>
    <prompt>
      Welcome to Ajax Travel
    </prompt>
  </block>
  <field name = "UserName">
    <prompt>
      Say your user name
    </prompt>
    <grammar type = "application/srgs+xml"
      version = "1.0">
      <rule id = "aUser" scope = "public">
```

© Macquarie University 2004
Application URL

Enter the URL to your VoiceXML below, and call 1-800-555-VXML to preview it. [International | VoIP]

Application URL

http://www.ics.mq.edu.au/~rolis/ajax.vxml

Update
VoiceXML Terminal

*** Welcome to VoiceXML Terminal ***

- VoiceXML Terminal lets you test your voice application phone-free!
- Specify the application you want to test in your Application URL or Scratchpad, and then click start. Simulate user input by entering a valid return value from an active grammar. Click ? for more information.

- <audio> Welcome to Ajax Travel </audio>
- <audio> Say your user name </audio>
- Sam
- <audio> Do you want to travel by air, rail, or boat? </audio>

Input: Air

You now use grammar phrases as input.
Grammar Scratchpad

• The Tellme platform provides two choices when writing grammars:
  – use a built-in grammar
  – define your own grammar
• Supported grammar languages are:
  – Nuance Grammar Specification Language (GSL)
  – Speech Recognition Grammar Specification (SRGS)
• You can execute GSL + SRGS in the VoiceXML scratchpad.
• But the Tellme grammar tools support GSL grammars only.
Grammar Scratchpad: SRGS

Some errors were found in your grammar:

Illegal token "<" found at line 1

```
<grammar type = "application/srgs+xml" version = "1.0">
  <rule id = "aUser" scope = "public">
    <one-of>
      <item>fred</item>
      <item>Sam</item>
    </one-of>
  </rule>
</grammar>
```
Grammar Scratchpad: GSL

Your grammar has successfully compiled.

Enter a grammar below, and click "Check Grammar" to verify it:

```xml
<grammar type="application/x-gsl" mode="voice">
  <![CDATA[
    [sam] {<name "samuel"/>}
    [fred] {<name "frederick"/>}
  ]]><grammar>
```
Grammar Phrase Checker

This tool allows you to test phrases against your grammar to determine if they will be recognized and, if so, display the returned value.

The last grammar checked with either the Scratchpad or Grammar URL will be used.

Enter the phrase you'd like to check in your grammar:

Did You Know?
Did you know that you can adjust the size of your Scratchpad in Edit My Preferences?
Grammar Phrase Checker: Returned Value

**Grammar Phrase Checker**

This tool allows you to test phrases against your grammar to determine if they will be recognized and, if so, display the returned value.

The last grammar checked with either the Scratchpad or Grammar URL will be used.

Enter the phrase you'd like to check in your grammar:

Input:

```
sam
```

**Results**

```
1. <name samuel>
```

**Did You Know?**

Did you know that you can adjust the size of your Scratchpad in [Edit My Preferences]?
Grammar Phrase Generator

This tool displays phrases your grammar is capable of recognizing. (You might be surprised!) You can view all phrases your grammar can recognize, or just generate a random sampling.

The last grammar checked with either the Scratchpad or Grammar URL will be used.

- Exhaustive (up to 100 phrases)
- Random sample of $\left[ 10 \right]$ (up to 100)
Grammar Phrase Generator

This tool displays phrases your grammar is capable of recognizing. (You might be surprised!) You can view all phrases your grammar can recognize, or just generate a random sampling.

The last grammar checked with either the Scratchpad or Grammar URL will be used.

- Exhaustive (up to 100 phrases)
- Random sample of: 100 (up to 100)

<table>
<thead>
<tr>
<th>Generated phrase</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. sam</td>
<td>{&lt;name samuel&gt;}</td>
</tr>
<tr>
<td>2. fred</td>
<td>{&lt;name frederick&gt;}</td>
</tr>
</tbody>
</table>
Connecting to Tellme Studio

• To preview your application, you can use a phone and call
  – (408)-678-4465

or you can use a soft phone and call
  – sip:8005558965@sip.studio.tellme.com
Nuance Grammar Specification Language (GSL)

- Key to understand speech are well-designed grammars.
- A grammar describes
  - the words and phrases
  - that the recogniser can understand
  - at a specific point in a VoiceXML document.
- GSL provides a rich set of functionality for writing grammars.
- But GSL syntax uses characters that are reserved by XML.
- Therefore, in-line grammars must be protected (CDATA section).
Here is an in-line grammar in GSL format:

