Speech Enabling Web Browsers

Dave Raggett <dsr@w3.org>

AVIOS/SpeechTek West, January 2006

with thanks to

W3C® WORLD WIDE WEB consortium

Loquendo VOCAL TECHNOLOGY AND SERVICES

Canon

The Ubiquitous Web

- The Web is an increasingly dominant applications platform
- Applications that dynamically adapt to the user, device and environment
- Moving beyond today's client/server model
- Extending device capabilities via the network
- W3C UbiWeb Workshop, Tokyo March 2006

Options for adding speech capabilities

- Handling speech modality in the network
  - Loose coupling of modality interfaces
    - e.g. XHTML locally with VoiceXML in the network, with CCXML for high level flow control
- Handling speech modality in the browser
  - Embedded vs networked speech
    - latency, quality, vocabulary, network, battery, ...
  - Plugin vs local speech proxy
  - Standard scripting interface?

Latency

- Simple commands with visual actions
  - up, down, select, ...
    - Feels slow if delay is much greater than 100mS
- Dialogue turn hand over
  - When user stops talking (or pauses)
  - When application stops talking (or pauses)
- Seizing the turn
  - User or application talks over the other party
- Network delays are not as bad as they seem

Using AJAX to add speech

- AJAX = JavaScript for accessing HTTP
  - XMLHTTP object
  - Supported by modern web browsers
- Local HTTP server handles device audio
- Remote HTTP server for speech services
  - ASR with audio in HTTP request, and EMMA in HTTP response
  - TTS with text or SSML in HTTP request, and audio in HTTP response

HTTP for Speech Services

- Speech synthesis
  - http://localhost:8888/say?text=”good afternoon”
  - http://localhost:8888/say?uri=<*ssml file*>
- Speech recognition
  - http://localhost:8888/hear?uri=<*srgs file*>
  - Additional parameters for
    - Listening on multiple grammars
    - Single result vs sequence of results
    - Time out parameters

Application to Pizza ordering

Speech libraries courtesy of Loquendo

SRGS + SISR → EMMA

- Use W3C Recommendations for speech grammars and semantic interpretation

```
<rule id="order">
   <tag>var index=0; out.pizza = new Array();</tag>
   <item repeat="0-1"><ruleref uri="#start"/></item>
   <item>
     <ruleref uri="#pizza"/>
     <tag>out.pizza[index]=$pizza; index+=1;</tag>
   </item>
   <item repeat="0-">
      <item><token>and</token></item>
      <item>
        <ruleref uri="#pizza"/>
        <tag>out.pizza[index]=$pizza; index+=1;</tag>
      </item>
   </item>
   <item repeat="0-1"><ruleref uri="#stop"/></item>
</rule>
```

Pizza Grammar

I would like four small cheese pizzas with olives and peppers.

```
[<start>] [<number>] [<size>] <type> (pizza | pizzas) [with <extras>] [<stop>]

<start> ::= I want | I would like | I'll have | I'd like | I'd love | Give me
<stop> :: thanks | please | if you please
<number> ::= a | one | two | ... | nine
<size> ::= small | medium | large
<type> ::= cheese | pepperoni | sausage
<extras> ::= <topping> [[and] <topping>]*
<topping> ::= mushroom | olives | onions | peppers | tomatoes

<emma:interpretation>
  <pizza>
     <size>small</size>
     <number>4</number>
     <type>cheese</type>
     <topping>olives</topping>
     <topping>peppers</topping>
  </pizza>
</emma:interpretation>
```

Pizza Grammar

A slightly more complex grammar allows for several kinds of pizza to be requested at once

Give me a medium pepperoni pizza and a large cheese pizza with peppers and onions.

```
<emma:interpretation>
   <pizza>
      <number>1</number>
      <size>medium</size>
      <type>pepperoni</type>
   </pizza>
   <pizza>
      <number>1</number>
      <size>large</size>
      <type>cheese</type>
      <topping>sausage</topping>
      <topping>onions</topping>
   </pizza>
</emma:interpretation>
```

Application to Pizza ordering

- Implemented in XHTML+CSS+JavaScript
- Supports compound utterances
  - Faster than filling out forms via GUI
  - But requires flexible dialogue to work around inevitable misunderstandings
- DIY solution for describing behavior
  - Combination of scripting and markup
  - Markup interpreted via JavaScript
    - Browser independent

Modeling Behavior

- Scripted handlers for XHTML events, e.g. onload, onmouseover, onfocus, onchange
- Asynchronous callbacks for HTTP responses
  - Used to handle results of speech recognition
  - Initiated via calls to XMLHTTP request
- Asynchronous timers (setTimeout)
- Use of custom markup
  - Application state, dialogue goals and history
  - Event driven state transition rules

Logging

- Usability is based upon real world experience
  - That means you need to collect lots of data
- Log dialogues and audio for later analysis
  - Speech server log's ASR, TTS requests
  - AJAX used for logging dialogue state
    - Including changes via visual modality
  - Application assigned session identifier
    - Used to associate log entries for same session
    - Must be sent as part of all server requests

Final Thoughts

- Complex utterances are more natural but require a more flexible approach for effective dialogues
- Exposing speech to Web pages via JavaScript offers flexibility for rolling your own solutions whilst remaining inter-operable across browsers
- There is an opportunity for a standard speech object that abstracts away from embedded vs networked speech