Chapter 13
Assembling the Jigsaw: How Multiple Open Standards Are Synergistically Combined in the HALEF Multimodal Dialog System

Vikram Ramanarayanan, David Suendermann-Oeft, Patrick Lange, Robert Mundkowsky, Alexei V. Ivanov, Zhou Yu, Yao Qian, and Keelan Evanini

Abstract As dialog systems become increasingly multimodal and distributed in nature with advances in technology and computing power, they become that much more complicated to design and implement. However, open industry and W3C standards provide a silver lining here, allowing the distributed design of different components that are nonetheless compliant with each other. In this chapter we examine how an open-source, modular, multimodal dialog system—HALEF—can be seamlessly assembled, much like a jigsaw puzzle, by putting together multiple distributed components that are compliant with the W3C recommendations or other open industry standards. We highlight the specific standards that HALEF currently uses along with a perspective on other useful standards that could be included in the future. HALEF has an open codebase to encourage progressive community contribution and a common standard testbed for multimodal dialog system development and benchmarking.

13.1 Introduction

Dialog systems nowadays are becoming increasingly multimodal. In other words, dialog applications, which started off mostly based on voice and text [15], have increasingly started to encompass other input–output (I/O) modalities such as

V. Ramanarayanan (✉) • D. Suendermann-Oeft • P. Lange • A.V. Ivanov • Y. Qian
Educational Testing Service (ETS) R&D, San Francisco, CA, USA
e-mail: vramanarayanan@ets.org

R. Mundkowsky • K. Evanini
Educational Testing Service (ETS) R&D, Princeton, NJ, USA

Z. Yu
Carnegie Mellon University, Pittsburgh, PA, USA

D.A. Dahl (ed.), *Multimodal Interaction with W3C Standards*,
DOI 10.1007/978-3-319-42816-1_13

video [3], gesture [3, 17], electronic ink [10, 11], avatars or virtual agents [6, 25, 26], and even embodied agents such as robots [7, 29], among others. While the integration of such technologies provides a more immersive and natural experience for the users and enables an analysis of their non-verbal behaviors, it also makes the design of such multimodal dialog systems more complicated. This is because, among other things, one needs to ensure a seamless user experience without any reduction in quality of service—this includes issues such as latency, accuracy, and sensitivity—while transporting data between each of these multimodal (and possibly disparate) I/O endpoints and the dialog system. In addition, dialog systems consist of multiple subsystems; for example, automatic speech recognizers (ASRs), spoken language understanding (SLU) modules, dialog managers (DMs), and speech synthesizers, among others, interacting synergistically and often in real-time. Each of these subsystems is complex and brings with it design challenges and open research questions in its own right. As a result, development of such multi-component systems that are capable of handling a large number of calls is typically done by large industrial companies and a handful of academic research labs since they require individual maintenance of multiple individual subsystems [5]. In such scenarios, it is essential to have industry-standard protocols and specification languages that ensure interoperability and compatibility of different services, irrespective of who designed them or how they were implemented. Designing systems that adhere to such standards also allow generalization and accessibility of contributions from a large number of developers across the globe.

The popularity of commercial telephony-based spoken dialog systems—also known as interactive voice response (IVR) systems—especially in automating customer service transactions in the late 1990s, drove industry developers to start working on standards for such systems [18]. As a core component of an IVR, the voice browser, essentially responsible for interpreting the dialog flow while simultaneously orchestrating all the necessary resources such as speech recognition, synthesis, and telephony, was one of the early components subject to standardization resulting in the VoiceXML standard dating back to 1999[1] (see Sect. 13.4.1.1 for more details on VoiceXML). Since the vast majority of authors responsible for creating standards such as VoiceXML come from the industry, most implementations of spoken dialog systems adhering to these standards are commercial, proprietary, and closed-source applications. Examples of voice browser implementations include

- Voxeo Prophecy[2]
- TellMe Studio[3]
- Plum DEV[4]

[1] http://www.w3.org/TR/2000/NOTE-voicexml-20000505.

[2] https://voxeo.com/prophecy/.

[3] https://studio.tellme.com/.

[4] http://www.plumvoice.com/products/plum-d-e-v/.

- Cisco Unified Customer Voice Portal[5]
- Avaya Voice Portal[6]

In addition to over 20 commercial closed-source voice browsers,[7] we are aware of a single open-source implementation that has been actively developed over the past few years:

- *JVoiceXML*.[8]

We adopted this voice browser for the creation of the multimodal spoken dialog system HALEF (Help Assistant–Language-Enabled and Free), which serves as an example of a standards-based architecture in this chapter.

