



voxeo

The Future of VoiceXML: VoiceXML 3 Overview

Dan Burnett, Ph.D.
Dir. of Speech Technologies, Voxeo
Developer Jam Session
May 20, 2010

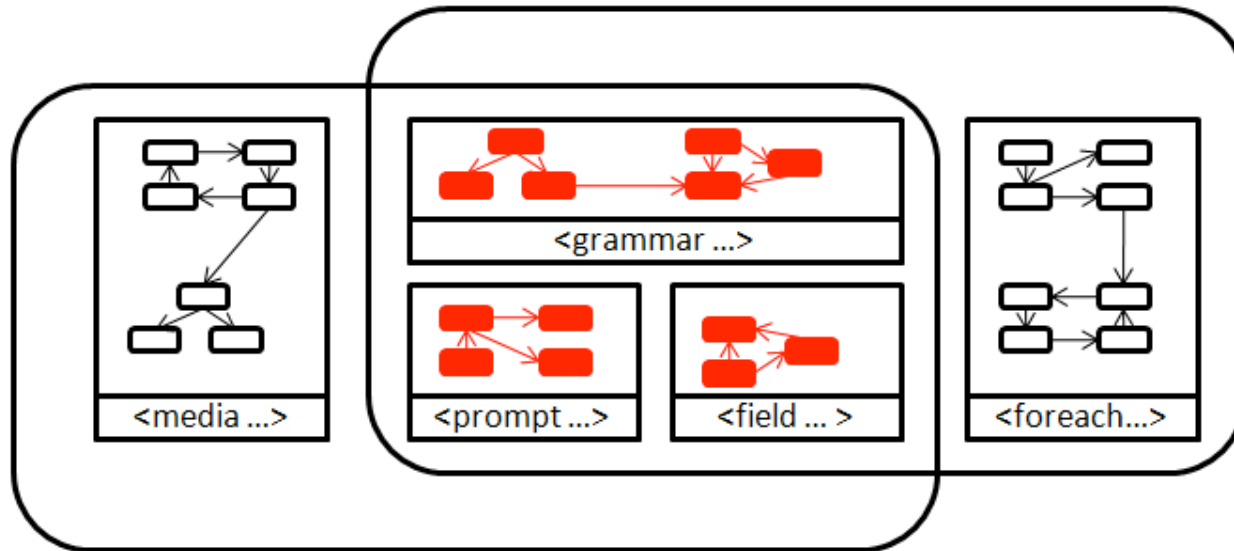
- ▶ V3 Motivations
- ▶ MVC, or DFP
- ▶ V3 is a presentation language
- ▶ V3 architecture
- ▶ New stuff
- ▶ Multimodal use of V3

- ▶ Extensibility
- ▶ FIA flexibility
- ▶ New features
- ▶ Better integration with other W3C languages

- ▶ Model-View-Controller paradigm
 - Model is the data being stored/tracked
 - View is the presentation
 - Controller is the piece that controls app flow
- ▶ VBWG: Data-Flow-Presentation
 - Data held in any XML format
 - Flow handled by SCXML, CCXML, or other such language
 - Presentation is HTML, Wiimote vibration, V3

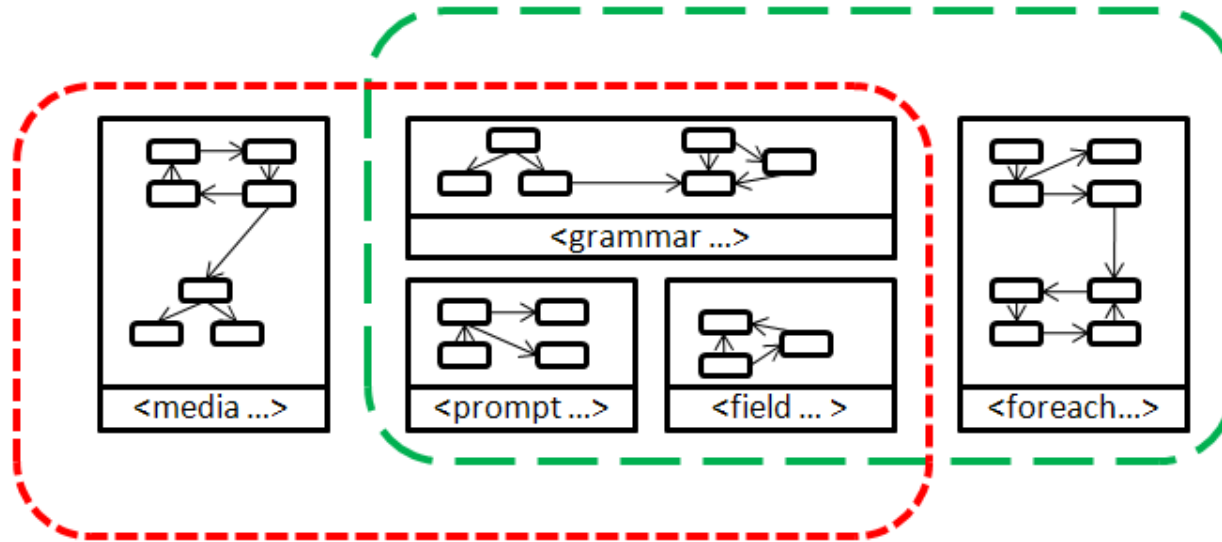
- ▶ As a presentation language
 - The Purpose of VoiceXML 3: To interact with a person by voice (and perhaps video)
 - NOT the Purpose of VoiceXML 3: To be an efficient and effective data repository
 - NOT the Purpose of VoiceXML 3: To efficiently describe business logic
- ▶ However,
 - Voice interaction requires some flow control and some data
 - So limited support for both is still included

