Standard Languages for Developing Multimodal Applications

James A. Larson

Intel Corporation
16055 SW Walker Rd, #402, Beaverton, OR 97006 USA
jim@larson-tech.com

Abstract

The World Wide Web Consortium (W3C) is standardizing languages for developing multimodal applications. Two specifications, SALT and XHTML + Voice, have been submitted to the W3C for developing web-based multimodal applications. The W3C has published the Extended Multimodal Application (EMMA) language for representing the semantics of information entered via keyboard, mouse, microphone, or stylus by the user.

1 Motivation for standard languages

The World Wide Web Consortium[1] (W3C), founded in 1994, has been active in developing languages for Web-based application development. Two of its working groups, Voice Browser and Multimodal Interaction, are actively working towards standardizing languages for developing speech and multimodal applications.

Standards are a double-edged sword. Standard languages hide many of the underlying technological details of how recognition systems work. They save developer time and effort by reusing existing language processors rather than implementing processors from scratch for new languages. Also, developers and researchers may swap modules with one another. In general, standard languages promote reusability and portability.

On the other hand, if the standard languages do not support the functions the researcher or developer needs, standard languages may be difficult to extend. This may limit the flexibility of the resulting prototype or application. Sometimes, standard languages stifle creativity.

2 Tour the W3C multimodal framework

Figure 1 below is a high-level representation of the W3C Multimodal Interaction Framework.[2] Figure 1 illustrates how the Interaction Manager accepts input from the user via one or more input components such as typing, pointing, speaking, and so forth. The Interaction Manager may invoke application specific functions and access information in a Dynamic Processing module. Finally, the Interaction Manager causes the result to be presented to the user via one or more output components such as audio or a display screen.

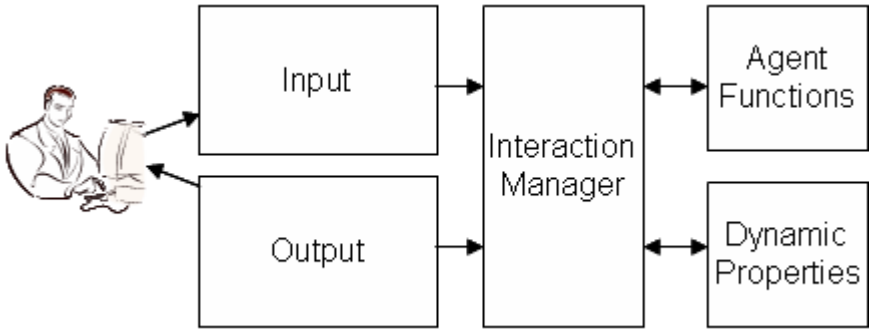

Figure 1. W3C Multimodal Interaction Framework

3 Interaction manager

The interaction manager coordinates data and manages execution flow among various input and output components. The interaction manager responds to inputs from the input components, updates the interaction state and context of the application, and initiates output to one or more output components.

Developers use several approaches to implement interaction managers, including:

- Traditional programming languages such as C or C++
- Speech Application Language Tags[3] (SALT) which extends HTML by adding a handful of HTML tags to support speech recognition, speech synthesis, audio file replay, and audio capture
- XHTML plus Voice[4] (often referred to simply as "X+V"), in which the VoiceXML 2.0 voice dialog control language is partitioned into modules that are embedded into HTML
- Formal specification techniques such as state transition diagrams and Harel state charts

3.1 Traditional programming languages

Many of the early multimodal applications were implemented using traditional programming languages, such as C or C++. Developers used these languages to controll the application flow and invoking low-level speech recognition, handwriting recognition, speech synthesizer, audio capture, and replay components. The application contains a mixture of:

- execution flow—the sequence of actions which the system performs
- interaction flow—how the user and the system take turns supplying each other with information
- input/out control—control of speech and handwriting recognizers, speech synthesizers, and other input and output components

Industrial consortiums have created languages to specify these very different functions, modularizing the applications, increasing the reusability of modules across multiple applications. These languages include SALT and X+V.

3.2 SALT

Speech Application Language Tags (SALT) is a small number of XML elements that may be embedded into a host language such as XHTML. SALT has no control structures itself, so it must be embedded into a host language which specifies the execution and interaction flow, while SALT controls the input/output control of speech recognition and synthesis engines. SALT is used to develop telephony (speech input and output only) applications and multimodal applications (speech input and output, as well as keyboard and mouse input and display output).