Note that in addition to industrial implementations of spoken and multimodal dialog systems, there exists an active academic community engaging in research on such systems. Prominent examples include

- CMU's Olympus [4]
- Alex,[9] by the Charles University in Prague [12]
- InproTK,[10] an incremental spoken dialog system
- OpenDial[11]
- the Virtual Human Toolkit [9]
- Metalogue,[12] a multimodal dialog system
- IrisTK,[13] a multimodal dialog system

Many of these examples, along with other (multimodal) dialog systems developed by the academic community, are built around very specific research objectives. For example, Metalogue provides a multimodal agent with metacognitive capabilities; InproTK was developed mainly for investigating the impact of incremental speech processing on the naturalness of human–machine conversations; OpenDial allows one to compare the traditional MDP/POMDP[14] dialog management paradigm with structured probabilistic modelling [14]. Due to their particular foci, they often use special architectures, interfaces, and languages paying little attention to existing speech and multimodal standards (e.g., see the discussions in [2]). For example, none of the above research systems implements VoiceXML, MRCP, or EMMA (see Sect. 13.4 for more details on these standards).

[5] http://www.cisco.com/c/en/us/products/customer-collaboration/unified-customer-voice-portal.

[6] https://support.avaya.com/products/P0979/voice-portal.

[7] Find a comprehensive list at https://www.w3.org/Voice/voice-implementations.html.

[8] https://github.com/JVoiceXML/JVoiceXML.

[9] https://github.com/UFAL-DSG/alex.

[10] https://bitbucket.org/inpro/inprotk.

[11] http://www.opendial-toolkit.net.

[12] http://www.metalogue.eu.

[13] http://www.iristk.net.

[14] Partially Observable Markov Decision Processes.

In this chapter, we describe a system that was designed to bridge the gap between the industrial demand for standardization and the openness, community engagement, and extensibility required by the scientific community. This system, HALEF, is an open-source cloud-based multimodal dialog system that can be used with different plug-and-play back-end application modules [21, 24, 30]. In the following sections, we will first describe the overall architecture of HALEF (Sect. 13.2) including its operational flow explaining how multimodal interactions are carried out (in Sect. 13.3). We will then review major components of multimodal dialog systems that have previously been subject to intensive standardization activity by the international community and discuss to what extent these standards are currently reflected (or are planned in the future) in the HALEF framework. These include

- standards for **dialog specification** describing system prompts, use of speech recognition and interpretation, telephony functions, routing logic, etc. (primarily VoiceXML), see Sect. 13.4.1.1 (also see [1]);
- standards controlling properties of the **speech recognizer**, primarily grammars, statistical language models, and semantic interpretation (e.g., JSGF, ARPA, WFST), see Sect. 13.4.1.2;
- standards controlling properties of the **speech synthesizer** (primarily SSML);
- standards controlling the **communication** between the components of the multimodal dialog system (SIP, MRCPv2, WebRTC, EMMA), see Sect. 13.4.2;
- standards describing the **dialog flow** and how **modalities** interact (SCXML, EMMA), see Sect. 13.5.

13.2 The HALEF Dialog System

The multimodal HALEF framework [21, 24, 30] is composed of the following distributed open-source modules (see Fig. 13.1 for a schematic overview):

- Telephony servers—Asterisk [28] and Freeswitch [16]—that are compatible with SIP (Session Initiation Protocol), PSTN (Public Switched Telephone Network) and WebRTC (Web Real-Time Communications) standards, and include support for voice and video communication.
- A voice browser—JVoiceXML [22]—that is compatible with VoiceXML 2.1, can process SIP traffic, via a voice browser interface called Zanzibar [20] and incorporates support for multiple grammar standards such as JSGF (Java Speech Grammar Format), ARPA (Advanced Research Projects Agency), and WFST (Weighted Finite State Transducer), which are described in Sect. 13.4.1.2.
- An MRCPv2 (Media Resource Control Protocol Version 2) speech server—which allows the voice browser to control media processing resources such as speech recorders, speech recognizers, or speech synthesizers over the network. It relies on other protocols such as SIP for session handling, RTP (Real-time Transport Protocol) for media streaming, and SDP (Session Description

