VOICEXML 2.0 AND THE W3C SPEECH INTERFACE FRAMEWORK

Jim A. Larson
Intel Corporation
jim.a.larson@intel.com

1.1. ABSTRACT

The World Wide Web Voice Browser Working Group has released specifications for four integrated languages to developing speech applications: VoiceXML 2.0, Speech Synthesis Markup Language, Speech Recognition Grammar Markup Language, and Semantic Interpretation. These languages enable developers to quickly specify conversational speech Web applications that can be accessed by any telephone or cell phone. The speech recognition and natural language communities are welcome to use these specifications and their implementations as they become available, as well as comment on the direction and details of these evolving specifications.

1. BACKGROUND

Since the release of the VoiceXML 1.0 specifications by the VoiceXML Forum in July 2000, developers have implemented and deployed dozens of VoiceXML browsers. VoiceXML 1.0 has enabled application developers to create and deploy thousands of conversational speech applications.

In May 199,9 the World Wide Web Consortium (W3C) chartered the Voice Browser Working Group (VBWG) to prepare and review markup languages that enable voice browsers. Members meet weekly via telephone and quarterly in face-to-face meetings. The VBWG is open to any member of the W3C Consortium. VBWG members include the founding companies of the VoiceXML Forum, telephony applications venders, speech recognition and text-to-speech engine venders, Web portal companies, hardware venders, software venders, and appliance manufacturers.

2. W3C SPEECH INTERFACE FRAMEWORK

The VBWG has recently released draft specifications for four new languages making up the W3C Speech Interface Framework. The VBWG has clarified the syntax and semantics of VoiceXML 1.0, called VoiceXML 2.0, and has developed three additional related languages for speech recognition grammars, speech synthesis, and semantic interpretation. All four languages are XML languages that use tags to describe language elements. Figure 1 illustrates an example of how the four languages work together to describe conversations speech interfaces.

```
<field name="recipient">

   <prompt>
      Welcome to the
      <emphasis level = "strong">
           electronic payment system
      </emphasis>
      <break time = "500ms"/>
      Whom do you want to pay?
  </prompt>

   <grammar>
      <rule id= "recipient">
          <one-of>
               <item> Ajax </item>
               <item>
                  <tag> 'Drugstore' </tag> Ajax
               </item>
               <item> Stanley </item>
               <item>
                  <tag> 'Supermarket' </tag> Stanley
               </item>
            </one-of>
         </rule>
   </grammar>

</field>
```

Figure 1. Example VoiceXML code (regular font) with embedded speech synthesis (italics), speech grammar (bold) and semantic interpretation tags (underlined text).

The <prompt> tag contains text that is converted to spoken speech by a speech synthesis engine. The <grammar> tag defines the grammar of words and phrases used by a speech recognition engine to convert speech into text. Together, the <prompt> and

<grammar> tags define a <field> into which a user places a value by speaking. The following briefly describes the four languages illustrated in Figure 1.

3. THE FOUR INTEGRATED LANGUAGES

The following briefly describes each of the four integrated languages of the W3C Speech Interface Framework:

3.1. VoiceXML 2.0

VoiceXML 2.0 specifies the basic structure of conversational dialogs. In Figure 1, VoiceXML tags (shown in regular font) describes a field consisting of a prompt to be read to the user by the speech synthesis engine and a grammar to be used by the speech recognition engine to listen to the user's response. The W3C Voice Browser Working Group examined nearly 300 change requests and made many clarifications and minor enhancements to VoiceXML 1.0. Probably the most significant addition is a <log> tag that enables developers to create and debug messages and collect data for performance analysis. Both VoiceXML 1.0 and 2.0 are able to describe system-directed and mixed initiative conversational dialogs.

3.2. Speech Synthesis Markup Language

The Speech Synthesis Markup Language enables developers to specify instructions to the speech synthesizer about how to pronounce words and phrases. Based on the Java Speech Markup Language (JSML) specification, the Speech Synthesis Markup Language provides tags for specifying the structure (<sentence> and <paragraph>), pronunciation (<sayas> and <phoneme>), poetics (<emphasis>, <break>, and <prosody>), and the use of prerecorded audio files (<audio>). In Figure 1, speech synthesis tags are shown in italics. They illustrate how to emphasize the name of the field and how to insert a pause. Developers use speech synthesis tags when they want to improve the speech synthesizer's default presentation of a prompt.

3.3. Speech Recognition Grammar Specification

The Speech Recognition Grammar Specification enables developers to describe the words and phrases that can be recognized by the speech recognition engine. The syntax of the grammar format has two forms. The form shown in Figure 1 uses XML elements to represent a grammar. The form shown in the bold font in Figure 2 uses an augmented BNF format, which is similar to many existing BNF-like representations commonly used in the field of speech recognition. The two forms are equivalent in the sense that a grammar that is specified in one form may also be specified in the other. Both forms are modeled after the JSpeech Grammar Format.

```
<grammar>
        $recipient = Ajax
                     | Drugstore {'Ajax'}
                     | Stanley
                     | Supermarket {'Stanley'};
</grammar>
```

Figure 2. Example of the ABNF grammar format (bold text) with embedded semantic interpretation tags (underlined text).

