Special Lecture (406)
Spoken Language Dialog Systems
Mixed Initiative

Rolf Schwitter
{schwitt}@ics.mq.edu.au
Today’s Program (Part I)

• Dialog Styles
• Problems with Form Filling Dialog Model
• Mixed Initiative Dialog Model
• Mixed Initiative in VoiceXML
• Implementation
## Dialog Styles

<table>
<thead>
<tr>
<th>Mixed Initiative</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Application Directed</strong></td>
</tr>
<tr>
<td>Forms</td>
</tr>
<tr>
<td>DTMF Menus</td>
</tr>
<tr>
<td><strong>User Directed</strong></td>
</tr>
<tr>
<td>Dictation</td>
</tr>
<tr>
<td>Query</td>
</tr>
<tr>
<td>Command and Control</td>
</tr>
</tbody>
</table>
Form Filling Dialog Model

• In a form filling dialog model
  the application typically prompts the caller
  for discrete pieces of information
  in a pre-determined order.

• In this model, the VoiceXML application mainly
  consists of a number of call states
  that collect input from the caller.
Example: Form Filling Dialog

Computer: Thanks for calling ACME Travel Company.
How can I help you today?

Caller: I'd like to book a flight.

Computer: Okay. Where are you flying from?

Caller: I wanna fly from San Francisco, California.

Computer: You want to fly from San Francisco, California.
Where are you flying to?
Example: Form Filling Dialog

Caller: To Boston, Massachusetts.

Computer: Okay, you'll be traveling from San Francisco, California to Boston, Massachusetts. Is that correct?

Caller: Yes.
Problems with the Form Filling Dialog Model

- For callers, "form filling" can be become quite cumbersome.
- Especially when callers are accustomed to provide multiple pieces of information
  - in succession without interruption of intermediary prompts
  - in a different order than specified by the application.
- For example:
  - \((A + B + C)\)
  - \((B + A) + C\)
Mixed Initiative Dialog Model

• In a mixed initiative dialog model
  – the call flow
    can be directed by the caller or by the application
  – the application
    collects pieces of information in a single call state.
• Caution: VoiceXML does not allow for true free-form utterances.
• However, it is possible to approximate this dialog style.
Example: Mixed Initiative Dialog

Computer:  Thanks for calling ACME Travel Company.
           How can I help you today?
Caller:    I'd like to book a flight.
Computer:  Okay. What is your point of origin, and where are you going?
Caller:    I wanna fly from San Francisco, California to Boston, Massachusetts.
Computer:  Okay, travelling from San Francisco, California to Boston, Massachusetts. Is that correct?
Caller:    Yes.
Another Example: Mixed Initiative Dialog

Computer: What is your birth date and Social Security Number?
Caller: 101-44-4938.
Computer: Got your SSN.
What is your birth date?
Please include the month, day, and year.
Caller: July 28, 1959.
Computer: Got your birth date.
In summary, you said that you were born on 07/28/1959, and your SSN is 101-44-4938.
Another Example: Mixed Initiative Dialog

Computer: What is your birth date and Social Security Number?
Caller: July 28, 1959.
Computer: Got your birth date.
What is your Social Security Number?
Caller: 101-44-4938.
Computer: Got your SSN.
In summary, you said that you were born on 07/28/1959, and your SSN is 101-44-4938.
How does a Mixed Initiative Dialog Work?

- A mixed initiative dialog in VoiceXML is essentially a way to
  - prompt the caller to answer multiple questions at once
  - have a grammar construct that allows this to happen
  - and then fall back on machine-directed dialog (if needed)
  - that sequentially walks the caller through any questions they neglected to answer in their original response.
How to Accomplish this Task?

