



# Internet Applications

## Lecture 14 – Building VoiceXML Applications

Jason D. Williams, Feb 2004

Cambridge University Engineering Department  
Speech Vision and Robotics Group

# Overview of this lecture



- An example spoken dialogue system
- VoiceXML Architectural challenges & options
- Implementation



# Executrain - requirements

## Use case scenario:

*Joan lives in Doncaster, and is at a client meeting in London today. She's stayed later than she expected, and is wondering whether to run to the train station, or stay and get dinner with the client. She needs to find out when the next few trains are from London to Doncaster.*

## Requirements

- Intended for busy people who need the *next few* train times
- System needs to ask for “from” and “to” station
  - Not: time, date, how many people, etc.
- Caller base will consist of many repeat callers
  - Allow “shortcuts” *where possible*

*For simplicity, we'll make a few assumptions (eg, all trains are direct trains)*

# ExecuTrain Sample Dialogues (1/2)



System: Thanks for calling ExecuTrain. Which stations are you travelling between?

User: To Doncaster from London

*{ from: london; to: doncaster }*

System: From London to Doncaster. Is that right?

User: yes

*{ confirm: yes }*

System: One moment...

The next train departs London Kings Cross at 18:32 and arrives in Doncaster at 19:36.

The train after that departs London Kings Cross at 19:32 and arrives in Doncaster at 20:36.

You can say REPEAT, or START OVER.

User: *[hang up]*

# ExecuTrain Sample Dialogues (2/2)



System: Thanks for calling ExecuTrain. Which stations are you travelling between?

User: I'm going to Doncaster

{ to: doncaster }

System: Where are you starting from?

User: London

{ from: london }

System: From London to Doncaster. Is that right?

...

---

System: Thanks for calling ExecuTrain. Which stations are you travelling between?

User: Uhhh...

[ platform raises no-match event ]

System: Sorry, I didn't catch that. Where are you starting from?

User: London

{ from: london }

System: Where are you travelling to?

User: I'm going to Doncaster

{ to: york } **Note: False-Accept**

System: From London to York. Is that right?

User: No

{ confirm: no }

System: Sorry, let me try that again. Which stations are you travelling between?

# ExecuTrain VUI design and architecture



## VUI Design decisions

- Globals
  - Short dialog: implement “start over” but not “go back”
  - Repeat
  - (For brevity, we won’t show “help”)
  - End call after 3 *nm*’s or 3 *ni*’s
- Many repeat callers
  - Keep initial prompts short & provide guidance/examples in escalating *nm* & *ni* messages
  - Allow mixed initiative for capturing “from” and “to”
- Use explicit confirmation (and allow just “yes” and “no”)
  - Dialog correction language is more complex
  - After a “no”, re-capture both slots & confirm again
- In output state, automatically repeat the train times in *nm* & *ni* messages

# Architectural options & decisions



## Issue

## Options

How is state maintained across VoiceXML pages?

- On browser: App variables (In App Root)
- On server: `<submit>` & server-side variables

How to insert dynamic content on pages?

- Static pages: `<subdialog>` trick
- Mostly static pages: ASP, JSP, XSL, etc.
- Generate pages from scratch

How do I implement ... HELP? REPEAT? START AGAIN? AGENT?

- `<link event="..."/>` or `<link next="..."/>`
- `<filled>` code

How do I implement GO BACK?

- Simple: Specify transitions manually
- Better: Maintain history (stack) of `<form>`'s

When do I use:

- (A) 2 `<form>`'s, each with 1 `<field>`, and
- (B) 1 `<form>` and 2 `<field>`'s?

- (A): simpler, "go back", page breaks, `<goto>`
- (B): mixed initiative, some good control of FIA *inside* `<form>` eg `<clear ...>`

How much code goes on one page?

- Interpreter speed; network latency; caching
- How is back-end data accessed?



