White Paper

SSML 1.0: an XML-based language to improve TTS rendering

Authors: Davide Bonardo, Paolo Baggia (Loquendo)
Date: January 19, 2005.

Table of Contents:

The **Speech Synthesis Markup Language** (**SSML**), the new standard way of producing content to be spoken by a speech synthesis system, is now a *W3C Recommendation* (see W3C Press Release and Testimonials[1] issued on September 8th 2004 or the full specification at www.w3.org/TR/speech-synthesis/). SSML is an XML-based markup language, which is aimed at controlling Text-To-Speech (TTS) for a variety of application contexts.

The functional and practical goal of a TTS engine is *generating spoken output* from textual or XML-based documents, such as SSML. TTS is not only useful for people suffering from visual impairments to read texts and interact with computers, but also for many applications that employ voice-output technologies. For example, alarms or announcements (e.g. in a railway station), interactive telephone services, information services (weather forecast, traffic conditions, news...), banking, e-mail or fax reading, games, etc. In all such diverse applications, TTS generates the right spoken output corresponding to the message to be communicated. In many cases, the use of TTS can replace recorded prompts from a professional speaker a lengthy and costly aspect of voice services. Moreover, TTS can be fine-grain controlled, by changing certain parameters to improve and optimize acoustic rendering. SSML helps to specify in detail how to utter a text, a paragraph, a sentence, a phrase, or a single word.

To be SSML compliant is becoming a must for TTS engines and voice platforms, because the SSML specification is part of a larger set of markup specifications for voice browsers, called W3C Speech Interface Framework, which has been finalized by the **Voice Browser Working Group**[2]. Even if the SSML is a "new" markup language, it is already crucial in many contexts. The **Voice Extendible Markup Language 2.0**[3] **- VoiceXML 2.0**, which enables Web-based development of speech applications, includes and extends SSML for prompt messages in speech applications. In multimodal applications, the SSML is mandatory both for the **Speech Application Language Tags**[4] **- SALT** prompts and for the **XHTML+Voice**[5] **- X+V** language, which reuses a modularized VoiceXML embedded into XHTML visual pages. The SSML is prescribed for prompts in the **Synchronized Multimedia Integration Language**[6] **- SMIL** and in the **Media Resource Control Protocol**[7] **- MRCP** interaction protocol.

Finally, the **Cascading Style Sheets**, employed to increase standard visual documents (like HTML), is developing a new **CSS3 Speech Module**[8], which is designed to produce SSML content. Using CCS3, will make listening to a Web page instead of reading it very easy indeed.

---

[1] http://www.w3.org/2004/09/ssml-pressrelease, http://www.w3.org/2004/09/ssml-testimonial.html.
[2] W3C Voice Browsers web site: http://www.w3.org/Voice/.
[3] VoiceXML 2.0 recommandation: http://www.w3.org/TR/voicexml20/.
[4] SALT Forum web site: http://www.saltforum.org/.
[5] X+V specification: http://www.voicexml.org/specs/multimodal/x+v/12/.
[6] SMIL Recommandation: http://www.w3.org/TR/REC-smil/.
[7] IETF MRCP protocol: http://www.ietf.org/internet-drafts/draft-shanmugham-mrcp-05.txt.
[8] CSS3 speech module specification: http://www.w3.org/TR/2004/WD-css3-speech-20040727/.

1. Inside a TTS engine

A look at the basics of TTS technology might be useful in order to understand how the SSML reading instructions may affect the behaviour of a TTS engine. While different TTS systems may adopt different algorithms to automatically obtain the correct output audio signal, a generic synthesis process can be described as shown in **Figure 1**.

