THE IMPACT OF STANDARDS ON TODAY'S SPEECH APPLICATIONS

Paolo Baggia
Loquendo SpA

ABSTRACT

At the end of the last century, the landscape of speech applications was abruptly changed due to the convergence of several factors: the maturity of speech technologies and the creation of standards to promote the development of speech applications.
The intention of this paper is to give a clear picture of this evolution, to summarize the major standards and to highlight future evolution paths.

Index Terms— Automatic Speech Recognition, Text-To-Speech, Spoken Dialog Systems, Voice Browsers, Speech Standards.

1. INTRODUCTION

At the end of the last century, the landscape of speech applications abruptly changed, not only because speech applications became common in many areas (e.g. customer care, self-service applications, voice portals), but also because of a shift from proprietary applications to standards based ones. A convergence of different factors drove this change, certainly speech technologies had reached a level of maturity to allow their use in many applications; however, the ongoing development of the Web promoted the adoption of a novel architecture called Voice Browsing. The development of standards for speech applications was another important driver, which was able to allow the creation of powerful building blocks. The intention of this paper is to give a clear picture of this evolution, to summarize the major standards and to highlight evolution paths.
All the major speech technologies were heavily studied during the second half of the last century and very important research results were found in every field. For instance:

- Text-To-Speech (TTS) reached the first goal of high intelligibility during the '80s [1], after decades of research, then during the late '90s the *Concatenative Unit Selection* [2] provided the more natural sounding voices in use today.
- Automatic Speech Recognition (ASR) research provided many results in the '70s-'80s, when the statistical approaches were developed, e.g. dynamic programming, hidden Markov models and statistical language models (cfr [3-4]). Performance was pushed by competition on large corpora like DARPA and EU funded projects.
- Natural Language Understanding (NLU) is another area that moved from pure recognition of words to meaning representation, for a survey see [5-6].
- Spoken Dialog Systems (SDS) [7] were created in the late '90s after successful projects like EU SUNDIAL and COMMUNICATOR in US.

If the technology was ready to allow the creation of a speech industry, the architectures were either results of research projects or proprietary IVR systems. In this context a standard approach mainly promoted by World Wide Web Consortium (W3C) was begun and it forced an abrupt change in the industry.
Section 2 will describe the Voice Browsing approach to speech applications, with a short description of the major standards, then Section 3 will briefly introduce the architectural changes related to the Voice Browsing Platforms in use today. Section 4 will talk of other areas of standardization and finally Section 5 will draw conclusions and discuss future evolutions of this area.

2. VOICE BROWSER STANDARDS

From an historical point of view the time was right to change the landscape of speech applications. The first signal was a W3C Workshop in 1999 promoted by Dave Raggett (W3C Lead) and James Larson, who soon became the co-chair of W3C Voice Browser working group (VBWG) [8]. This group catalyzed efforts from different companies to create a new framework for developing speech applications, which was called "Voice Browsing".
The group received the VoiceXML 1.0 proposal, which had been recently developed by the VoiceXML Forum. This became the basis of subsequent standardization activities. The seminal insights were captured by Jim Larson in the "W3C Speech Interface Framework" [9], whose illuminating chart is depicted in Fig. 1.