The SALT Forum,[5] originally consisting of Cisco, Comverse, Intel, Microsoft, Philips, and SpeechWorks (now ScanSoft), published the initial specification in June 2002. This specification was contributed to the World Wide Web Consortium (W3C) in August of that year. Later in June 2003, the SALT Forum contributed a SALT profile for Scalar Vector Graphics (SVG) to the W3C.

The SALT specification contains a small number of XML elements enabling speech input from the user, called prompts, and speech output to the user, called responses. SALT elements include:

- <*prompt*>—presents audio recordings and synthesized speech to the user. SALT also contains a prompt queue and commands for managing the presentation of queued prompts to the user.
- <*listen*>—recognizes spoken words and phrases spoken by the user. There are three listen modes:
  - *Automatic*—used for recognition in telephony or hands-free scenarios. The speech platform rather than the application controls when to stop the recognition facility.
  - *Single*—used for push-to-talk applications. An explicit stop from the application returns the recognition result.

  - *Multiple*—used for "open-microphone" or dictation applications. Recognition results are returned at intervals until the application makes an explicit stop.
- <*grammar*>—specifies the words and phrases a user might speak
- <*dtmf*>—recognizes DTMF (telephone touchtones)
- <*record*>—captures spoken speech, music, and other sounds
- <*bind*>—integrates recognized words and phrases with application logic
- <*smex*>—communicates with other platform components

SALT has no control elements, such as <*for*> or <*goto*>, so developers embed SALT elements into other languages called host languages. For example, SALT elements may be embedded into languages such XHTML, SVG, and JavaScript. Developers use the host language to specify application functions and execution control while the SALT elements provide advanced input and output using speech recognition and speech synthesis. The following example illustrates SALT elements embedded into XHTML. This application asks the user how many pizzas the user wants.

```
<body onload="askPizzaQuantity.start();">

<input type= "text" id="pizzaQuantity"  />

<salt:prompt  id="askPizzaQuantity" onComplete = "lsnPizzaQuantity.start();">
    How many pizzas would you like?
</salt:prompt>

<salt:listen id="lsnPizzaQuantity">
  <salt:grammar scr = "onetotwenty.jsgf">
  <salt:bind targetelement = "pizzaQuantity" value="//"/>
</salt:listen>
```

When the application is loaded, the input text box with id *pizzaQuantity* is displayed and the user is prompted with the verbal message, "How many pizzas would you like?" The user may respond by typing a number into the input text box with id *pizzaQuantity* or by speaking a digit between one and twenty. (We assume that no one will order more than twenty pizzas at the same time.) The <*bind*> element is used to place the value recognized by the speech recognition engine into the input text box with id *pizzaQuantity*.

The SALT forum has submitted the SALT specification to the W3C for use in future standardized languages for both speech and multimodal applications.

3.3 X+V

Like SALT, developers use X+V to insert voice elements into XHTML. However, rather than inventing new tags, X+V reuses the existing tags from VoiceXML 2.0, an established W3C language for developing telephony applications in which the user speaks and listens to a computer via a telephone or cell phone. X+V partitions the VoiceXML language into multiple modules, including modules for speech synthesis, grammars, and dialogs. Input obtained from a speaking user is integrated with information entered by the user via a keyboard and mouse by a <*sync*> tag that binds speech data to XHTML variables.

While SALT relies on the host language to provide the execution flow of the application, X+V can use a VoiceXML form (which has its own interaction flow in the form of a Forms Interpretation Algorithm) as well as the execution flow features of the host language.

Developers use X+V to develop multimodal applications targeted for the desktop (with its large display screen) and advanced cell phones and PDAs with small screens. While theoretically the same application can be executed in both environments, the different screen size requires different visual user interfaces; developers usually create different user interfaces for the same application running in each of these environments. Developers use VoiceXML 2.0, the language on which X+V is based, to specify telephony speech applications.