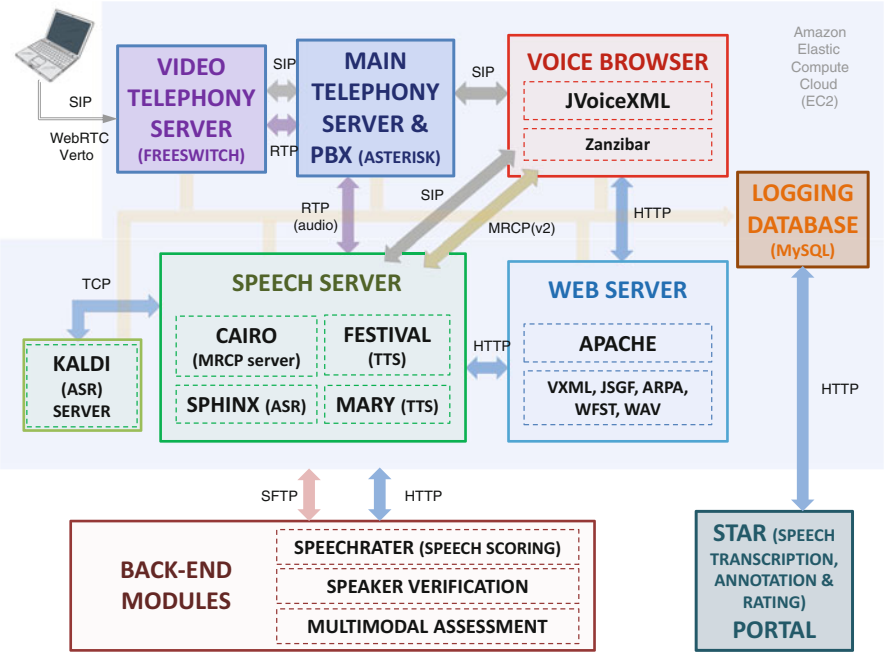

Fig. 13.1 System architecture of the HALEF spoken dialog system depicting the various modular open-source components as well as W3C standard protocols that are employed

Protocol) to allow the exchange of other capabilities such as supported codecs over the network. HALEF supports multiple speech recognizers (Sphinx [13], Kaldi [19]) and synthesizers (Mary [23], Festival [27]).

- A webserver—Apache Tomcat[15] that can host web applications that serve dynamic VoiceXML pages, web services, as well as media libraries containing grammars and audio files.
- OpenVXML, a voice application authoring suite that generates dynamic web applications that can be housed on the web server (also see Sect. 13.4.1.1).
- A MySQL[16] database server for storing call log information. All modules in HALEF connect to the database and write their log messages to it. We then post-process this information with stored procedures into easily accessible views.
- A custom-developed, open-source Speech Transcription, Annotation and Rating (STAR) portal that we implemented using PHP and the JavaScript framework jQuery. The portal allows one to analyze, listen to (or watch) full-call (video) recordings, transcribe them, rate them on a variety of dimensions such as caller experience and latency, and perform various semantic annotation tasks required to train automatic speech recognition and spoken language understanding modules.

[15] http://tomcat.apache.org/.
[16] https://www.mysql.com/.

- A custom-developed interactive dashboard written in R that allows one to view a variety of key performance indicators, including completion rate, latency, busy rate, etc.

We will illustrate the basic architecture and components of the HALEF spoken dialog system using an example application that is currently deployed in the educational domain. Finally we will conclude with a discussion of ongoing and future research and development into the system, including potential support for additional W3C standards such as EMMA (Extensible Multimodal Annotation), SSML (Speech Synthesis Markup Language), EmotionML (Emotion Markup Language), and SCXML (State Chart XML).

13.3 Operational Flow Schematic

In this section we describe how video and audio data flow to/from the multimodal HALEF system. In case of regular PSTN telephony, users call into a phone number which connects them to the telephony server in the cloud where they need to provide an extension to connect to (different extensions are associated with different dialog system instances that in turn have different task content). Alternatively, users can use softphones (or SIP phones) to connect directly to the IP address of the cloud-based telephony server using the extension. Even more convenient is the use of a web application to call directly out of a web browser application on either a computer, smartphone or tablet device. Here, the only information required by the user is the URL of the website containing the connection configuration (which includes the telephony server IP address and the extension). The Media Capture and Streams API[17] enables access to the computer's audio and video input devices via the web browser. WebRTC[18] is then used via a Javascript implementation to send video and audio to FreeSWITCH and receive audio back from FreeSWITCH. When the call comes in from the user, HALEF starts the dialog with an audio prompt that flows out of the HALEF system via Asterisk over SIP/RTP to FreeSWITCH. FreeSWITCH then sends the audio to the web browser via WebRTC. The user then gives a response to the system that flows through WebRTC to FreeSWITCH and then through SIP/RTP to Asterisk. During the teleconference, the user's video and audio interactions are continuously streamed and recorded.

Once the Asterisk server receives the call, it sends a notification to the voice browser to fetch the VXML code from the web server. The voice browser in turn identifies the resources that the speech server will need to prepare for this application. It then notifies the MRCPv2 server and starts sessions and channels for all required resources including the provisioning of speech recognition grammars.

[17] https://www.w3.org/TR/mediacapture-streams.

[18] http://www.w3.org/TR/webrtc/.