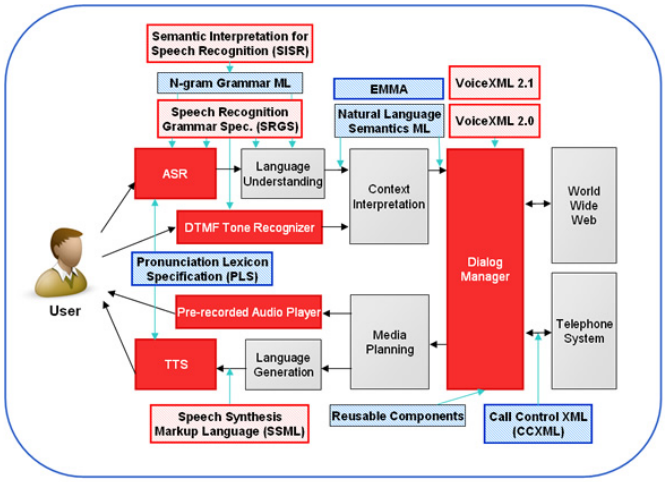

Fig. 1. Speech Interface Framework revisited

Figure 1 shows the processing modules needed to accomplish a speech (or DTMF) interaction - ASR, Language Understanding (for input processing), Context Interpretation, Dialog Manager (for dialog interaction), Media Planning, Language Generation and TTS (for output processing). The boxes outlined in red/blue are the standards to be created to support this framework; many of which are now W3C Recommendation (the ones outlined in red.) Moreover, if data are based on standards, the modules of the framework can also be completely standardized today (shown in red boxes).
The industry very soon adopted this change and interest grew significantly, which forced even the big IVR vendors to take this picture very seriously. The following sections give a brief introduction to the major standards shown in Figure 1.

2.1 Spoken Dialog (VoiceXML)

The VoiceXML 2.0 [10] standard was the key factor in the innovation. Its key features are:

- It is an XML declarative language;
- It is easy to author, the motto was: “Simple things must be easy and complex things must be possible!”
- It assumes the existence of the Web architecture.

All these features carry clear advantages; to be an XML language allows: a clean syntax checked by DTD/Schema, extensibility by Namespaces, and encodings are available from XML open source processors. Because of the second feature, VoiceXML 2.0 can be either edited by a normal text editor (then uploaded as a static page in a Web Server) or it can be dynamically generated by sophisticated Web applications, like the majority of Web pages today. Finally, to be within the Web architecture today means to share an enormous background of tools and techniques and to be placed in the mainstream of technology evolution.

From a functional point of view VoiceXML 2.0 allows the replacement of DTMF and pre-recorded applications (i.e. customer care today still offers this kind of application). This was the reason why all the IVR platforms migrated, in the last five years, to VoiceXML applications, instead of the exclusively proprietary applications developed previously. However, VoiceXML was designed to allow applications to take advantage of ASR to recognize user speech input, and of TTS to do prompting, perhaps mixed with pre-recorded audio files for speech or audio jingles. VoiceXML 2.1 [11] added eight additional features to extend VoiceXML 2.0.

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml"
  xml:lang="en-GB">

<form id="dep_arr_airports">

 <grammar src="dep_arr.grxml"
          type="application/srgs+xml"/>

 <initial name="start">
  <prompt>
     What are the arrival and departure airports?
  </prompt>
 </initial>

 <field name="fromcity">
  <prompt>Tell me the departure airport.</prompt>
 </field>

