11.4 Voice Browser & Multimodal Interaction

Dr. Max Froumentin, W3C

Outline
- New devices allow web access
- But they aren’t phones
- So far most new devices adopt themselves to the visual-page-point-click paradigm
- But not the web
- The web that is going to change:
  - New behavior in (effect of)
  - New expectations

New devices (2)

In more detail
- New output devices: Small screens, speech synthesizers, fragile devices
- New input devices: Electronic pens (handwriting), speech recognition, gestures
- New environments: Low-bandwidth networks, PCs, smartphones, tablets
- The 3 items above are all dynamic
- Web pages are becoming Web applications

New devices

So what?
Each new device manufactures its own browser to suit existing Web content.
- Smart browsers
- Source-side adaptation
Complex example

This works so far but let’s look at something more complex: “google maps in my car.”

Google maps in my car. I want to have my car navigation system use google maps.

Requirements:
- I want to be able to touch screen to manipulate map (pin, label the street)
- I want to speak to it, and be able to ask it to repeat, or move to next item or previous in itinerary,
- I want it to show me the screen when I’m stopped but to talk to me when I drive
- Use my car’s GPS obviously, get traffic info, weather, etc.

Multimodal Interaction on the Web

- New browsers that know what modalities they are connected to.
- New browser-server communication
What to standardise?

- The browser
- Web content

But why standardise things that are happening inside the browser? Why change the Web?

---

The MMI framework

- the boxes don't necessarily map to devices
- Reuse of existing markup: XHTML, CSS, SVG for output, SRGS for input

---

Input Modality Interfaces

For input we need: Grammars, Integration, Interface

Interfaces as IDL/WSDL APIs can be used in JavaScript directly, or Web Services, respectively

Grammar, feature, setGrammar, setModel, metadata, context, invoke events

---

Leading to two types of architectures:

What to transmit through the pipes

- Website to Browser: HTML, XBL, CSS, VoiceXML (interaction management), PNG, SML, SVG, CSS (output)
- Website to Browser to Recognition Grammars, from information
- Recogniser to Browser to Web sites recognition results
Defining handwritten gestures and grammar: InkML

```
<ink>
  <text>
    31 6 9 14 8 38 7 42 6 56 6 70 8 48 8 98 9 131 184 145 31 154 14 146 17 132 18 189 23 197 30 160 38 147 45 139 50 124 10 121 177 150 0 269 82 165 40 177 47 131 51 265 
  </text>
</ink>
```

```
210 188 234 262 235 176 238 150 241 204
</text>
</ink>
```

```
<ink>
  <text>
    31 139 353 143 154 157 349 171 152 185 355 127 311 294 385 260 399 202 408 151 433 177 643 183 415 153 351 143 179 244 345 150
  </text>
</ink>
```

```
EMMA: representing and annotating input

Goals:
- define a markup language to combine user input information from recogniser to the interaction manager
- merge a post interface to recogniser output

EMMA: example
```xml
<emma:emma
  emma:version="1.0"
  emma:xmlns="http://www.w3.org/2001/04/emma">
  <emma:one-of
    emma:id="10"
    emma:confidence="0.01"/>
  <emma:interpretation
    emma:id="102"
    emma:confidence="0.49"/>
  <emma:mention
    origin="/the/origin"/>
  <emma:mention
    emma:begin="03:15:05/s"/>
  <emma:mention
    emma:end="03:15:10/s"/>
</emma:emma>
```
Compositing

Making one emma live out of two

- programming language (e.g. Java+DOM)
- XML

- Declarative: SML
-

The Dynamic Properties Framework

The DPF specification defines an API to access system properties. E.g.
- battery level
- signal strength
- latitude/longitude from GPS
- ambient noise
- user preferences

DPF: example

```
<html>
<head>
<title>GPS location example</title>
<script type="text/javascript">
<![CDATA[
    SystemEnvironment.location.format="zip code";
    SystemEnvironment.location.updateFrequency="29s";
]]>
</script>

<body>
<p>Your current zip code is: <span id="location">(please wait)</span></p>
</body>
</html>
```

Interaction Manager

The manager...
- raises the web page or web application
- knows what modalities are available
- gets information from the DPF and Session components

Interaction Manager (2)

... and shapes the interaction accordingly.
- with visual media: shows the page on the screen.
- with audio media: presents an application as a dialogue (à la VoiceXML)

Could be coded JavaScript using the APIs mentioned above, or declarative interaction markup like VoiceXML, with a mapping to HTML+CSS+JavaScript.

Writing multimodal web content

Existing web pages and applications will still work but won't provide:
- modality dependent content
- modality dependent interaction

So extensions will be useful:
- You can already do it in HTML+JavaScript+CSS+DAP (see above)
- Declarative markup (better): VoiceXML, SCXML, CCXML
Summary 1

The Voice Browser Activity

Historically preceded the MMI Framework
A specific framework
Now integrates into MMI

Summary 2

VoiceXML

VoiceXML, one of VSI’s most successful specifications
Simple form-filling applications on the phone:

CCXML

A standard for telephony platforms
Handles events such as incoming calls
Makes outgoing calls, conference calls, start VoiceXML dialogues
**CCXML System Architecture**

**SCXML**

Hidden State Tables: a general interaction management paradigm

CCXML provides markup for HST

Can plug in to CCXML, or drive VoiceXML dialogs or MMI interaction

**Voice+MMI**

How does the Voice Interface Framework specifications fit into the MMI architecture?

- VoiceXML CCXML and SCXML provide interaction manager and telephony capabilities
- But are mainly focused on voice applications with telephony contexts
- They integrate in the MMI framework and add new components

**Summary**

New multimodal devices can make a better Web experience

The MMI Framework generalises the standard visual browser model

The MMI Framework generalises the standard voice browser model

New specifications needed for:
- Recognition
- Interaction
- New components: environment, events, integration, etc.