• The following things need to be done:
  – define subgrammars to collect each piece of information
  – define a form level grammar that uses the subgrammars to collect the information
  – define a mixed initiative dialog that collects input from the caller.
Defining the Subgrammar

- In the ACME example, the origin and destination are similar pieces of information:
  - from San Francisco, California
  - to Boston, Massachusetts
- Therefore, a single subgrammar can be defined for this data.
Defining the Subgrammar (airports.gsl)

Airports [ [ (albuquerque new_mexico) (a b q) ]
  { return(albuquerque_nm) }
  [ (boston massachusetts) (b o s) ]
  { return(boston_ma) }
  [ (charlotte north_carolina) (c l t) ]
  { return(charlotte_nc) }
  [ (los angeles california) (l a x) ]
  { return(los_angeles_ca) }
  [ (portland oregon) (p d x) ]
  { return(portland_or) }
  [ (san francisco california) (s f o) ]
  { return(san_francisco_ca) }
  [ (seattle washington) (s e a) ]
  { return(seattle_wa) }
]
Defining the Form Level Grammar

• In the next step, we need to define a grammar that utilizes the Airport subgrammar.

• The caller should be able to utter sentences such as:
  – I want to go to Albuquerque, New Mexico from San Antonio, Texas.
  – I wanna fly from S F O to Boston, Massachusetts.
  – From Cleveland, Ohio. To Portland, Oregon.
Defining the Form Level Grammar (travel.gsl)

( ?( i [(want to) wanna] [ go fly ] )
[ ( from Airports:x ) { <from $x> }  
  ( to Airports:y ) { <to $y> }  
  ( from Airports:x to Airports:y ) { <from $x> <to $y> }  
  ( to Airports:y from Airports:x ) { <from $x> <to $y> }  
] )
Defining the Form Level Grammar

- The recogniser fills the appropriate slot depending on the utterance:

<table>
<thead>
<tr>
<th>Sentence</th>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>I want to go to Albuquerque, New Mexico from San</td>
<td>san_antonio_tx</td>
<td>albuquerque_nm</td>
</tr>
<tr>
<td>Antonio, Texas.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I wanna fly from S F O to Boston, Massachusetts.</td>
<td>san_francisco_ca</td>
<td>boston_ma</td>
</tr>
<tr>
<td>From Cleveland, Ohio</td>
<td>cleveland oh</td>
<td>n/a</td>
</tr>
<tr>
<td>To Portland, Oregon</td>
<td>n/a</td>
<td>portland_or</td>
</tr>
</tbody>
</table>
Defining the Mixed Initiative Dialog

• A mixed initiative dialog in VoiceXML consists of the following parts:
  – grammars defined at form level (already done)
  – an <initial> element that prompts for form-wide information
  – a field for each piece of information to collect
  – a confirmation field.
Example: VoiceXML with Form

```xml
<vxml version="2.0">

<!-- helper script that maps city/state names to audio/tts -->
<script src="citystate.js"/>
<script>
  var csobj = new CityStateReader();
</script>

<!-- application entrypoint -->
<form id="start">
  <block>
    <audio src="wav/01.wav">Welcome to ACME travel.</audio>
    <break time="500ms"/>
    <goto next="#get_origin_dest"/>
  </block>
</form>

<!-- retrieves the origin and destination using mixed initiative -->
<form id="get_origin_dest">
  <property name="confidencelevel" value="0.4"/>
  <grammar type="application/x-gsl" mode="voice" src="travel.gsl"/>
  <catch event="nomatch noinput">
    <audio src="wav/04.wav">sorry, i didn't catch that.</audio>
    <reprompt/>
  </catch>
</form>
```

The `<initial>` Element

- The `<initial>` element prompts the caller for a mixed initiative dialog.
- The `<initial>` element
  - should not define any grammars or a filled element
    since it relies upon those elements at the form level
  - should appear before the other field elements in the dialog
    so that the VoiceXML interpreter executes it first.
Example: `<initial>` Element

<!-- designates the initial state in a mixed initiative dialog -->
<initial name="init">
  <prompt>
    <audio src="wav/03.wav">
      please tell me your starting and destination cities.
    </audio>
  </prompt>
  <catch event="nomatch noinput">
    <audio src="wav/04.wav">sorry, i didn't catch that.</audio>
    <audio src="wav/11.wav">
      please say where you'd like to go to and from.
    </audio>
  </catch>
  <catch event="nomatch noinput" count="2">
    <audio src="wav/04.wav">sorry, i didn't catch that.</audio>
    <assign name="init" expr="true"/>
  </catch>
  <reprompt/>
</catch>
  <help>
    <audio>
      to book a flight you need to specify your origin and destination cities.
    </audio>
    <audio>
      for example, you can say, from san francisco, california to boston massachusetts.
    </audio>
  </help>
</initial>
Explanation: `<initial>` Element