 <field name="tocity">
  <prompt>Tell me the arrival airport.</prompt>
 </field>

 <field name="go_ahead" type=”boolean” modal="true">
  <prompt>Do you want to leave from
    <value expr="fromcity"/> and arrive
    to <value expr="tocity"/>?
  </prompt>
  <filled>
     <if cond="go_ahead">
       <submit next="/servlet/dep_date"
                namelist="fromcity tocity"/>
     </if>
     <clear namelist="fromcity tocity go_ahead"/>
  </filled>
 </field>
</form>
</vxml>
```

Fig. 2. A simplified VoiceXML document

Figure 2 shows a simplified VoiceXML 2.0 document that implements a dialog to obtain departure and arrival airports. The dialog tries first to recognize both the locations, if that fails, they are asked in sequence. A final confirmation is given before transitioning to another page of the application. For a detailed introduction see [12].

2.2 Automatic Speech Recognition (SRGS and SISR)

Two standards were created to define knowledge sources for ASR, which specify prior knowledge about the language to be recognized by the ASR. The syntax for speech grammars is in SRGS 1.0 - “Speech Recognition Grammar Specification Version 1.0” [13], which has been largely adopted by the speech industry and fully supported by all ASR engines. SRGS allows the definition of grammars for speech as well as for DTMF inputs.

Moreover, *semantic interpretation* (SI), which is the part of a speech grammar devoted to the generation of a semantic result, is given in SISR 1.0 - “Semantic Interpretation for Speech Recognition Specification Version 1.0” [14]. SISR can be a simple transliteration (e.g. “coke” for “coca cola”), called *literal semantics*, or a *script semantics*, which is based on ECMAScript/JavaScript.

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar version="1.0" xml:lang="en-GB"
  xmlns="http://www.w3.org/2001/06/grammar"
  tag-format="semantics/1.0" root="fromto">
  <rule id="fromto" scope="public">
    from <ruleref uri="#city"/>
         <tag>out.fromcity=rules.latest();</tag>
    to <ruleref uri="#city"/>
       <tag>out.tocity= rules.latest();</tag>
  </rule>
  <rule id="city">
    <one-of>
      <item>London<tag>out="LHR"</tag></item>
      <item>Paris<tag>out="CDG"</tag></item>
      <item>Rome<tag>out="FCO"</tag></item>
    </one-of>
  </rule>
</grammar>
```

Fig. 3. Simple SRGS grammar with SISR script

The SRGS grammar in Figure 3 includes SISR script semantics inserted in `<tag>` elements, while the syntax is organized into rules with sequences and alternatives for words/phrases. For example, for the utterance “from Rome to Paris” the result is the ECMAScript object `{fromcity: "FCO", tocity: "CDG"}`.

2.3 Text-To-Speech (SSML)

The SSML 1.0 - “Speech Synthesis Markup Language Version 1.0” [15] is a standard to control and improve TTS rendering.

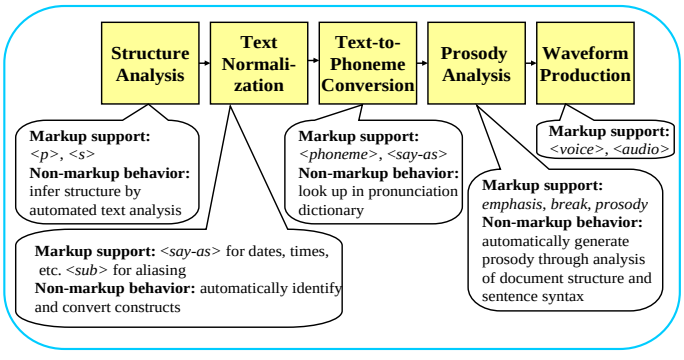

Fig. 4. Steps for TTS rendering and SSML markup

Figure 4 shows the five major processing steps present in all TTS engines. For each of them the engine already offers a normal behaviour, called “non-markup behaviour” in the picture. If needed, SSML allows the engine to improve the default rendering by means of elements of the language. Each element is related to one specific processing step and it is interpreted as a request by the author to perform an action. It

is then up to the processor to determine whether and in what way to realize the command.

```
<?xml version="1.0" encoding="UTF-8"?>
<speak version="1.0"
  xmlns="http://www.w3.org/2001/10/synthesis"
  xml:lang="en-GB">
  <p>The requested flight leaving from
         <s xml:lang="it-IT">
           <sub alias="Roma Fiumicino">FCO</sub></s>
         airport
         <emphasis>with destination
           <sub alias="London Heatrow">LHR</sub>
           </emphasis>
       are: <break time="1s"/>
       <s>
         <sub alias="British Airways 0 3 0 2">BA0302</sub>
         <break time="0.5s"/>
         leaving at
              <say-as interpret-as="time">3:45pm</say-as>
         <break time="0.5s"/>
         from gate number A63.
       </s>
       <!-- Other flight options -->
  </p>
</speak>
```

Fig. 5. A simple SSML document

The SSML example in Figure 5 shows a prompt for a flight information system. The prompt is organized into a single paragraph (<*p*>) and two sentences (<*s*>). Acronyms are substituted (<*sub*>) into expanded versions, pauses are added (<*break*>) and a time expression is explicitly labelled (<*say-as*>) to select the correct way of reading it.