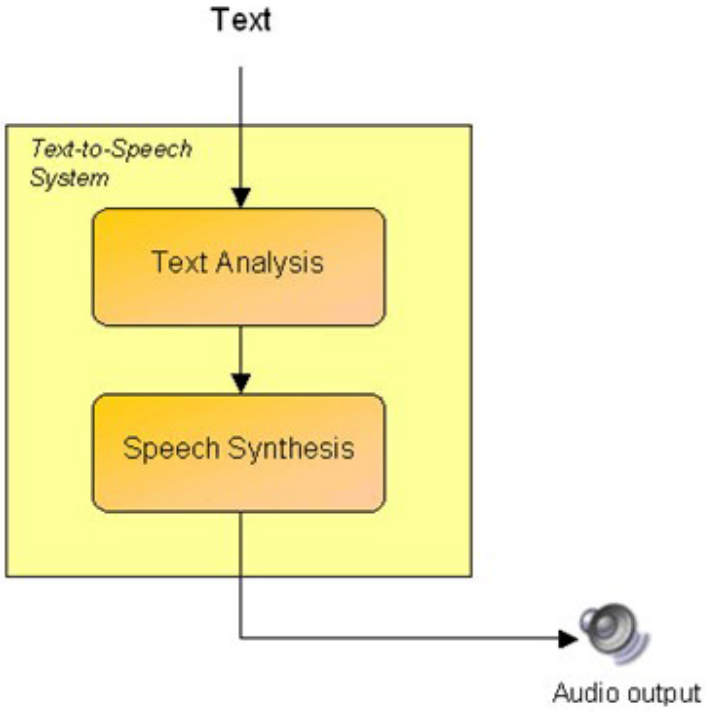

Figure 1 - A generic TTS system

The **Text Analysis step** is aimed at converting the input text into a more detailed and precise description of what should be pronounced. The exact specification of the sounds (phonemes) to be synthesized and their intonation requires an analysis of the text which solves its ambiguities, interprets punctuation, symbols, acronyms, etc. Once the output to be pronounced has been specified, the **Speech Synthesis step** generates the actual speech signal. Different techniques may be applied. The most effective in yielding high-quality and natural-sounding speech is the *Unit Concatenation* synthesis technique, which relies on a database of human speech samples from which the most suitable segments are extracted and combined to match the input text **[1]**.

SSML makes it possible to both direct the Text Analysis step and to specify a number of the features of the voice used by the Speech Synthesis step. It may be useful to look more closely at the Text Analysis step, where SSML can play a

major role. To obtain a precise description of the pronunciation of the input text, the TTS engine has to perform the following tasks:

1. ***Text normalization***
   The input text needs to be segmented into sentences and words, relying on blanks and punctuation marks. Numbers, special symbols, acronyms, abbreviations, have to be conveniently expanded to normalize the text into a standard format, consisting of graphemes (the words written in the text) and punctuation marks.

   For example, in this input text to be spoken: "*Dr. John Smith lives at Jefferson dr. 94, Paris, Texas.*" will be normalized into: "*Doctor John Smith lives at Jefferson drive ninety-four {pause} Paris {pause} Texas {pause}*" Currencies will also be expanded. If the input text is: "*The winner will receive $1000 cash!" it will be normalized into: "The winner will receive one thousand dollars cash! {pause}.*"

2. ***Word pronunciation (lexical stress, phonetic transcription)***
   Each word must be assigned its lexical stress and should be converted into a phonetic representation. Rules and lexicons are designed for this purpose. The letter-to-sound (grapheme-to-phoneme) correspondence is relatively direct for some languages, while it can be highly unpredictable for others, like English. In these cases word pronunciation can be considered lexicon-dependent.

   An example of a grapheme-to-phoneme transformation is the following: "*The page cannot be displayed!*" will be transcribed as a sequence of phonemes: /ðə/ /pˈeɪʤ/ /kˈænɑːɾ/ /bˈiː/ /dɪspˈeɪd/ (the phoneme alphabet used in this example is IPA, that stands for International Phonetic Association **[2]**).

   In addition to the rules and lexical knowledge implemented in the TTS engine, user-defined exceptions usually can be added in order to improve the TTS pronunciation on the application domain. It is possible to put a phonetic transcription directly in the text or to use a lexicon file containing all the transcriptions, acronyms and abbreviations (see point 1.)

3. Sentence structure
   Intonation and rhythm, and in some cases also stress and phonetic transcription, depend on the role of words in the sentence, i.e. on sentence structure. What is reflected in prosody (prominence, phrase separation, pauses, intonation) is the meaning of the sentence, but for synthesis systems the bridge between text and prosody is generally syntax. Syntax-based prosodic rules are usually implemented in TTS engines in order to determine the position of breath pauses, the relative prominence of words and the most suitable intonation.

4. ***Sentence-level modifications in phonetic transcription***
   Phonetically transcribed words need to be concatenated in the sentence. This requires some modifications in the phonetic transcription, accounting for changes due to continuous speech, also called co-articulation phenomena (phoneme change, reductions, de-accentuation.)

5. ***Computation of prosodic parameters***
   The analysis of a text in terms of a sequence of phonemes is completed when their prosodic description is given. Based on the previously determined general prosodic structure (step 3. above,) the target intonation and rhythm are assigned to the phoneme sequence and expressed either with numerical values or with prosodic labels, depending on the synthesis technique adopted. Prosodic value assignment can rely on explicit rules or be performed by an automatic learning system.

