COMP349
Spoken Language Dialog Systems
Introduction to VoiceXML

Rolf Schwitter
schwitt@ics.mq.edu.au
Today’s Program

- Developing Speech User Interfaces
- VoiceXML
- VoiceXML Architecture
- Example Dialogs
- VoiceXML Code Fragments
- VoiceXML Elements
- Attributes and Values
- Using Grammars in VoiceXML
Developing Speech Interfaces

• Speech interfaces can be developed using
  – general-purpose languages (e.g. C++, Java, Python)
  – special-purpose languages (e.g. VoiceXML, SALT)

• A special-purpose language can
  – simplify application development
  – separate interaction code from application logic code
  – reduce network traffic
  – provide portability and simplicity
  – support prototyping and refinement.
VoiceXML

- VoiceXML
  - is a high level special-purpose (markup) language
  - is designed to describe the user interface
  - is portable between speech servers
  - 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.
W3C Speech Interface Framework

- VoiceXML relies on other markup languages:

- VoiceXML uses JavaScript as client-side scripting language.
A VoiceXML Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
  <form>
    <block>
      Welcome to Ajax Travel.
      <audio src = "http://www.pre-recorded.audiostream.wav"/>
    </block>
  </form>
  ...
</vxml>
```
Recap: 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
Recap: 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 an 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.
Recap: 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.
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 version="1.0" encoding="UTF-8"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
<form>
  <block>
    Welcome to Ajax Travel.
  </block>
</form>
A VoiceXML Code Fragment

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

    <grammar root = "aUser"
        mode = "voice"
        type = "application/srgs+xml">
        <rule id = "aUser">
            <one-of>
                <item> fred </item>
                <item> sam </item>
            </one-of>
        </rule>
    </grammar>

</field>
```
A VoiceXML Code Fragment

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

<!–- ... transition to another dialog in the current document -->
<menu id="travel">
  <prompt>
    Do you want to travel by air, rail, or boat?
  </prompt>

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

  <!-- ... more choices on the following slides -->
A VoiceXML Code Fragment

```
<choice next = "#train">
    <grammar root = "by_train"
        mode = "voice"
        type = "application/srgs+xml">
        <rule id = "by_train">
            <item> rail </item>
        </rule>
    </grammar>
</choice>
```
A VoiceXML Code Fragment

```xml
<choice next = "#boat">
  <grammar root = "by_boat"
    mode = "voice"
    type = "application/srgs+xml">
    <rule id = "by_boat">
      <item> boat </item>
    </rule>
  </grammar>
</choice>
</menu>
```
A VoiceXML Code Fragment

...<form id="train" encoding="UTF-8"> >
   <block>
      <prompt>
         You have selected to travel by rail.
         <!-- Details for making travel arrangement
            would be here in a real application -->
      </prompt>
   </block>
</form>
...</vxml>
VoiceXML Elements

- So far, we used the following 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>` queues speech synthesis and audio output to 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>`: defines a menu item
- `<grammar>`: specifies a speech recognition or DTMF grammar
- `<goto>`: goes to another dialog in the same or different document
Another Example Dialog

Computer: Welcome to the weather information service.
Computer: What state?
User: Help!
Computer: Please speak the state for which you want the weather.
Computer: What city?
User: Seattle.
Attributes and Values

• VoiceXML elements have attributes with specific values:

```xml
<?xml version = "1.0" encoding="UTF-8"?>
<vxml version = "2.0" xmlns = "http://www.w3.org/2001/vxml">
  <form id = "weather_info">
    <block> Welcome to the weather information service. </block>
    <field name = "state">
      <prompt> What state? </prompt>
      <grammar src = "state.grxml" type = "application/srgs+xml"/>
      <catch event = "help">
        Please speak the state for which you want the weather.
      </catch>
    </field>
  </form>
</vxml>
```
Attributes and Values

```xml
<field name = "city">
  <prompt> What city? </prompt>
  <grammar src = "city.grxml" type = "application/srgs+xml"/>
  <catch event = "help">
    Please speak the city for which you want the weather.
  </catch>
</field>

<block>
  <submit next = "cgi-bin/weather.py" namelist = "city state"/>
</block>
</form>
</vxml>
```
More VoiceXML Elements

• Previous example:
  – `<catch>` catches an event
  – `<submit>` submits values to a document server

• Next example:
  – `<noinput>` catches a noinput event
  – `<nomatch>` catches a nomatch event
  – `<if>` simple conditional logic
  – `<else>` used in `<if>` element
Example Dialog

Computer: Are you sleepy?
User: <says nothing>
Computer: Hey, don’t sleep!
User: Ooops.
Computer: Say 'yes' or 'no'.
User: Yes.
Computer: So, you are sleepy, me too.
<?xml version = "1.0" encoding="UTF-8"?>
<vxml version = "2.0" xmlns = "http://www.w3.org/2001/vxml">

<form id = "start">
    <field name = "answer">
        <prompt> Are you sleepy? </prompt>
        <grammar src = "yesno.grxml" type = "application/srgs+xml"/>
        <noinput> Hey, don't sleep! </noinput>
        <nomatch> Say 'yes' or 'no'. </nomatch>
    </field>
</form>

Are You Sleepy?

```xml
<filled>
  <if cond = "answer == 'yes'">
    So you are sleepy. Me too.
  </if>
  <else/>
    So you are not sleepy. But I am.
  </else/>
</filled>
</field>
</form>
</vxml>

• Note: "answer == 'yes' " is a JavaScript expression!
<?xml version = "1.0"?>

<grammar root = "main" version = "1.0" xml:lang="en">
   <rule id = "main" scope = "public">
      <one-of>
         <item tag = "yes"/> <ruleref uri = "#yes"/> </item>
         <item tag = "no"/> <ruleref uri = "#no"/> </item>
      </one-of>
   </rule>
</grammar>
<rule id = "yes">
  <one-of>
    <item> yes </item>
    <item> yeah </item>
    <item> yep </item>
    <item> sure </item>
  </one-of>
</rule>

<rule id = "no">
  <one-of>
    <item> no </item>
    <item> not </item>
    <item> nope </item>
  </one-of>
</rule>
Take-Home Messages

• VoiceXML
  – is a special-purpose programming language,
  – simplifies the development of voice-enabled applications,
  – relies on other markup languages,
  – uses JavaScript as client-side scripting language,
  – uses forms and menus as basic elements,
  – is interpreted by a voice browser.