2.4 Pronunciation Lexicon (PLS)

A complimentary standard of the previous ones, but still under development[1], is focused on improving pronunciation for both ASR and TTS processing. This W3C specification is PLS 1.0 – "Pronunciation Lexicon Specification Version 1.0" [16].

```
<?xml version="1.0" encoding="UTF-8"?>
<lexicon version="1.0"
  xmlns="http://www.w3.org/2005/01/pronunciation-lexicon"
  alphabet="ipa" xml:lang="en-GB">
  <lexeme>
    <grapheme>Alitalia</grapheme>
    <phoneme>ˌæl.ɪ.ˈtæi‿ə</phoneme>
    <phoneme prefer="true">a.li.ˈta:.lja</phoneme>
  </lexeme>
  <lexeme>
    <grapheme>Lufthansa</grapheme>
    <phoneme>ˈluft.hænzə</phoneme>
    <phoneme prefer="true">ˈluftˌhan.za</phoneme>
  </lexeme>
  <lexeme>
    <grapheme>AF</grapheme>
    <alias>Air France</alias>
  </lexeme>
  <lexeme>
    <grapheme>BA</grapheme>
    <alias>British Airways</alias>
  </lexeme>
</lexicon>
```

Fig. 6. PLS 1.0 document for flight applications

A PLS document stores words, or tokens, (<*grapheme*>) with the corresponding pronunciations, which may be expressed either by textual substitutions (<*alias*>) or phonetic transcriptions (<*phoneme*>). The pronunciations might be multiple to accommodate different ways of saying a word/token, or different spelling for the same pronunciation.

[1] PLS 1.0 became a Candidate Recommendation on 21 Dec. 2007 and it is in the final stage of specification, for details see: http://www.w3.org/TR/pronunciation-lexicon/

A simple PLS 1.0 document is given in Figure 6 to be used in a flight application. For "Alitalia" and "Lufthansa" the pronunciations inside the <*phoneme*> element are given in IPA (International Phonetic Alphabet) [17] - a standard way of expressing the pronunciations for all spoken human languages. Moreover, the two lexemes have a double pronunciation; the first is the normal English one while the second is more similar to the original language (Italian and German respectively). The *prefer* attribute indicates which pronunciation has to be selected for TTS rendering.

2.5 Call Control (CCXML)

If a speech interaction is a mostly synchronous activity of collecting prompts, then speaking them and waiting to listen for some input from the user, telephony by contrast is completely based on asynchronous events. Therefore, the VBWG decided to start a new separate but interlaced standard. This was CCXML 1.0 – "Call Control XML" [18].

```
<?xml version="1.0" encoding="UTF-8"?>
<ccxml version="1.0"
  xmlns="http://www.w3.org/2002/09/ccxml">
  <var name="currentState"/>
  <var name="myDialogId"/>
  <var name="myConnId"/>
  <eventprocessor statevariable="currentState">
    <transition event="connection.alerting">
      <assign name="myConnId" expr="event$.connectionid"/>
      <accept connectionid="event$.connectionid"/>
    </transition>
    <transition event="connection.connected">
      <dialogstart
        src="'http://www.example.com/flight.vxml'"
        connectionid="myConnId" dialogid="myDialogId"/>
    </transition>
    <transition event="dialog.started">
      <log expr="'VoiceXML appl is running now'"/>
    </transition>
    <transition event="connection.disconnected">
      <dialogterminate dialogid="myDialogId"/>
    </transition>
    <transition event="dialog.exit">
      <disconnect connectionid="myConnId"/>
    </transition>
    <transition event="*">
      <log expr="'Closing, unexpected:' + event$.name"/>
      <exit/>
    </transition>
  <eventprocessor/>
</ccxml>
```

Fig. 7. Basic handling of incoming calls with CCXML

This new specification is still under development, but it is on the final stage of specification. CCXML 1.0 has the potential to be a second shake up in the IVR field. CCXML 1.0 addresses simple tasks of call handling (see Figure 7), as well as complex tasks, i.e. conditional call handling, conferencing, coaching, etc.