SSML directs all Text Analysis steps, providing a standard way to control aspects of speech such as pronunciation, acronym expansion, volume, pitch, rate, range, duration, pause, emphasis, etc., across different synthesis-capable platforms.

2. Basic description of SSML language

The following simple and explicit example, may prove useful as a reference for understanding SSML syntax.

Example 1

```
<?xml version="1.0"?>
<speak version="1.0" xml:lang="en-US">
   <voice name="Dave">
        Hello, world; my name is Dave.
   </voice>
</speak>
```

In this example the voice named *"Dave"* is to pronounce the following sentence: *"Hello, world; my name is Dave."*

Like other XML-based markup languages, SSML is composed of elements. All elements can have some attributes, which will be briefly described in the following.
The SSML language root element is **`<speak>`** it contains the SSML text to be spoken. The **`<speak>`** element has two required attributes: **`xml:lang`** (necessary to specify the language to be used for speaking) and version (indicating the version of the specification, currently "**`1.0`**"). Besides these two attributes, there are a few optional ones, such as: **`xmlns`**, used to indicate the XML Schema for SSML namespace; **`xsi:schemalocation`**, which is useful to

indicate the location of the XML Schema; **xml:base**, for specifying the base URI of the root document.

The following table shows which SSML elements are associated to the five points of Text Analysis described before; the single elements will be described below.

Step	Description	SSML elements
1.	Text Normalization:	**<sub>**, **<say-as>**, **<p>**, **<s>**, **<lexicon>**
2.	Word Pronunciations: lexical stress and phonetic transcription	**<phoneme>**, **<lexicon>**, **xml:lang** attribute
3.	Sentence Structure	**<break>**
4.	Sentence-level modification in phonetic transcription	**<emphasis>**, **<break>**
5.	Computation of prosodic parameters	**<prosody>**

Table 1 - SSML elements for controlling Text Analysis step

All other elements composing the SSML language and the text to be spoken are contained within the root **<speak>** element, but the **<meta>**, **<metadata>** and **<lexicon>** elements must occur immediately after the root element. **<meta>** and **<metadata>** elements help to annotate the document (i.e. author, date, etc.)

This is an example specifying a lexicon which helps to transform the abbreviations commonly used in SMS text messages into a normalized and speakable form.

Example 2

```
<?xml version="1.0"?>
<speak version="1.0" xml:lang="en-US">
        <lexicon uri="file://localhost/share/SMSLexicon.lex"/>
           <!-- This is the text of a SMS to be read using a
                lexicon file. -->
           I love U :-)
</speak>
```

The main elements for expressing the text structure (step 3. above) are: **<p>**, representing a paragraph, and **<s>**, representing a sentence. Another one is the

<phoneme> element, which provides in the ph attribute a phonemic/phonetic pronunciation for the contained text. It is possible to choose different phonemic/phonetic alphabets using the alphabet attribute ("**ipa**" for the International Phonetic Alphabet, see **[2]**). IPA alphabet is mandatory, but different engines can support other pronunciation alphabets (e.g. use "**x-loquendo**" for Loquendo alphabet).

Example 3

```
<?xml version="1.0"?>
<speak version="1.0" xml:lang="en-US">
  Tipical italian dishes are
  <phoneme alphabet="ipa"
        ph="&#112;&#712;&#105;&#720;&#116;&#794;&#115;&#601;">
    pizza
  </phoneme>
  <-- "pizza" will be pronunced as: pˈiːʦə -->
  and
  <phoneme alphabet="ipa"
        ph="&#115;&#112;&#601;&#103;&#712;&#101;&#116;&#812;&#105;">
    spaghetti.
  </phoneme>
  <-- "spaghetti" will be pronunced as: spəgˈeṭi -->
</speak>
```

To expand an abbreviation (step 1. above), the **<sub>** element will be used. It indicates that the contained text for pronunciation will be replaced by the **alias** attribute value.

The Text Analysis step 1 showed how a TTS system works automatically. As the knowledge contained in lexicon files (for abbreviation, acronyms, special symbols, etc.) might be not enough, it is possible use the **<sub>** element to express the expansion or abbreviations. Supposing to use a TTS to synthesize the phrase: *"The SSML v. 1.0 is a new standard of W3C"* the TTS output can be: *"The SSML v. one point o is a new standard of W3C"*. Using the SSML the synthesis will improve:

Example 4

```
<?xml version="1.0"?>
<speak version="1.0" xml:lang="en-US">
  The <sub alias="Speech Synthesis Markup Language">SSML</sub>
  <sub alias="version">v.</sub> 1.0 is a new standard of
  <sub alias="World Wide Web Consortium">W3C</sub>.
</speak>
```

All the acronyms and the abbreviations have been expanded.

