COMP349
Spoken Language Dialogue Systems
Mixed Initiative

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

- Dialog Styles
- Problems with Form Filling Dialog Model
- Mixed Initiative Dialog Model
- Mixed Initiative in VoiceXML
- Implementations
## Dialog Styles

<table>
<thead>
<tr>
<th>Mixed Initiative</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Application Directed</strong></td>
<td><strong>User Directed</strong></td>
</tr>
<tr>
<td>Forms</td>
<td>Dictation</td>
</tr>
<tr>
<td>DTMF Menus</td>
<td>Query</td>
</tr>
<tr>
<td></td>
<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 want to fly from San Francisco, California.
Computer: Okay, 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 travelling from San Francisco, California to Boston, Massachusetts. Is that correct?

Caller: Yes.
Problems with the Form Filling Dialog Model

• For callers, "form filling" can 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 want to 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 for multiple pieces of information 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.

• The mixed initiative dialog can be built on top of a form-filling dialog.
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 form level grammar that utilizes the Airport subgrammar.

• The caller should be able to utter sentences such as:
  – I wanna fly from S F O to Boston, Massachusetts.
  – I want to go to Albuquerque, New Mexico from San Antonio, Texas.
  – 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 Antonio, Texas.</td>
<td>san_antonio_tx</td>
<td>albuquerque_nm</td>
</tr>
<tr>
<td>I wanna fly from S F O to Boston, Massachusetts.</td>
<td>san_francisco_ca</td>
<td>bostom_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 Level Grammar

```xml
<vxml version="2.0">
  <!-- helper script that maps city/state names to audio/tts -->
  <script src="citystate.js"/>
  <script>
    var csoj = 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>
</vxml>
```
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"/>
    <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 "init" 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: First Field with Subgrammar

```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="cobj.GetCSTTS(origin)="/>
    </prompt>
    <filled>
    </filled>
  </filled>
</field>
```
Example: Second Field with Subgrammar

```xml
<!-- 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="csobj.GetCSTTS(origin)"/>
  </prompt>
  <audio src="wav/05.wav"> to </audio>
  <prompt>
    <value expr="csobj.GetCSTTS(to)"/>
  </prompt>
  <audio>is that correct?</audio>
</prompt>
  <catch event="nomatch noinput">
    Sorry I didn't get that.
  </catch>
</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>
```
Appendix: An Example with an XML Grammar

```xml
<vxml version = "1.0" encoding="UTF-8"/>
<vxml version = "2.0" xmlns = "http://www.w3.org/2001/vxml">
<form id = "get_from_and_to_cities">
   <grammar src = "cities.grxml"/>
   <block>
      Welcome to the automated airline ticket reservation system.
   </block>
</form>
```
Appendix: An Example with XML Grammar

<initial name = "init">

Where do you want to fly from and to?

<nomatch count = "1">
  Sorry, I did not understand you.
  Say for example: "from London to Sydney".
</nomatch>

<nomatch count = "2">
  I am sorry, I still don't understand.
  I will ask you for information one piece at a time.
  <assign name = "init" expr = "true"/>
  <reprompt/>
</nomatch>

</initial>
Appendix: An Example with XML Grammar

<field name = "from_city" slot = "from">
    <grammar src = "cities.grxml#from"/>
    From which city are you leaving?
    <prompt count = "2">
        Tell me from which city you are leaving.</prompt>
</field>

<field name = "to_city" slot = "to">
    <grammar src = "cities.grxml#to"/>
    To which city do you want to fly?
    <prompt count = "2">
        Tell me to which city you want to fly.</prompt>
</field>
A ticket from <value expr = "from_city"/> to <value expr = "to_city"/> is reserved for you.

Your departure city is <value expr = "from_city"/>.

Your arrival city is <value expr = "to_city"/>.
Appendix: An Example with XML Grammar

<filled>
  <if cond = "from_city == to_city">
    <prompt>
      Sorry, you can't fly from
      <value expr = "from_city"/> to
      <value expr = "to_city"/>!
    </prompt>
    <prompt>
      Please repeat your booking.
    </prompt>
    <clear namelist = "init from_city to_city"/>
  </if>
</filled>

</form>

</vxml>
<?xml version = "1.0"?>
<grammar root = "main" version = "1.0">
  <rule id = "main" scope = "public">
    <item repeat = "0-1">
      <ruleref uri = "#filler"/>
    </item>

    <one-of>
      <item>
        from
        <ruleref uri = "#city"/> <tag> $.from = $city; </tag>
        to
        <ruleref uri = "#city"/> <tag> $.to = $city; </tag>
      </item>
    </one-of>
  </rule>
</grammar>
Appendix: XML Grammar

```xml
<one-of>
  <item>
    from
    <ruleref uri="#city"/>
    <tag> $.from = $city; </tag>
  </item>
  <item>
    to
    <ruleref uri="#city"/>
    <tag> $.to = $city; </tag>
  </item>
</one-of>
</rule>
```
Appendix: XML Grammar

<rule id = "from" scope = "public">
  <item repeat = "0-1">
    <ruleref uri = "#filler"/>
  </item>
  <item repeat = "0-1">
    from
  </item>
  <ruleref uri = "#city"/>
  <tag> $.from = $city; </tag>
</rule>
Appendix: XML Grammar

<rule id = "to" scope = "public">
    <item repeat = "0-1">
        <ruleref uri = "#filler"/>
    </item>
    <item repeat = "0-1">
        to
    </item>
    <ruleref uri = "#city"/>
    <tag> $.to = $city; </tag>
</rule>
Appendix: XML Grammar

<rule id = "filler">
   I want
   <item repeat = "0-1">
      to fly
   </item>
</rule>

<rule id = "city">
   <one-of>
      <item> London </item>
      <item> New York </item>
      <item> Sydney </item>
      <item> Tokyo </item>
   </one-of>
</rule>
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 for multiple pieces of information at once
  – needs a grammar construct that accepts such answers
  – falls back on directed dialog (if needed)
  – has an (optional) confirmation field.