```xml
<grammar type = "application/x-gsl" mode = "voice">
  <![CDATA[
    [ [(new york) (big apple)] {<destination "new york">} 
      [washington (the capital)] {<destination "washington">} 
    ]]]>
</grammar>
```
GSL Syntax by Example

- All words within a GSL grammar are lowercase\(^1\).
- Square brackets define an "or" condition.
- Parentheses define an "and" condition.

\[
[ \\
  [(new york) (big apple)] \text{ } \text{ } \texttt{<destination "new york">} \\
  [washington (the capital)] \text{ } \text{ } \texttt{<destination "washington">} \\
]\]

\(^1\) Uppercase letters are reserved to reference subgrammars.
GSL Syntax by Example

• This grammar fragment

```
[ [(new york) (big apple)] {<destination "new york">}  
  [washington (the capital)] {<destination "washington">} 
]
```

recognises
  – either "New York" or the synonym "Big Apple"
  – or "Washington" or the synonym "The Capital"
as destination.
GSL Syntax by Example

- The value of the name attribute of a field in an active form is set to the value returned by the grammar.

```xml
<form>
  <field name = "destination">
    <prompt>Do you want to fly to New York or Washington?</prompt>
    <grammar type = "application/x-gsl" mode = "voice">
      <![CDATA[
        [[(new york) (big apple)] {<destination "new york"/>}
          [washington (the capital)] {<destination "washington"/>}]
      ]]>  
    </grammar>
  </field>
...
What Happens if . . .

... the caller does not respond with an predefined expression or
... the caller does not respond at all?

• In these cases events are thrown by the platform.

• Events are caught by the catch element:

  <catch event = "nomatch noinput">
    <reprompt/>
  </catch>
<grammar type = "application/x-gsl" mode = "voice">
  <![CDATA[
    [(new york) (big apple)] {<destination "new york"} ]
    [washington (the capital)] {<destination "washington"}]
  ]>
</grammar>

<catch event = "nomatch noinput">
  <reprompt/>
</catch>
GSL Grammar at Work

<prompt>You said <value expr = "destination"/></prompt>
</filled>
</field>
</form>
</vxml>
More on GSL Syntax

- The following grammar fragment

  \[ ([\text{enroll add} \ me]) \ (\text{sign me up}) ]

recognises the three sentences:

- enroll me
- add me
- sign me up

because an "or" condition is nested within an "and" condition.
More on GSL Syntax

• Variations in input are supported through additional operators.
• Operators are attached as prefixes to individual words or phrases.
• Operators can be used to indicate that a word or phrase may occur
  – zero or one time (?)
  – zero or more times (*)
  – one or more times (+)
Using Operators

• Use "?" before an expression when it is completely optional.
• In the following example, the word "me" is optional:

  [(enroll ?me) (sign ?me up)]

• The utterance should not be rejected because the caller neglects to say the word "me".
Using Operators

• Use "*" before an expression when it is optional, but the caller may say it multiple times.

• In the following example, the word "please" is optional
  
  
  [(*please sign ?me up)]

  but the caller may say this word more than once.
Using Operators

• Use "+" before an expression when it must occur at least one time but may occur more than once.
• This operator is provided for completeness only.
• It is not typically used.
GSL Grammar for Voice Input

<grammar type="application/x-gsl" mode="voice">
<![[CDATA[
[ 
 [sales] {<dept "010">} 
 [marketing] {<dept "020">} 
 [engineering] {<dept "030">} 
 [(public relations) (p r)] {<dept "040">} 
 ] 
 ]]> 
</grammar>
GSL Grammar for Touchtone Input

<grammar type="application/x-gsl" mode="dtmf">
  <! [CDATA[
    [
      [dtmf-1] {<dept "010"}]
      [dtmf-2] {<dept "020"}]
      [dtmf-3] {<dept "030"}]
      [dtmf-4] {<dept "040"}]
  ]]> 
</grammar>
Take-Home Message

• You can develop a VoiceXML application using
  – a Web-based development environment
  – a VoiceXML platform on a desktop computer.

• Tellme Studio
  – is a suite of Web-based VoiceXML development tools
  – enables you to build, test and publish Voice XML applications
  – supports the Nuance Grammar Specification Language (GSL).