- ▶ Core functionality defined in modules
- ▶ Modules combined with convenience syntax into profiles



- ▶ Module behavior defined precisely as state machines

Modules + Conv. Syntax = Profiles



- ▶ Modules grouped into profiles
- ▶ Legacy (V2.1), Basic, Maximal
- ▶ Convenience syntax simplifies authoring

- ▶ New elements and attributes, but no new functionality
- ▶ Behavior defined in terms of core functionality
- ▶ For example, `<menu>` defined in terms of `<form>` with grammars and prompts

Convenience syntax example



Note: Work in progress!!!!

```
<!-- convenience syntax version -->
```

```
<vxml ...>
```

```
<menu>
```

```
<prompt> Say one of: <enumerate/> </prompt>
```

```
<choice next="#operator"> operator </choice>
```

```
<choice next="http://sports.example.com/start.vxml"> sports </choice>
```

```
<choice next="http://weather.example.com/intro.vxml"> weather </choice>
```

```
</menu>
```

```
...
```

```
</vxml>
```

Convenience syntax example (cont.)



Note: Work in progress!!!!

```
<!-- previous code behaves as if it were this -->
<vxml ...>
  <form>
    <link next="#operator"> <grammar ...> <rule ...> operator </rule> </grammar> </link>
    <link next="http://sports.example.com/start.vxml">
      <grammar ...> <rule ...> sports </rule> </grammar>
    </link>
    <link next="http://weather.example.com/intro.vxml">
      <grammar ...> <rule ...> weather </rule> </grammar>
    </link>

    <field name="main">
      <prompt> Say one of: operator, sports, weather .</prompt>
    </field>
  </form>
  ...
</vxml>
```

- ▶ **Definite candidates are**
 - menu/choice/enumerate/option
 - error/help/noinput/nomatch shortcuts
 - link
- ▶ **Possible (but different) candidates might be**
 - if/else/elseif (using SCXML)
 - transfer (using CCXML)

- ▶ Author-specifiable transition controllers
- ▶ V2 eventing model now async & compatible with DOM Level 3
- ▶ New media, SIV functions
- ▶ Session root documents
- ▶ Real-time controls

- ▶ Inter-element transitions now under author control
- ▶ Controllers at form, document, application, and perhaps session levels
 - e.g. form controller specifies which form item to execute next
- ▶ Controllers can be in SCXML or another flow control language
- ▶ Default controllers will give FIA behavior in Legacy Profile

Transition controllers example 1



Note: Work in progress!!!!

```
<!-- document-level transition controller controls inter-form transitions -->
<vxml ...>
  <controller ...>
    <scxml:scxml version="1.0" ...>
      <!-- SCXML code determining which form to go to next -->
    </scxml>
  </controller>

  <form id="form_a" >
    ...
    <goto next="form_b"/>      <!-- goto is only a suggestion now -->
  </form>

  <form id="form_b" >
    ...
  </form>

  ...
</vxml>
```

Transition controllers example 2



Note: Work in progress!!!!

```
<!-- form-level transition controller controls inter-field transitions -->  
<vxml ...>  
  <form>  
    <controller src= "myformbehavior.scxml">  
  
    <field name="field_a" > ... </field>  
    <field name="field_b" > ... </field>  
    <field name="field_c" > ... </field>  
    <field name="field_d" > ... </field>  
  </form>  
  ...  
</vxml>
```

- ▶ V2 event handlers: defined in XML code in the document tree; complex event handling (e.g., prompt counters)
- ▶ HTML event handlers: typically ECMAScript, can be attached to any element in the tree, DOM 3 event handling
- ▶ V3 events & handling: same as V2, but
 - event handling now defined as specific use case of DOM 3 Events -- may allow multi-markup documents
 - underlying event model now asynchronous