Each CCXML document describes transitions to handle specific events. In Figure 7 a "connection.alerting" event (incoming call) is accepted, a VoiceXML dialog is started when the "connection.connected" event is received, and then the CCXML processor waits until either the caller disconnects ("connection.disconnect") or the VoiceXML dialog exits ("dialog.exit"). These are simple actions performed by telephony calls, both TDM and VoIP.

3. VOICE BROWSER PLATFORM

Besides the standards described in the previous section, another interesting advance was made on architectures to speech applications. In the past an application was developed on proprietary SDK and then deployed inside a proprietary IVR platform. After the advent of standards the architecture changed: the platform become independent from the service

to be deployed, where the application is generated by a Web Application and accessed through HTTP, like in a Web browser.

Figure 8 shows a diagram of a voice browser platform (VoxNauta from Loquendo). On the left side it interfaces to either traditional TDM telephony (fixed or mobile) or Voice-over-IP, which is today the mainstream of evolution. On right side is shown all the knowledge to be uploaded from a Web Server, which includes CCXML and VoiceXML documents, but also the SRGS grammars and audio files. For all of them caching policies can be applied to allow efficient multi-channel applications.

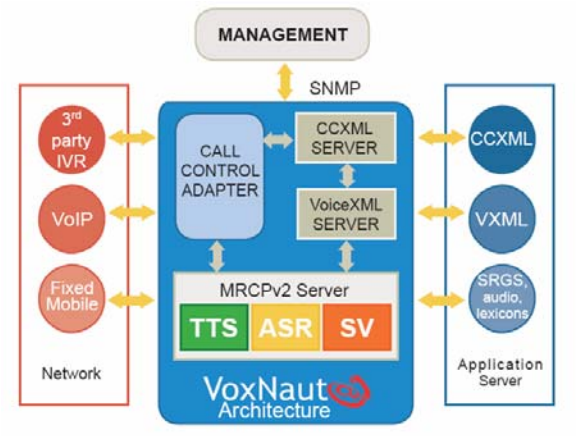

Fig. 8. VoxNauta - a Voice Browser platform

An example of a complete interaction can be the following. The platform receives an incoming call to start a new CCXML session to be controlled by a CCXML document. If the application requires a voice interaction, a VoiceXML session is started on a VoiceXML document, which in turn will provide the prompts and audio to be played by a TTS engine and SRGS grammars to be fed into the ASR engine. In VoxNauta platform the speech engines are integrated by means of the IETF MRCPv2 protocol [19].

4. RELATED STANDARDS

Other very interesting standards under development inside the W3C VBWG are SCXML [20], a generic language to represent and synchronize interactions based on State Charts that will allow the integration of multiple input modalities, and the next version of VoiceXML 3.0, which will be more modular and even more integrated with the Web.

Standards under development from the W3C Multimodal Interaction WG (MMIWG) [21] include: EMMA [22], an XML annotation of input results, InkML [23], an XML representation of digital ink to represent gesture and handwriting in a standard way, and the Multimodal Architecture and Interfaces specification [24].

Many developments have been, and are currently, done by IETF, for instance, the Media Resource Control Protocol (MRCP) [19] is the best way to integrate server speech technology into a platform.

5. CONCLUSIONS/FUTURE ADVANCES

The "Speech Interface Framework" [9], designed by W3C VBWG in 2002, is near to being completely achieved. This means that the current industry around speech technology has very solid roots and it is well placed in the midst of the advanced development of the Web.