Finally, the speech server sends a SIP response back to the voice browser and Asterisk to confirm session initiation. Completion of this process successfully establishes a communication channel between the user and HALEF's components. Once the session is established, Asterisk streams audio via RTP to the speech server. When the caller starts speaking, the Sphinx engine's voice activity detector fires and identifies speech portions; then, the speech is sent to the ASR engine (HALEF supports both Kaldi and Sphinx) which starts the decoding process. When the voice activity detector finds that the caller has finished speaking, the recognition result is sent back to the voice browser, which processes it and sends this answer to the spoken language understanding module. The output of the natural language understanding module is subsequently sent to the dialog manager which evaluates and generates VXML code with the final response to be spoken out by the speech synthesizer (either Festival or Mary). The voice browser then interprets this VXML code and sends a synthesis request to the speech server with the response. The speech synthesizer synthesizes the response and passes the result back via RTP to Asterisk, which forwards the audio signal to the user. At the same time, Cairo sends a confirmation signal to the voice browser. After receiving this signal, the voice browser sends a cleanup request to close all open channels and resources. This ends the SIP session with Asterisk, which finally triggers Asterisk to send an end-of-call signal to the user.

There are other endpoints that are supported or likely can be supported by HALEF. An endpoint is defined as a device at the edge of the network (e.g., a telephone or a soft phone). Note that HALEF also natively supports audio-only dialogs with PSTN (public switched telephone network) or soft phone endpoints (that, for example, can use PSTN/SIP proxies such as ipKall).[19] We have successfully tested and used SIP clients for this purpose such as Peers[20] for PC and 3XC[21] for smartphones. We have also used SIP over WebRTC, and SIP/WebRTC clients such as sipml5,[22] jssip,[23] etc. to connect to HALEF directly through Asterisk as well as via webrtc2sip[24] to Asterisk.

13.4 Standards Used in HALEF

The following section examines in more detail how different specific industry standard specifications are synergistically combined within the HALEF multimodal dialog framework. Since HALEF is primarily a spoken dialog system, we first

[19] http://www.ipkall.com/.

[20] http://peers.sourceforge.net/.

[21] http://www.3cx.com/voip/sip-phone/.

[22] https://www.doubango.org/sipml5/.

[23] http://www.jssip.net/.

[24] http://webrtc2sip.org/.

examine the key voice standards used in its operation. We then describe the various communication standards used to transport voice and video data across different components of the dialog system.

13.4.1 Voice Standards

13.4.1.1 VoiceXML

The origins of VoiceXML[25] began in 1995 as an XML-based dialog design language intended to simplify the speech recognition application development process within an AT&T project called Phone Markup Language (PML). VoiceXML or VXML was designed for creating audio dialogs that feature synthesized speech, digitized audio, recognition of spoken and DTMF key input, recording of spoken input, telephony, and mixed initiative conversations. It was conceived to integrate the advantages of web-based development and content delivery into interactive voice response applications. The code listing below shows an example VXML page as used by HALEF. This example VXML page illustrates how various system parameters can be specified, such as the timeout value of 3 s specified in the `timeout` variable. Also, this example shows several of the components required for the interactive conversation, such as the system prompt (a prerecorded audio file, in this case) specified in the `<prompt>` element and the grammar file (see Sect. 13.4.1.2) that controls which user utterances can be recognized by the ASR system.

```
<vxml version="2.1">
<form id="InputRequestForm" scope="document">
 <field name="A_try_peanuts">
  <property name="bargein" value="true"/>
  <property name="timeout" value="3s"/>
  <property name="confidencelevel" value="0.5"/>
  <property name="sensitivity" value="0.5"/>
  <property name="speedvsaccuracy" value="0.5"/>
  <property name="completetimeout" value="3s"/>
  <property name="incompletetimeout" value="3s"/>
  <property name="maxspeechtimeout" value="10s"/>
  <property name="inputmodes" value="voice"/>
  <property name="com.telera.speechenabled" value="true"/>
  <prompt bargein="true" xml:lang="en-US">
  <audio
     src="/7703/-/resources/EPS_Builder_Voice/Default/peanuts_offer.wav"/>
  </prompt>
  <grammar mode="voice" type="application/srgs+xml"
     src="/7703/-/resources/EPS_Builder_Voice/Default/try_peanuts.
     gram"/>
 <filled>
```

[25] http://www.w3.org/TR/voicexml20/.

```
  <var name="lastresult" expr="'<lastresult>'"/>
   <submit
        next="/7703/-/next?Action_216121ee52ce43378ca2e014b92f71b4=
        success.filled"
        method="post" namelist="A_try_peanuts last result"/>
  </filled>
  <noinput></noinput>
 <nomatch></nomatch>
 <catch event="connection.disconnect.hangup"></catch>
 </field>
<catch event="externalmessage.cpa.machine"></catch>
<catch event="externalmessage.cpa.beep"></catch>
<catch event="externalmessage.cpa.machine"></catch>
</form>
<catch event="connection.disconnect.hangup"></catch>
</vxml>
```