In order to control prosody and style (step 5. above) one may use the **<voice>** element and its attributes (**xml:lang**, **gender**, **variant**, **age**, and **name**) to request the most suitable voice, or the **<emphasis>** and **<break>** elements, to respectively increase/decrease emphasis in a sentence and to insert or remove pauses.

The following is an extract from *Hamlet, Act I, Scene 1*. The dialog is rendered changing voices for different characters on the base of variant attribute. "1" is the narrator; "2" is Bernardo, an officer; "3" is Francisco, a soldier.

The example includes some **<emphasis>** and **<break>** elements.

Example 5

```
<?xml version="1.0"?>
<speak version="1.0" xml:lang="en">

        <voice gender="male" variant="1">
          Elsinore. A platform before the Castle.
        </voice>

        <voice gender="male" variant="2">
          <emphasis>Who's there?</emphasis>
        </voice>

        <voice gender="male" variant="3">
          Nay, answer me: stand, and unfold yourself.
        </voice>

        <voice gender="male" variant="2">
          <emphasis level="strong"> Long live the king! </emphasis>
        </voice>

        <voice gender="male" variant="3">
          Bernardo?
        </voice>

        <voice gender="male" variant="2">
          He.
        </voice>

        <voice gender="male" variant="3">
          You come most carefully upon your hour.
        </voice>

        <voice gender="male" variant="2">
          'Tis now struck twelve. Get thee to bed, Francisco.
        </voice>
```

```
        <voice gender="male" variant="3">
          For this relief much thanks: 'tis bitter cold,
          And I am sick at heart.
        </voice>

        <voice gender="male" variant="2">
          Have you had quiet guard?
        </voice>

        <voice gender="male" variant="3">
          Not a mouse stirring.
        </voice>

        <voice gender="male" variant="2">
          Well, good night.
          <break/> If you do meet Horatio and Marcellus,
          The rivals of my watch, bid them make haste.
        </voice>

        <voice gender="male" variant="3">
          I think I hear them. Stand, ho! Who is there?
        </voice>
</speak>
```

This is, of course, a joke. Currently available TTS is not suitable for interpreting plays. Nevertheless, some dialogs may be fruitfully rendered.

With its attributes to define **`pitch`**, **`contour`**, **`range`**, **`rate`**, **`duration`**, and **`volume`** the **`<prosody>`** element might be helpful for a closer control of intonation. The next example shows how to use the **`<prosody>`** element and its attributes.

Example 6

```
<?xml version="1.0"?>
<speak version="1.0" xml:lang="en-GB">
        Using prosody element it is possible
        <prosody rate ="x-fast" >
          read fast a sentence
        </prosody>
        or
        <prosody pitch ="+60Hz">
          manipulate the voice pith
        </prosody>
        and other important parameters.
</speak>
```

The SSML also provides controls for the Speech Synthesis step (see **Figure 1**). The **`<voice>`** element takes effect in this step, allowing to choose a specific voice; the **`<audio>`** element inserts an audio file in the synthesized text, and the

<mark> element inserts a bookmark that will be used for control and synchronization at the application level.

3. Final remarks

The SSML simplifies the use of TTS in a variety of contexts and applications. Resources for further study of the SSML are the W3C specification document[9] itself or tutorials, among which a very good one written by Jim Larson[10], Voice Browser Working Group co-chairman.

If you would like to test SSML documents please use the Loquendo TTS online interactive demo http://actor.loquendo.com/actordemo/default.asp?language=en.

4. References

If you are interested in a deeper analysis of how a speech synthesis engine is implemented, we invite you to read the following paper:

[1] Silvia Quazza, Laura Donetti, Loreta Moisa, Pier Luigi Salza, **"ACTOR®: A Multilingual Unit-Selection Speech Synthesis System"**, *Proc. of 4th ISCA Tutorial and Research Workshop on Speech Synthesis*, Atholl, Scotland, 2001.
See: http://www.loquendo.com/en/brochure/art_TTS_2001.pdf

[2] **International Phonetic Association.**
See http://www.arts.gla.ac.uk/ipa/ipa.html for the organization's website.

[9] SSML 1.0 Recommendation: http://www.w3.org/TR/speech-synthesis/
[10] Jim Larson's tutorial: http://www.larson-tech.com/guide/17.html