- The recogniser attempts to match what the caller says in response to the initial prompt against the form level grammar.
- The `<initial>` element can contain all the standard events:
  - noinput
  - nomatch
  - help

to aid the user in completing the dialog.
Explanation: `<initial>` Element

- If any of the fields in the form are filled by the caller, the `<initial>` element will not be revisited unless its form item variable is cleared.
- In our example:
  If the recognition fails twice due to a timeout (noinput) or a misrecognition (nomatch), the VoiceXML code sets the value of the variable "init" to "true":

    `<assign name = "init" expr = "true"/>`
Defining the Fields

• If the caller
  – does not provide all necessary information in the <initial> dialog
  – then the interpreter executes the contents of individual fields,
  – if they exist, to collect the missing information.
Example: Field

```xml
<!-- retrieve origin in case it didn't happen in initial state -->
<field name="origin" slot="from">
  <grammar type="application/x-gsl" mode="voice" src="airports.gsl"/>
  <prompt>
    <audio src="wav/06.wav">where are you flying from?</audio>
  </prompt>
  <filled>
    <prompt>
      <value expr="csobj.GetCSTTS(origin)"/>
    </prompt>
  </filled>
</field>
```
Example: Field

<!-- retrieve destination in case it didn't happen in initial state -->
<field name="to">
  <grammar type="application/x-gsl" mode="voice" src="airports.gsl"/>
  <prompt>
    <audio src="wav/08.wav"> where do you want to go? </audio>
  </prompt>
  <filled>
    <prompt>
      <value expr="csobj.GetCSTTS(to)"/>
    </prompt>
  </filled>
</field>
Defining a Confirmation Field

- To ensure that the dialog has collected the information from the user correctly, consider defining a confirmation field.
- This field should be the last one defined in the dialog.
- The interpreter will visit the confirmation field only after the data collection fields are filled.
Example: Confirmation

<!-- confirm origin and destination -->
<field name="confirm" type="boolean">
  <prompt>
    <audio src="wav/10.wav">
      Okay! To summarize, you'd like to fly from
    </audio>
    <prompt>
      <value expr="cstobj.GetCSTTS(origin)"/>
    </prompt>
    <audio src="wav/05.wav"> to </audio>
    <prompt>
      <value expr="cstobj.GetCSTTS(to)"/>
    </prompt>
    <audio> is that correct? </audio>
  </prompt>
  <catch event="nomatch nocinput">
    Sorry I didn't get that.
  </catch>
  <filled>
    <if cond="confirm">
      <goto next="#bookit"/>
    </else/>
    <clear/>
  </if>
</filled>
</field>
Example: End of VoiceXML Code

```xml
</form>

<!-- move along now that origin and dest have been collected... -->
<form id="bookit">
  <block>
    <audio>booking your flight</audio>
    <goto next="#start"/>
  </block>
</form>
</vxml>
```
Take-Home Messages

• Mixed initiative dialog offers more control over the way fields are filled than is provided by a standard form filling dialog.

• A mixed initiative dialog in VoiceXML
  – prompts the caller to answer multiple questions at once
  – needs a grammar construct that accepts such answers
  – falls back on directed dialog (if needed)
  – has an (optional) confirmation field.
Special Lecture (406)
Spoken Language Dialog Systems
Towards Multimodality

Rolf Schwitter
{schwitt}@ics.mq.edu.au
Today’s Program (Part II)

- Multimodal Dialogs
- Current Technical Limitations
- Extending VoiceXML for Multimodality
- Speech Tags
- Point and Speak
The Dream

• Adapting the Web to allow multiple modes of interaction:
  – GUI, speech, vision, pen, gestures, and haptic interfaces.

• Augmenting human to computer and human to human interaction:
  – services that involve multiple devices and multiple people.

• Anywhere, any device, any time:
  – services that adapt to the device, user preferences and environmental conditions.

• Accessible to all.
Scenario

• User wants to make a flight reservation with his mobile device while s/he is on the way to work.

• The user initiates the service via means of making a phone call to a multimodal service.

• The user moves between networks with very different bandwidth.