Exciting new applications may arise from the adoption of multimodal applications that integrate speech and other modalities on small, portable devices. From an architectural point of view both multimodal and voice browsing will take advantage of the current work of the VBWG with SCXML and new features for VoiceXML 3.0, i.e. biometrics for Speaker Verification and Identification.

Speech technologies and applications are a reality today and they will certainly expand their presence in our everyday lives.

7. ACKNOWLEDGEMENTS

I would like to thank Dr. James Larson (co-chair of W3C Voice Browser WG) and Dr. Deborah Dahl (chairman of Multimodal Interaction WG) for introducing me to this world and for spending precious time reading this paper; Simon Parr for fighting with my imperfect English and many other Loquendo people, a great team to work with.

8. REFERENCES

[1] D. Klatt, "Review of text-to-speech conversion for English", JASA, 82(3), Sept. 1987.

[2] M. Balestri, A. Pacchiotti, S. Quazza, P. L. Salza and S. Sandri, "Choose the best to modify the least: a new generation concatenative synthesis system", *Proc. of EUROSPEECH-99*, Vol. 5, pp. 2291-2294, 1999.

[3] X. Huang, A. Acero, H.W. Hon, *Spoken Language Processing*, Prentice Hall, 2001.

[4] R. De Mori, "Spoken Dialogues with Computers: Signal Processing and Its Applications", *Academic Press*, 1998.

[5] D. Jurafsky, and J. Martin, *Speech and Language Processing*, Prentice Hall, 2000.

[6] J. Allen, *Natural Language Understanding*; Addison-Wesley, 1995.

[7] D. Dahl, *Practical Spoken Dialog Systems*; Springer; 2005.

[8] W3C Voice Browser WG: http://www.w3.org/voice/

[9] J. Larson, "Speech Interface Framework", *W3C note*, 2000, http://www.w3.org/TR/voice-intro/

[10] S. McGlashan, et al., "Voice Extensible Markup Language (VoiceXML) Version 2.0", *W3C Recommendation*, Mar. 2004.

[11] M. Oshry, et al., "Voice Extensible Markup Language (VoiceXML) 2.1", *W3C Recommendation*, Jun. 2007.

[12] J. Larson, *VoiceXML: Introduction to Developing Speech Applications*, Prentice Hall, 2003.

[13] A. Hunt, and S. McGlashan, "Speech Recognition Grammar Specification Version 1.0", *W3C Recommendation*, Mar. 2004.

[14] L. van Tichelen, and D. Burke, "Semantic Interpretation for Speech Recognition (SISR) Version 1.0", *W3C Recommendation*, Apr. 2007.

[15] D. Burnett, et al., "Speech Synthesis Markup Language (SSML) Version 1.0", *W3C Recommendation*, Sep. 2007.

[16] P. Baggia, "Pronunciation Lexicon Specification (PLS) Version 1.0", *W3C Candidate Recommendation*, Dec. 2007.

[17] IPA, *Handbook of the International Phonetic Association*, Cambridge University Press, 1999.

[18] RJ Auburn, "Voice Browser Call Control: CCXML version 1.0", W3C Working Draft, Jan. 2007.

[19] S. Shanmugham, and D. Burnett, "Media Resource Control Protocol Version 2 (MRCPv2)", IETF, Jan. 2008.

[20] J. Barnett, et al., "State Chart XML (SCXML): State Machine Notation for Control Abstraction", *W3C Working Draft*, Feb. 2007.

[21] W3C Multimodal Interaction: http://www.w3.org/2002/mmi

[22] M. Johnston, "EMMA: Extensible MultiModal Annotation markup language", *W3C Candidate Recommendation*, Dec. 2007.

[23] Y.-M. Chee, et al.,, "Ink Markup Language (InkML)", *W3C Working Draft*, Oct. 2006.

[24] J. Barnett, et al., "Multimodal Architecture and Interfaces", *W3C Working Draft*, Dec. 2006.