# Architecture summary

- 2 leaf pages
  - Page 1: From/to/confirm – 1 form or 2?
  - Page 2: 1 form
- Insert train times with ASP/JSP directly into VoiceXML code
- Root page contains common code
  - Use global <link> to event for “repeat”
  - Use global <link> to URL for “start again”

```
<!-- executrain-root.vxml -->  
<!-- repeat and start again links -->  
<link ...>  
<!-- event handlers for max NM/NI -->  
<catch ...>
```

```
<!-- executrain-getstations.vxml -->
```

```
<!-- get from & to -->  
<!-- confirm -->  
<submit next=“...”/>
```

```
<!-- executrain-output.vxml -->
```

```
<!-- train times filled in by ASP -->  
<!-- play available trains -->  
<!-- allow caller to repeat or start again -->
```



# Executrain-stations.gram

Note that root specifies the *default* root rule – we can over-ride this with <grammar root=“...”/>

Will be the **form-level grammar**: Caller can specify just one or both stations; **must include** “from” or “to”

Will be the **field-level grammars**. They need to use the same slot values as the form-level grammars. We’ll use <grammar root=“...”/>

The list of stations. \$station is utterance content by default

```
#ABNF 1.0;  
mode voice;  
language en-us;  
root $top;
```

```
$top = [(I ((would like) | want | need) to go) | (I’m going) ] (  
  (from $station {$.from = $station}) |  
  (to $station {$.to = $station}) |  
  (from $station {$.from = $station} to $station {$.to = $station})  
) [please];
```

```
$from = $station {$.from = $station};
```

```
$to = $station {$.to = $station};
```

```
$station = london | (the capitol {“london”}) | york | doncaster | ...
```



# Executrain-root.vxml

Event fired when caller says "repeat" or "start again"

Catches user-defined "end-call" event – plays prompt and hangs up

After caller has 3 nm's or ni's, throws the "end-call" event (handled above)

See next slide...

```
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
<link event="repeat">
  <grammar type="application/srgs" src="executrain-repeat.gram"/>
</link>
<link next="executrain-getstations.vxml">
  <grammar type="application/srgs" src="executrain-startagain.gram"/>
</link>
<catch event="end-call">
  <prompt>Hmm. Sorry, but I'm having trouble helping you today. Feel
  free to try again later. Goodbye. </prompt>
  <disconnect/>
</catch>
<catch event="nomatch" count="3">
  <throw event="end-call"/>
</catch>
<catch event="noinput" count="3">
  <throw event="end-call"/>
</catch>
<script>var initprompt="Thanks for calling ExecuTrain."; </script>
</vxml>
```

# Executrain-getstations.vxml



- Form level grammar
- 5 form elements
- 1 Try at the initial prompt
- Text of initial prompt is changed after first entry
- Field-level grammar
- Escalating *nm* and *ni* messages
- Handle “repeat”
- Fine-grain control over the “from” prompt using the prompt queue
- *Modal* confirmation field: need to re-add links
- Need to clear from, to and confirm fields

```
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml" application="executrain-root.vxml">
<form id="GETSTATIONS">
  <grammar type="application/srgs" src="executrain-stations.gram" root="top"/>
  <initial name="greeting">
    <prompt><value expr="initprompt"/>Which stations are you travelling between?</prompt>
    <noinput count="1"> <prompt> Sorry, I didn't hear anything. Where are you starting from?</prompt>
    <assign name="greeting" expr="true"/> </noinput>
    <nomatch count="1"> <prompt> Sorry, I didn't catch that. Where are you starting from?</prompt>
    <assign name="greeting" expr="true"/> </nomatch>
    <catch event="repeat"> <prompt> Which stations are you travelling between?</prompt> </catch>
  </initial>
  <block><script>initprompt = "We're back at the beginning.";</script></block>
  <field name="from">
    <grammar type="application/srgs" src="executrain-stations.gram" root="from"/>
    <!-- NB: no prompt element here -->
    <nomatch count="1"><prompt>Sorry, I didn't catch that. Tell me the station you're starting from... </prompt></nomatch>
    <nomatch count="2"><prompt>Sorry, I still didn't catch that. Say just the name of the station ... </prompt></nomatch>
    <noinput count="1"><prompt>Sorry, I didn't hear anything. Tell me the station you're ... </prompt></noinput>
    <noinput count="2"><prompt>Sorry, I still didn't hear anything. Say just the name of the station... </prompt></noinput>
    <catch event="repeat"> <prompt> Which station are you leaving from? </prompt> </catch>
  </field>
  <field name="to">
    <grammar type="application/srgs" src="executrain-stations.gram" root="to"/>
    <prompt>and where to?</prompt>
    ...
    <filled><if cond="!from"><prompt>and where from?</prompt></if></filled>
  </field>
  <field name="confirm" modal="true" type="boolean">
    <link event="repeat"> <grammar type="application/srgs" src="executrain-repeat.gram"/> </link>
    <link next="executrain-getstations.vxml"> <grammar type="application/srgs" src="executrain-startagain.gram"/> </link>
    <prompt>From <value expr="from"/> to <value expr="to"/>. Is that right?</prompt>
    <filled>
      <if cond="confirm"> <submit next="executrain-output.vxml" namelist="from to"/>
    <else/> <clear namelist="from to confirm"/>
      <prompt>Oh, sorry, let me try again. Which station are you starting from?</prompt> </if>
    </filled>
  </field>
</form>
</vxml>
```