However, developers of dialog applications who are not familiar with the VXML markup language may prefer to define dialog flows using a simpler, flowchart-based GUI instead of manual coding. Therefore we have integrated the OpenVXML toolkit into the HALEF framework. OpenVXML is an open-source software package[26] written in Java that allows designers to author dialog workflows using an easy-to-use graphical user interface, and is available as a plugin to the Eclipse Integrated Developer Environment.[27] OpenVXML allows designers to specify the dialog workflow as a flowchart, including details of specific grammar files to be used by the speech recognizer and text-to-speech prompts that need to be synthesized. In addition, they can insert "Script" blocks of Javascript code into the workflow that can be used to perform simple processing steps, such as natural language understanding on the outputs of the speech recognition. The entire workflow can be exported to a Web Archive (or WAR) application, which can then be deployed on a web server running Apache Tomcat.

Figure 13.2 shows a simple OpenVXML dialog flow where callers are required to accept or decline an offer of food in a pragmatically appropriate manner. This example can be compared to the example VXML page shown in the earlier code listing to illustrate the differences between designing a dialog directly using VXML or through the OpenVXML authoring tool. The VXML code therein corresponds to the first block in Fig. 13.2 in which a system prompt is played ("Would you like some of these chocolate covered peanuts? . . .") By double-clicking on this block in the OpenVXML tool, the designer specifies the prompt that should be played or generated by the TTS engine (as indicated in the `<prompt>` element in the VXML page), the grammar that should be used to recognize the utterance by the ASR system (corresponding to the `<grammar>` element in the VXML page), as well as a variety of system parameters, such as the timeout variable. This GUI-based representation in OpenVXML is then translated into VXML pages at run-time so that it can be interpreted by the voice browser.

[26] https://github.com/OpenMethods/OpenVXML.

[27] www.eclipse.org.

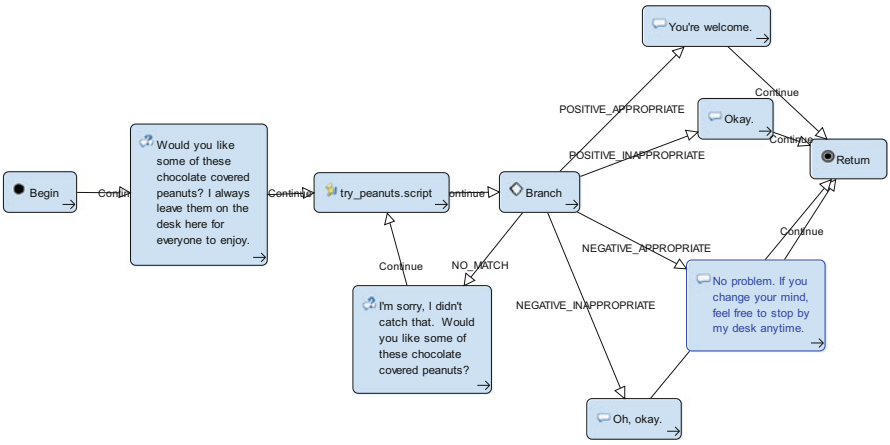

Fig. 13.2 Example design of a workplace pragmatics-oriented application targeted at non-native speakers of English where the caller has to accept or decline an offer of food (peanuts, in this case) in a pragmatically appropriate manner

The aforementioned item was designed to measure two primary constructs of English language proficiency: (1) task comprehension, i.e., correctly understanding the stimulus material and the questions being asked and (2) pragmatic appropriateness, i.e., the ability to provide a response that is appropriate to the task and the communicative context. The caller dials into the system and then proceeds to answer one or more questions, which can either be stored for later analysis (so no online recognition and natural language understanding is needed) or processed in the following manner: depending on the semantic class of the callers' answer to each question (as determined by the output of the speech recognizer and the natural language understanding module), they are redirected to the appropriate branch of the dialog tree and the conversation continues until all such questions are answered.

13.4.1.2 Voice Grammar and Language Model Standards

Grammars are used by speech recognizers to determine what a speech recognizer should listen for, and so describe the utterances a user may say. This section describes the standard grammar formats (JGSF, ARPA, WFST, and SRGS) in use by the spoken dialog community. Note that while currently HALEF only includes support for the first three, we plan to include support for this in the future.