3.4. Semantic Interpretation

Semantic Interpretation tags enable developers to compute information returned to an application using grammar rules and tokens matched by the speech recognition engine. As examples, if the speech recognition engine recognizes the phrase "Drugstore," the underlined semantic interpretation tags in Figures 1 and 2 calculate the value to be returned as "Ajax." Likewise, "Stanley" is returned in place of "Supermarket." Semantic interpretation tags can also be used to extract information from a user utterance, perform calculations, and assign values to multiple variables. The Semantic Interpretation tags were designed to be very similar to a subset of the ECMAScript language.

4. FUTURE LANGUAGES

The VBWG has also published requirements and working drafts of other languages within the W3C Speech Interface Framework, including:.

4.1 N-gram Grammar Markup Language

Context-free grammars are widely used in conversational systems to model what the user may say at each specific point in the dialog. However, it is difficult or impossible to write a context-free grammar that can process all the different sentence patterns users speak in spontaneous speech input applications. Context-free grammars are not sufficient for robust speech recognition and understanding tasks or for free-text input applications such as dictation.

In contrast, N-gram language models rely on the likelihood of sequences of words, such as word pairs (in the case of bigrams) or word triples (in the case of trigrams) and are therefore less restrictive. The use of stochastic N-Gram models have a long and successful history in the research community and iare being used in commercial systems, as the market asks for more robust and flexible solutions. N-Gram grammars have the advantage of being able to cover a much larger language than would normally be derived directly from a corpus. Open vocabulary applications are easily supported with N-gram grammars.

N-gram grammars are typically constructed from statistics obtained from a large corpus of text using the co-occurrences of words in the corpus to determine word sequence probabilities. Developers should never need to create an N-gram grammar by hand.

4.2. Natural Language Semantics Markup Language

The Natural Language Semantics Markup Language supports XML semantic representations. This application-independent information includes confidences, the grammar matched by the interpretation, speech recognizer input, and timestamps. For example, suppose the user responds to a system request by saying, “I would like a large Coca Cola and a large pizza with pepperoni and cheese.” The semantic is first converted to XML structure in Figure 3.

```
<result>
  <interpretation>
    <instance>
      <order>
        <pizza>
          <number>1</number>
          <size>large</size>
          <topping>pepperoni</topping>
          <topping>cheese</topping>
        </pizza>
        <drink>
          <number>1</number>
          <size>large</size>
          <liquid>coke</liquid>
        </drink>
      </order>
    </instance>
  <input mode="speech">
    I would like a large coca cola and a large
    pizza with pepperoni and cheese.
  </input>
 </interpretation>
</result>
```

Figure 3. Natural Language Semantic Language example

Additional information, including confidence scores and input mode (speech) is inserted into this structure, resulting in the representation illustrated in Figure 4, which is suitable for additional natural language processing.

```
<result grammar="src='pizza_order.grm' ">
  <interpretation confidence="100" >
    <instance>
     <order>
       <pizza>
         <number>1</number>
         <size>large</size>
         <topping confidence="100">
           pepperoni
         </topping>
        <topping confidence="100">
          cheese
        </topping>
      </pizza>
      <drink confidence="100">
        <number>1</number>
        <size>large</size>
        <liquid>coke</liquid>
      </drink>
    </order>
  </instance>
   <input mode="speech">
     I would like a large coca cola and a large
    pizza with pepperoni and cheese.
   </input>
 </interpretation>
</result>
```

Figure 4. Natural Language Semantic Language example with confidence factors.

4.3 Other Future Languages

Other future languages include:

- Reusable Modules are reusable components that meet specific interface requirements. The purpose of reusable components is to reduce the effort to implement a dialog by reusing encapsulations from common dialog tasks and to promote consistency across applications.

- The pronunciation Lexicon Markup Language enables open, portable specification of pronunciation information for speech recognition and speech synthesis engines.

- The Call Control Markup Language enables the management of telephone calls and conferences.

5. COMMUNITY PARTICIPATION

The speech and natural language communities are welcome to use the above specifications, and their implementations as they become available. Reusing thee specifications and implementations avoids reinventing the languages and makes the implementation easer to productize. Details about these languages can be viewed at www.w3.org/voice. Anyone in the speech and natural language communities is also welcome to comment on these languages by sending an e-mail to www-voice@w3.org.

7. SUMMARY

The W3C Speech Interface Framework languages will be used to implement commercial quality speech and natural language applications. The VBWG solicits advice about the direction and details of these evolving languages.