- ▶ Video -- `<audio>` replaced by `<media>`, which allows both audio and video

```
<media type="audio/x-wav" src="http://www.example.com/resource.wav"/>
```

```
<media type="video/3gpp" src="http://www.example.com/resource.3gp"/>
```

```
<media>      <!-- inline SSML with audio media fallback-->
  <speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis">
    Ich bin ein Berliner.
  </speak>
  <media type="audio/x-wav" src="ichbineinberliner.wav">
</media>
```

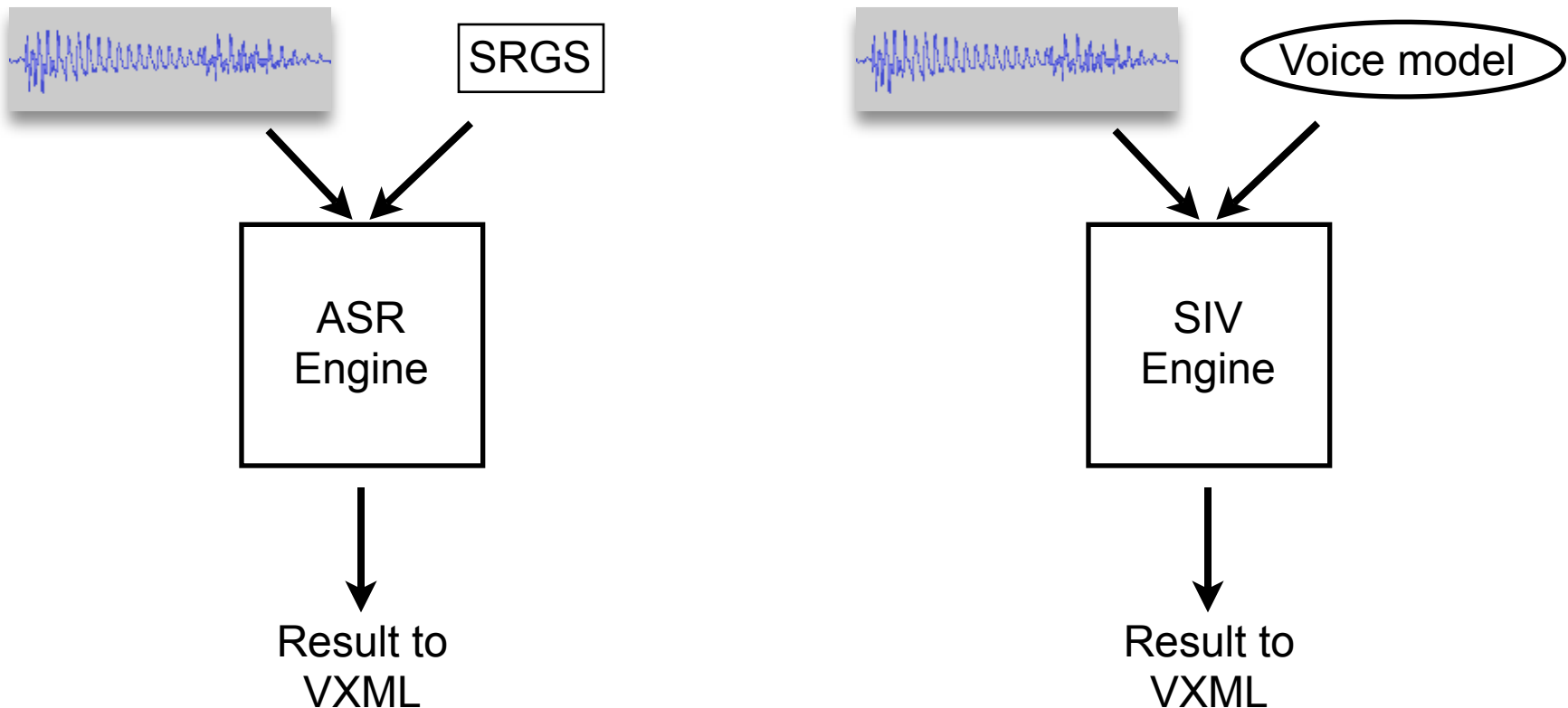
- ▶ Media control -- media clipping, speed, and volume control now possible without resorting to SSML

```
<media type="audio/x-wav" soundLevel="+6.0dB" speed="50%" repeatcount="2"  
src="http://www.example.com/resource.wav"/>
```

```
<media type="video/3gpp" clipBegin="2s" clipEnd="5s" repeatDur="25s"  
src="http://www.example.com/resource.3gp"/>
```

- ▶ SIV -- speaker authentication capabilities available as core functionality
 - Enrollment -- creates voice model, associates it with id in speaker database
 - Identification -- which voice model in speaker database is a match for the speech?
 - Verification -- for the claimed id, does the speech match the voice model in the speaker database?