1. *JGSF*:
   The JSpeech Grammar Format (JSGF[28]) is a platform- and vendor-independent textual representation of grammars for use in ASR. It adopts the style and

[28]JSGF (see http://www.w3.org/TR/jsgf/) is technically not a W3C standard. It is a member submission and is published as a W3C note.

conventions of the Java Programming Language in addition to use of traditional grammar notations. For example, the following JSGF grammar accepts one of two speech recognition outputs, "yes" or "no."

```
#JSGF V1.0;
grammar yesno;
public <yesno> = yes | no;
```

2. *ARPA*:
Although not a W3C recommendation, the Advanced Research Projects Agency (ARPA) format was one of the first popular ones that allowed specification of *statistical* grammars (also called language models or LMs) such as finite state automata (FSA) or statistical n-gram models. The language model is a list of possible word sequences. Each sequence listed has an associated statistically estimated language probability tagged to it. The following listing shows an example of a yes/no ARPA grammar.

```
This is an example ARPA-format language model file
\data\
ngram 1=4
ngram 2=4
ngram 3=4

\1-grams:
-0.7782 </s> -0.1761
-0.3010 <s> -0.5228
-0.7782 no -0.3978
-0.7782 yes 0.0000

\2-grams:
-0.1761 </s> <s> -0.0791
-0.3978 <s> no 0.1761
-0.3978 <s> yes -0.2217
-0.1761 no </s> 0.1761

\3-grams:
-0.3010 </s> <s> yes
-0.3010 <s> no </s>
-0.3010 <s> yes </s>
-0.3010 no </s> <s>

\end\
```

3. *WFST*:
Speech and dialog system developers nowadays are increasingly moving to the Weighted Finite State Transducer (WFST) representation to write statistical grammars for their applications owing to its simplicity and power, even though it is not an official W3C recommendation. WFSTs are automata where each transition has an input label, an output label, and a weight. The weights can be used to represent the cost of taking a particular transition. The following shows an example of a WFST grammar (in text form) that accepts the words "yes" or "no."

```
# arc format: src dest ilabel olabel [weight]
# final state format: state [weight]
# lines may occur in any order except initial state must be first line
# unspecified weights default to 0.0 (for the library-default Weight type)
0 1 yes yes 0.5
0 1 no no 1.5
1 2.0
EOF
```

4. *SRGS*:
   The Speech Recognition Grammar Specification (SRGS[29]) allows the grammar syntax to be written in one of two forms—an Augmented Backus-Naur Form (ABNF) or an Extensible Markup Language (XML) form—which are semantically mappable to allow transformations between themselves. Note that although the current version of HALEF does not include support for SRGS grammars, we plan to include this in the future. The following code snippet shows how a yes/no grammar can be defined in the ABNF format of SRGS.

```
#ABNF 1.0 UTF-8;
language en-US; //use the American English pronunciation dictionary.
mode voice; //the input for this grammar will be spoken words.
root $yesorno;
$yes = yes;
$no = no;
$yesorno = $yes | $no;
```

13.4.2 Communication Standards

WebRTC[30] or Web Real-Time Communication is a free, open W3C project that provides browsers and mobile applications with Real-Time Communications (RTC) capabilities via simple APIs. It defines a set of ECMAScript APIs in WebIDL to allow media to be sent to and received from another browser or device implementing the appropriate set of real-time protocols. As explained earlier, HALEF leverages the Verto protocol implemented in the Freeswitch video telephony server that is WebRTC-based to transmit video and audio data between the user and the dialog system.

The Media Resource Control Protocol Version 2 (MRCPv2) is a standard communication protocol for speech resources (such as speech recognition engines, speech synthesis engines, etc.) across VoIP networks which is designed to allow a client device to control media processing resources on the network.

[29] http://www.w3.org/TR/speech-grammar/.

[30] See http://www.w3.org/TR/webrtc/ and https://webrtc.org/.

13.5 Other Useful Standards for Multimodal Dialog Systems

There are several other useful standards that we are exploring for potential future integration into the HALEF framework. This section takes a closer look at some of these standards.

13.5.1 EMMA

The Extensible MultiModal Annotation (EMMA[31]) markup language is intended for use by systems that provide semantic interpretations for a variety of inputs, including but not necessarily limited to speech, natural language text, GUI, and ink input. The language is focused on annotating single inputs from users, which may be either from a single mode or a composite input combining information from multiple modes, as opposed to information that might have been collected over multiple turns of a dialog. The language provides a set of elements and attributes that are focused on enabling annotations on user inputs and interpretations of those inputs. EMMA would be a very useful standard to integrate into the HALEF framework given the focus on multimodal dialog, and hence this is one standard we are looking to include support for in HALEF going forward.

13.5.2 EmotionML

Emotion Markup Language or EmotionML,[32] as the name suggests, is "intended to be a standard specification for processing emotions in applications such as: (1) manual annotation of data; (2) automatic recognition of emotion-related states from user behavior; and (3) generation of emotion-related system behavior." Given the importance and ubiquity of emotions in dialog interactions and the subsequent requirement for automated analysis and processing of emotional state data, developing systems that are compatible with EmotionML would extend the accessibility and generalizability of those systems.