Here is the example from above written in X+V:

```
<body onload = "document.getElementBuild("pizzaQuantity").focus()">

<input type = "test" id = "pizzaQuantity" ev:event = "focus" evhandler = "#voice_quantity"/>

<vxml:form id = "voice_quantity">
    <vxml: field name = "vquantity">
        <vmxl:prompt>
            How many pizzas would you like?
        </vxml:prompt>
        <vxml:grammar src = "onetotwenty.jsgf" />
       <vxml:filled>
            <vxml:assign name = "document.getElementByld("pizzaQuantity").value" expr = "vquantity"/>
      </vxml:filled>
    </vxml: field>
</vxml: form>
```

When the HTML file is loaded, focus will be set to the *pizzaQuantity* input text box. When the *pizzaQuantity* text box gets the focus, the *voice_quantity* VoiceXML form is invoked, which asks the user "How many pizzas to you want?" The user responds by either speaking the number or typing the number into the *quantityPizza* input text box. The <assign> element copies the value spoken by the user into the *pizzaQuantity* text box.

IBM, Motorola, and Opera have submitted the X+V specification to the W3C for use in future development languages for both speech and multimodal applications.

3.4 Harel state charts

Both SALT and X+V enable developers to write multimodal applications that enable the user to both speak and listen as well as read and type. Developers use the host language to specify execution flow (the sequence of actions which the system performs) and interaction flow (how the user and the system take turns supplying each other with information). The interaction between the system and user can be represented using several techniques, including state transition systems, production rules, and Harel state charts.

Like state transition systems, developers can use Harel state charts[6] to specify execution and interaction flow. Unlike traditional state transition systems, Harel state charts can specify parallel input activities, such as speaking and drawing/pointing at the same time.

The W3C is developing a new XML language based on Harel state charts (frequently used in traditional application design documents) so developers can use this model to specify the execution flow of multimodal applications.

4 Input components

An interactive multimodal implementation will use multiple input modes such as audio, speech, handwriting, and keyboarding, and other input modes. User input may be either decoded or recognized:

- *Decoded input* includes keyboard, mouse, and joystick input. The event generated by pressing a button or repositioning a mouse are decoded into character strings or screen coordinates.
- *Recognized input* includes speech, handwriting, and vision. Special speech, handwriting and vision recognition subsystems are required to convert the user input into character strings.

All recognition systems occasionally make mistakes. In the best situations, speech recognition systems will make recognition errors 3−5 percent of the time. The user interface compensates for recognition errors by (1) prompting

the user with specific questions and (2) listening only for prespecified input that may be spoken or written by the user. The collection of words that can be recognized by a recognition system is called a *grammar*. Using grammars to guide recognition systems is beneficial because the grammar greatly improves the accuracy of recognition systems. Small grammars also enable the speech recognition system to work faster, which minimizes response delays. The W3C Speech Recognition Grammar Specification (SRGS)[7] is an XML language that is widely used in speech recognition applications, and is a leading candidate for use by other recognition techniques. The SRGS syntax for an example grammar looks like:

```
<grammar type="application/grammar+xml" version="1.0"  xml:lang = "en" root="command">
   <rule id = "command">
      <item>send </item>
      <item> this </item>
      <item repeat = "0-1"> file </item>
       item> to </item>
      <one-of>
         <item> this </item>
         <item> here </item>
       </one-of>
   </rule>
</grammar>
```

The above grammar listens for any of the following user utterances:

Send this file to this
Send this file to here
Send this to this
Send this to here

where the word “file” is optional, and the user may say either “this” or “here” for the final word of the phrase.

While each recognition system uses grammars to produce text based upon the user’s input, the text from various recognizers may produce different formats making integration difficult. The W3C has established a standard language for representing input—Extended MultiModal Annotation (EMMA).[8] Like SRGS, EMMA is an XML language that contains a textual representation of the user input as well as various attributes and qualifiers that assist the integration module to combine inputs from multiple sources into a unified input for the interaction manager. Developers may also use the W3C Semantic Interpretation Language[9] to instruct recognition engines and input modules how to convert raw text to the appropriate EMMA format. The Semantic Interpretation Language is based on ECMAScript with some extensions that tie ECMAScript variables to grammar constructs.