• The user is offered the flexibility to interact using the preferred and most appropriate modes for the situation.
Scenario

• While sitting in a train,
  – the use of pen and handwriting can achieve higher accuracy than speech and protect privacy.

• When the user is walking,
  – the input and output modalities that are more appropriate would be voice with some visual output.

• Finally at the office,
  – the user can use pen and voice in a synergistic way.
Multimodal Dialogs

• Today, spoken interfaces are possible that prompt users with speech and understand simple words or phrases.
• As the technology improves we can expect richer conversations.
• Speech can be combined with other modes of interaction.
• Multimodal interaction will enable users to
  – speak
  – write and type
  – hear and see
  using a more natural interface than today's single mode browsers.
Multimodal Access

- Different modalities may be supported
  - on a single device or
  - on separate devices
  working in tandem.
- For example, you could be
  - talking into your mobile phone and
  - seeing the results on a PDA.
- Voice may also be offered as an adjunct to browsers.
Example: Multimodal Browser
Current Limitations

• Mobile devices working in isolation generally lack power.
• They only recognize a few hundred spoken commands.
• Storage limitations restrict the use of pre-recorded speech prompts.
• Small speech synthesizers are possible but tend to produce robotic sounding speech that many users find tiring to listen to.
An Intermediate Solution

- Process speech recognition and synthesis
  - remotely on more powerful platforms.
- Process simple dialogs
  - locally.
- Couple the device
  - with a remote dialog engine for richer interactions.
Requirements to Multimodal Applications

• Multimodal applications should be able to adapt to
  – changing device capabilities
  – user preferences and
  – environmental conditions.

• For instance, users should be able to
  disable speech input and output when this would be distracting to nearby people.
Requirements to Multimodal Applications

• It should be easy for developers to
  – tailor applications to dynamically adapt to changes
  – make best use of the available modes of interaction
  – create applications involving multiple devices and users.
Ideal Architecture

- MM Browser with ASR and TTS
- Multiple Operator Network
- MM Gateway
- Internet
- Server
Short-term Architecture

- MM Browser
- Multiple Operator Network
- Internet
- MM Gateway
- ASR Server
- Audio Server
- Server
Extending VoiceXML for Multimodality

• One possibility to support multimodality in VoiceXML is to
  – extend the <prompt> and <grammar> elements with three new tags:
    – <sequence-of>
      indicates that the actions should be performed in sequence
    – <one-of>
      indicates that only one of the actions needs to occur
    – <all-of>
      indicates that all of the actions must occur.
Example: VoiceXML for Multimodality

<prompt>
  <all-of>
    <media type = "voice">
      Welcome to Ajax Travel
    </media>
    <media type = "display-text">
      Welcome to Ajax Travel
    </media>
  </all-of>
</prompt>
Example: VoiceXML for Multimodality

<form id = "form1">
  <field name = "User_name">
    <prompt>
      <all-of>
        <media type = "voice">
          Your name?
        </media>
        <media type = "display-text">
          Your name?
        </media>
      </all-of>
    </prompt>
  </field>
</form>
Example: VoiceXML for Multimodality

```xml
<grammar>
  <one-of>
    <mode = "voice" src = "user-name-voice.grammar" />
    <mode = "keyboard" src = "user-name-text.grammar" />
  </one-of>
</grammar>
</field>
</form>
```
Example: VoiceXML for Multimodality

<menu>
  <prompt>
    <all-of>
      <media type = "voice">
        How would you like to travel?
        By airplane, train, boat?
      </media>
      <media type = "display-text">
        How would you like to travel?
        By airplane, train, boat?
      </media>
    </all-of>
  </prompt>
</menu>
Example: VoiceXML for Multimodality

<choice next = "www.airplane.xml">
  <one-of>
    <mode = "voice" src = "plane-voice.grammar"/>
    <mode = "keyboard" src = "plane-text.grammar"/>
  </one-of>
</choice>
Example: VoiceXML for Multimodality

```xml
<choice next = "www.train.xml">
  <one-of>
    <mode = "voice" src = "train-voice.grammar"/>
    <mode = "keyboard" src = "train-text.grammar"/>
  </one-of>
</choice>
```
Example: VoiceXML for Multimodality