# Executrain-output.vxml



- Content server has dynamically inserted this code

- This code is static
- CDATA allows special characters like <, >, etc.

- 1 form item
- **No** form or field grammars (just use <link> grammars in app root)
- For *nm* & *ni*, automatically repeat with <reprompt/>

```
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml" application="executrain-root.vxml">
<!-- BEGIN section automatically inserted by our content server -->
<script>
  var info = new Array();
  info[0] = new Object();
  info[0].fromName = "London Kings Cross"; info[0].fromTime = "18 32";
  info[0].toName = "York"; info[0].toTime = "19 36";
  info[1] = new Object();
  info[1].fromName = "London Kings Cross"; info[1].fromTime = "19 32";
  info[1].toName = "York"; info[1].toTime = "20 36";
</script>
<!-- END section automatically inserted by our content server -->
<var name="tt"/>
<script> <![CDATA[
  if (info.size == 0) { tt = "Sorry, there aren't any trains in the near future for those stations."; } else {
    var x;
    for (x in info) {
      if (x==0) { tt = "The next train departs "; } else { tt = tt+"The train after that departs "; }
      tt = tt+info[x].fromName + " at " + info[x].fromTime;
      tt = tt+" and arrives in " + info[x].toName + " at " + info[x].toTime + ". ";
    }
  }
]]></script>
<form id="OUTPUT">
  <field name="output">
    <prompt><value expr="tt"/></prompt>
    <prompt>You can say REPEAT or START AGAIN.</prompt>
    <nomatch count="1"><prompt>Here's that again. Remember, you can also say START AGAIN.</prompt>
      <reprompt/></nomatch>
    <nomatch count="2"><prompt>Here's that one more time. </prompt><reprompt/></nomatch>
    <noinput count="1"><prompt>Here's that again. Remember, you can also say START AGAIN. </prompt>
      <reprompt/></noinput>
    <noinput count="2"><prompt>Here's that one more time. </prompt><reprompt/></noinput>
    <catch event="repeat"><reprompt/> </catch>
  </field>
</form>
</vxml>
```



# A few closing notes

- The best way to really learn VoiceXML is...
  - To build some sample applications...
    - A variety of *free* tools exist; e.g., <http://café.bevocal.com>
    - You can either call them (in the USA...) or use an keyboard terminal
  - ... and to read the specifications (all linked from main VoiceXML spec)
- Caveat
  - W3C recommendations/proposals are at various stages of implementation
  - For example, no one yet fully implements Semantic Interp.

<http://www.w3.org/TR/semantic-interpretation/>

<http://www.w3.org/TR/speech-grammar/>

<http://www.w3.org/TR/voicexml20/>

*Thanks!*



*Thanks!*

Jason D. Williams  
jdw30@cam.ac.uk



# APPENDIX

Students are not responsible for  
new material beyond this point

# Architectural issues



*How can dynamic content be added to applications?*



*Static pages*

*“Write static VoiceXML”*

- *Trick with <subdialog>  
Proprietary tag: <data>*
- *Easiest to debug  
Best separation of  
application & data*
- *“State” on both client &  
server  
More on-page processing*

*Mostly static pages*

*“Write VoiceXML with special insert tags”*

- *ASP, JSP  
XML style sheets  
Perl*
- *Lots of skill transfer  
from the web world*
- *More effort to debug*

*Mostly dynamic pages*

*“Write in a higher-level language”*

- *Pages are “generated”  
eg server Java objects*
- *Promise of better  
tools*
- *Code generating code:  
very difficult to debug!*