[31] http://www.w3.org/TR/emma.

[32] https://www.w3.org/TR/emotionml/.

13.5.3 SCXML

State Chart XML (SCXML[33]) is, according to the spec, "a general-purpose event-based state machine language that combines concepts from Call Control eXtensible Markup Language (CCXML) and Harel State Tables." CCXML[34] is "an event-based state machine language designed to support call control features in Voice Applications (including, but not limited to, VXML). The CCXML 1.0 specification defines both a state machine and event handing syntax and a standardized set of call control elements." Harel State Tables are a state machine notation that was developed by the mathematician David Harel [8]. They offer a clean and well-thought out semantics for sophisticated constructs such as parallel states. Although we do not require an additional state machine language as such in the current version of HALEF for smooth function, including support for SCXML in HALEF would lead to an expanded and more versatile dialog functionality, allowing one to specify dialog trees as generic state machines.

13.5.4 SSML

SSML, or Speech Synthesis Markup Language,[35] is an XML-based markup language that provides users with a standardized method for controlling different aspects of the speech output generated by a text-to-speech synthesizer. SSML allows one to alter prosody attributes such as rate, pitch, and volume. It also includes support for inserting pauses of any length, changing the speaking voice while reading, and controlling many other aspects of how the text is read by the synthetic voice.

13.6 Conclusions and Outlook

We have presented the current state of the art of the HALEF system—a fully open-source, modular, and standards-compliant spoken dialog system that can be interfaced with a number of potential back-end applications. We have illustrated the various open and W3C recommendations such as VoiceXML, WebRTC, and MRCPv2, among others, associated with different parts of the HALEF operational flow, demonstrating how these help in seamlessly assembling multiple components into a fully functional multimodal dialog system. The HALEF sourcecode is open-source and accessible online.[36]

[33] https://www.w3.org/TR/scxml/.

[34] https://www.w3.org/TR/ccxml/.

[35] https://www.w3.org/TR/speech-synthesis/.

[36] http://halef.org.

There remain many exciting directions for future research and development. For instance, the current HALEF implementation allows for audio and video input from the user and can synthesize output audio, but does not support full-fledged multimodal synthesis. In the future we would like to be able to incorporate support for video and emotion generation, as well as the control of avatars and simulations. Additionally, we would like to incorporate W3C recommendations such as EMMA and EmotionML into the HALEF architecture.

References

1. Baggia, P., Burnett, D., Marchand, R., & Matula, V. (2016, to appear). The role and importance of speech standards. In *Multimodal interaction with W3C standards: Towards natural user interfaces to everything*. Springer.
2. Baumann, T., Buß, O., & Schlangen, D. (2010). *Inprotk* in action: *Open-source software for building German-speaking incremental spoken dialogue systems*. Fachbereich Informatik: Hamburg.
3. Bohus, D., & Horvitz, E. (2010). Facilitating multiparty dialog with gaze, gesture, and speech. In *International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction* (ICMI-MLMI'10), November 8–12, 2010, Beijing, China (p. 5). ACM.
4. Bohus, D., Raux, A., Harris, T., Eskenazi, M., & Rudnicky, A.: Olympus: An open-source framework for conversational spoken language interface research. In *Proceedings of the HLT-NAACL*, Rochester (2007).
5. Damnati, G., Béchet, F., & De Mori, R. (2007). Experiments on the France telecom 3000 voice agency corpus: Academic research on an industrial spoken dialog system. In *Proceedings of the Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies*, NAACL-HLT, Rochester, NY, April 2007 (pp. 48–55). Association for Computational Linguistics.
6. DeMara, R. F., Gonzalez, A. J., Jones, S., Johnson, A., Hung, V., Leon-Barth, C., et al. (2008). Towards interactive training with an avatar-based human-computer interface. In *The Interservice Industry Training, Simulation & Education Conference, ITSEC (December 2008)*. Citeseer.
7. Gorostiza, J. F., Barber, R., Khamis, A. M., Pacheco, M., Rivas, R., Corrales, A., et al. (2006). Multimodal human-robot interaction framework for a personal robot. In *The 15th IEEE International Symposium on Robot and Human Interactive Communication, 2006. ROMAN 2006* (pp. 39–44). Hatfield, UK: IEEE.
8. Harel, D., & Politi, M. (1998). *Modeling reactive systems with statecharts: The STATEMATE approach*. New York: McGraw-Hill, Inc.
9. Hartholt, A., Traum, D., Marsella, S.C., Shapiro, A., Stratou, G., Leuski, A., et al. (2013). All together now. In *Proceedings of the 13th International Conference on Intelligent Virtual Agents, IVA 2013*, Edinburgh, UK, August 29–31, 2013 (pp. 368–381). Berlin/Heidelberg: Springer.
10. Hastie, H. W., Johnston, M., & Ehlen, P. (2002). Context-sensitive help for multimodal dialogue. In *Proceedings of the 4th IEEE International Conference on Multimodal Interfaces* (p. 93). Washington, DC, USA, IEEE Computer Society.
11. Johnston, M., Bangalore, S., Vasireddy, G., Stent, A., Ehlen, P., Walker, M., et al. (2002). Match: An architecture for multimodal dialogue systems. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL)*, Philadelphia, July 2002 (pp. 376–383).