The semantic representation assigned to the spoken input "send this file to this" indicates two locations where content is required from ink input using *emma:hook="ink":*

```
<emma:emma version="1.0" xmlns:emma="http://www.w3.org/2003/04/emma">
   <emma:interpretation>

      <command>
         <action>send</action>
         <arg1>
            <object emma:hook="ink">
               <type>file</type>
               <number>1</number>
            </object>
         </arg1>
         <arg2>
            <object emma:hook="ink">
```

```
            <number>1</number>
          </object>
        </arg2>
      </command>
    </emma:interpretation>
  </emma:emma>
```

The user gesturing on the two locations on the display can be represented using <*emma:sequence*>:

```
<emma:emma version="1.0" xmlns:emma="http://www.w3.org/2003/04/emma">
  <emma:sequence>
    <emma:interpretation emma:mode="ink">
      <object>
        <type>file</type>
        <number>1</number>
        <id>test.pdf</id>
      </object>
    </emma:interpretation>
    <emma:interpretation emma:mode="ink">
      <object>
        <type>printer</type>
        <number>1</number>
        <id>lpt1</id>
      </object>
    </emma:interpretation>
  </emma:sequence>
</emma:emma>
```

Many different techniques, including unification, could be used for integrating the semantic interpretation of the pen and spoken input. A general purpose unification-based multimodal integration algorithm could use the *emma:hook* annotation as follows. It identifies the elements marked with *emma:hook* in document order. For each of those in turn, it attempts to unify the element with the corresponding element in order in the <*emma:sequence*>. Since none of the subelements conflict, the unification goes through and, as a result, we have the following EMMA for the composite result:

```
<emma:emma version="1.0" xmlns:emma="http://www.w3.org/2003/04/emma">
  <emma:interpretation>
    <command>
      <action>send</action>
      <arg1>
        <object>
          <type>file</type>
          <number>1</number>
          <id>test.pdf</id>
        </object>
      </arg1>
      <arg2>
        <object>
          <type>printer</type>
          <number>1</number>
          <id>lpt1</id>
        </object>
      </arg2>
    </command>
  </emma:interpretation>
```

```
</emma:emma>
```

The EMMA is transferred to the interaction manager, which launches a print operation..

5 Output components

Standard W3C languages used to present information to user include the following:

- *Extended Hypertext Markup Language (XHTML)*[10]—an XML version of HTML for presenting visual information on screens
- *Speech Synthesis Markup Language (SSML)*[11]—an XML-based language used to render text as speech
- *Scalar Vector Graphics 1.2 (SVG)*[12]—an XML-based language for writing two-dimensional vector and mixed vector/raster graphics
- *Synchronized Multimedia Integration Language 2.0 (SMIL)*[13]—an XML-based language for writing interactive multimedia presentations

6 Dynamic Properties Framework

The W3C Dynamic Properties[14] module enables the interaction manager to find out about and respond to changes in device capabilities, user preferences, and environmental conditions. For example,

- Which output modes are available?
- Has the user muted the audio input?
- What is the width and height of the display screen?
- Does the display screen uses monochrome or does it support multiple colors?
- How much battery power is available?

The application may dynamically reconfigure itself to take advantage of the capabilities and restrict itself to the current constraints of the device on which it is executing.

7 Summary

The W3C Voice Browser and Multimodal Interaction Working Groups want to learn about your experiences using the languages during the standardization process. Developer input has an impact on the final form for these languages. Send comments to the public mailing lists (www-voice@w3.org or www.mmi@w3.org).

---

[1] http://www.w3.org/
[2] http://www.w3.org/TR/mmi-framework/
[3] http://www.saltforum.org/#Downloads
[4] http://www-306.ibm.com/software/pervasive/multimodal/x%2Bv/11/spec.htm
[5] http://www.saltforum.org/
[6] http://www.wisdom.weizmann.ac.il/~dharel/papers/Statecharts.pdf
[7] http://www.w3.org/TR/2004/REC-speech-grammar-20040316/
[8] http://www.w3.org/TR/emma/
[9] http://www.w3.org/TR/semantic-interpretation/
[10] http://www.w3.org/TR/2004/WD-xhtml2-20040722/
[11] http://www.w3.org/TR/2004/REC-speech-synthesis-20040907/
[12] http://www.w3.org/TR/SVG12/
[13] http://www.w3.org/TR/2005/REC-SMIL2-20050107/
[14] http://www.w3.org/TR/DPF/