<choice next = "www.boat.xml">
  <one-of>
    <mode = "voice"  src = "boat-voice.grammar"/>
    <mode = "keyboard" src = "boat-text.grammar"/>
  </one-of>
</choice>
</menu>
Example: Dialog for Multimodality

<table>
<thead>
<tr>
<th>Step</th>
<th>Prompt to user (all of)</th>
<th>User Response</th>
<th>(one of)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Spoken</strong></td>
<td><strong>Spoken</strong></td>
<td><strong>Keyboard</strong></td>
</tr>
<tr>
<td>1</td>
<td>Welcome to Ajax Travel</td>
<td>Welcome to Ajax Travel</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Your name</td>
<td>Your name</td>
<td>Joe Kagan</td>
</tr>
<tr>
<td>3</td>
<td>How would you like to travel, by airplane, train, boat?</td>
<td>How would you like to travel? &lt;menu airplane, train, boat&gt;</td>
<td>Airplane</td>
</tr>
</tbody>
</table>

Joe Kagan

Welcome to Ajax Travel
The following grammar would require the user first to speak, then to point, then to speak again, and finally to point.

```xml
<grammar>
  <sequence-of>
    <mode = "voice"> I want to fly from here </mode>
    <mode = "pen-point"/>
    <mode = "voice"> to here </mode>
    <mode = "pen-point"/>
  </sequence-of>
</grammar>
```
Speech Tags

- An alternative to extending VoiceXML is to embed speech tags into (X)HTML documents as advocated by the SALT Forum.
- SALT stands for (Speech Application Language Tags).
- SALT adds a speech and telephony interface to web applications.
- The SALT specification is available at www.saltforum.org.
SALT

- The main elements of SALT include:
  - `<salt: prompt>`
    produces a verbal prompt
  - `<salt: listen>`
    invokes a speech recognition engine
  `<salt: grammar>`
    specifies the grammar for the speech recogniser
  `<salt: bind>`
    moves the speech recognition result to the HTML variable.
Example: Dialog in HTML + SALT

```html
  <head>
    <title> Ajax Travel </title>
  </head>
  <body onload = "RunAsk()">
      <!-- The RunAsk() function activates prompts and recognitions
           until the values of the input fields are filled.
           Implementations of SALT are expected to provide
           scriptlets for common dialog processing task. -->
</body>
</html>
```
Example: Dialog in HTML + SALT

```
<form id = "form1" >
  User name
  <input = "text" id = "user-name">
  <salt:listen id = "reco-user-name">
    <salt:grammar name = "g-user-name"
                 src = "user-name.gram"/>
    <salt:bind targetElement = "user-name"/>
  </salt:listen>
</form>
```
Example: Dialog in HTML + SALT

```html
<form id = "form2" >
    Mode of transportation
    <input = "text" id = "transportation-mode">
    <salt:listen id = "reco-transportation-mode">
        <salt:grammar name = "g-transportation-mode"
            src = "transportation-mode.gram"/>
        <salt:bind targetElement = "transportation-mode"/>
    </salt:listen>
</form>

...
Point and Speak

- This style of user interface is called "point and speak".
- The user points to a field and then speaks.
- When the user points to a field, the browser loads a grammar.
- The recognition result is assigned to the variable of the input box.

```html
<input = "text" id = "transportation-mode">
...
<salt:bind targetElement = "transportation-mode"/>
...
```
Point and Speak

- The dialog depends on the order in which the user points and speaks.

<table>
<thead>
<tr>
<th>Step</th>
<th>User Selects Objects</th>
<th>User Action</th>
<th>User Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><em>Speak</em></td>
<td><em>Keyboard</em></td>
</tr>
<tr>
<td>1</td>
<td>Points to &quot;User name&quot;</td>
<td>Joe Kagan</td>
<td>Joe Kagan</td>
</tr>
<tr>
<td>2</td>
<td>Points to &quot;Mode of transportation&quot;</td>
<td>Airplane</td>
<td>Airplane</td>
</tr>
</tbody>
</table>
Take-Home Messages

• The key to building multimodal applications is to use the mode that best enables the user to interact with the multimodal application.

• Each mode has its strengths and weaknesses.

• VoiceXML can be extended to support multimodality: in particular (complex) application-directed dialogs.

• Voice tags can be embedded into HTML to support multimodality: in particular (simple) user-directed dialogs.