12. Jurčíček, F., Dušek, O., Plátek, O., & Žilka, L. (2014). Alex: A statistical dialogue systems framework. In *Proceedings of the 17th International Conference on Text, Speech and Dialogue, TSD 2014*, Brno, Czech Republic, September 8–12, 2014 (pp. 587–594). Switzerland: Springer.
13. Lamere, P., Kwok, P., Gouvea, E., Raj, B., Singh, R., Walker, W., et al. (2003). The CMU SPHINX-4 speech recognition system. In *Proceedings of the ICASSP'03*, Hong Kong, China.
14. Lison, P. (2013). *Structured probabilistic modelling for dialogue management*. Ph.D. thesis, University of Oslo.
15. López-Cózar, R., Callejas, Z., Griol, D., & Quesada, J. F. (2015). Review of spoken dialogue systems. *Loquens, 1*(2), e012.
16. Minessale, A., & Schreiber, D. (2012). *FreeSWITCH Cookbook*. Packt Publishing Ltd.
17. Neßelrath, R., & Alexandersson, J. (2009). A 3D gesture recognition system for multimodal dialog systems. In *6th IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems* (pp. 46–51).
18. Pieraccini, R., & Huerta, J. (2005). Where do we go from here? Research and commercial spoken dialog systems. In *6th SIGdial Workshop on Discourse and Dialogue*.
19. Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., et al. (2011). The Kaldi speech recognition toolkit. In *Proceedings of the ASRU*, HI, USA.
20. Prylipko, D., Schnelle-Walka, D., Lord, S., & Wendemuth, A. (2011). Zanzibar openIVR: An open-source framework for development of spoken dialog systems. In *Proceedings of the TSD*, Pilsen, Czech Republic.
21. Ramanarayanan, V., Suendermann-Oeft, D., Ivanov, A., & Evanini, K. (2015). A distributed cloud-based dialog system for conversational application development. In *16th Annual SIGdial Meeting on Discourse and Dialogue (SIGDIAL 2015)*, Prague, Czech Republic.
22. Schnelle-Walka, D., Radomski, S., & Mühlhäuser, M. (2013). JVoiceXML as a modality component in the W3C multimodal architecture. *Journal on Multimodal User Interfaces 7*(3), 183–194.
23. Schröder, M., & Trouvain, J. (2003). The German text-to-speech synthesis system MARY: A tool for research, development and teaching. *International Journal of Speech Technology, 6*(4), 365–377.
24. Suendermann-Oeft, D., Ramanarayanan, V., Teckenbrock, M., Neutatz, F., & Schmidt, D. (2015). HALEF: An open-source standard-compliant telephony-based modular spoken dialog system—A review and an outlook. In *Proceedings of the IWSDS Workshop 2015*, Busan, South Korea.
25. Swartout, W., Artstein, R., Forbell, E., Foutz, S., Lane, H.C., Lange, B., et al. (2013). Virtual humans for learning. *AI Magazine, 34*(4), 13–30.
26. Swartout, W., Traum, D., Artstein, R., Noren, D., Debevec, P., Bronnenkant, K., et al. (2010). Ada and grace: Toward realistic and engaging virtual museum guides. In *Proceedings of the 10th International Conference on Intelligent Virtual Agents, IVA 2010*, Philadelphia, PA, USA, September 20–22, 2010. Lecture Notes in Computer Science (pp. 286–300). Berlin/Heidelberg: Springer.
27. Taylor, P., Black, A., & Caley, R. (1998). The architecture of the festival speech synthesis system. In *Proceedings of the ESCA Workshop on Speech Synthesis*, Jenolan Caves.
28. van Meggelen, J., Smith, J., & Madsen, L. (2009). *Asterisk: The future of telephony*. Sebastopol: O'Reilly.
29. Yu, Z., Bohus, D., & Horvitz, E. (2015). Incremental coordination: Attention-centric speech production in a physically situated conversational agent. In *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue* (p. 402).
30. Yu, Z., Ramanarayanan, V., Mundkowsky, R., Lange, P., Ivanov, A., Black, A.W., et al. (2016). Multimodal HALEF: An open-source modular web-based multimodal dialog framework. In *Proceedings of the IWSDS Workshop 2016*, Saariselka, Finland.