- ▶ Verification process similar to speech recognition



- ▶ SIV process different from speech recognition?
 - ASR -- single turn, possibly with repeats
 - SIV -- multi-turn state info must be maintained

 - Specification of SIV speaker database
 - introduces different kinds of errors

- ▶ Lots of stuff to work out

- ▶ Just like application root

```
<vxml session="blahblah.vxml" ...>
```

- ▶ Well, not exactly
 - If not specified, no session root
 - Session root change is ignored or causes error
- ▶ First, let's review application roots

A: <vxml>

AppRoot A

B: <vxml>

AppRoot B

C: <vxml root="B">

AppRoot B

D: <vxml root="E">

AppRoot E

F: <vxml root="E">

AppRoot E

G: <vxml>

AppRoot G

A: <vxml>	No Session Root
B: <vxml session="C">	Session Root C
D: <vxml>	Session Root C
E: <vxml session="F" >	Session Root C
G: <vxml session="H" requiresession="true">	error.badfetch

- ▶ Special grammars that are always active (not just in the wait state)
 - Allows arbitrary speech/dtmf
 - Immediate: volume, speed, skip
 - At next event processing: cancel, goto

```
<form>
  <rtc grammar="digit3.grxml" action="volume" params="+5"/>
  <field name="a"> ... </field>
  <field name="b">
    <cancelrtc grammar="digit3.grxml"/>
    ...
  </field>
</form>
```

Note: Work in progress!!!!

- ▶ Acts as pre-filter on input stream, replacing matches with silence

- ▶ Loose integration, e.g.
 - SCXML as controller (separate document/process)
 - send/receive events
 - optional MMI lifecycle events
- ▶ Tight integration, e.g.
 - HTML as controller (same document/process)
 - Direct DOM event handlers attached to both V3 and HTML elements for control

```
<!-- BankApp.vxml -->
<vxml version="3.0">
  <form id="getAccountNum">
    <field name="accountNum">
      <grammar src="accountNum.grxml" type="application/grammar+xml"/>
      <prompt>
        Please tell me your account number.
      </prompt>
      <filled>
        <exit namelist="accountNum"/>
      </filled>
    </field>
  </form>
  ...
</vxml>
```

```
<!-- bankapp.scxml -->
<scxml ...>
  <state id="getAccountNum">
    <invoke targettype="vxml3" src="BankApp.vxml#getAccountNum" />
    <transition event="vxml3.gotAccountNum" target="getBalance"/>
  </state>

  <state id="getBalance">
    <datamodel>
      <data name="method" expr="getBalance"/>
      <data name="accountNum" expr="_data.accountNum"/>
    </datamodel>
    <send targettype="basichttp" target="BankDB.do" namelist="method accountNum" />
    <transition event="basichttp.gotBalance" target="playBalance"/>
  </state>

  <state id="playBalance"> ... </state>
  <final id="exit"/>
</scxml>
```

```
<!-- Note: code dreamed up in my head -- never reviewed by W3C -->  
<html>  
  ...  
  vxmlAccountNumElement.addEventListener('done',doSomething,false);  
  function doSomething() {  
    this.style.backgroundColor = '#cc0000';  
  }  
  ...  
  <vxml> <form>  
    <field name="accountNum">  
      <grammar src="accountNum.grxml" type="application/grammar+xml"/>  
      <prompt>  
        Please tell me your account number.  
      </prompt>  
      <filled>  
        <throw event="done"/>  
      </filled>  
    </field>  
  </form> </vxml>
```

- ▶ Follow the work
 - <http://www.w3.org/Voice>
- ▶ Contact me
 - dburnett at voxeo dot com

Dan Burnett, Ph.D.
Dir. of Speech Technologies, Voxeo
Developer Jam Session
May 20, 2010



**REGISTER TODAY TO RESERVE YOUR
ALL ACCESS PASS FOR VOXEO'S CUSTOMER SUMMIT
JUNE 21-23 IN THE HARD ROCK HOTEL AT
UNIVERSAL ORLANDO**

**SPACE IS LIMITED.
SAVE YOUR SEAT TODAY!**



WWW.VOXEO.COM/SUMMIT2010

**UNIFIED SELF-SERVICE:
CREATING MULTI-CHANNEL COMMUNICATIONS
APPS USING VOXEO TOOLS**

- ▶ Speaker: **Adam Kalsey**
 - Manager, Voxeo Development Network
- ▶ Date: **June 17, 2010**
- ▶ Time: **8 am PDT, 11 am EDT, 5 pm EST**

