VoiceXML and SALT - How are they different, and why?

Stephen Potter, Microsoft and Dr. Jim A. Larson, Intel Corporation

Introduction

VoiceXML and SALT are both markup languages that describe a speech interface. However, they work in very different ways, largely due to two reasons: (i) they have different goals, (ii) they have different Web heritages.

VoiceXML is designed for telephony applications. It was developed to allow the specification of Interactive Voice Response (IVR) applications in a markup language that leveraged the benefits of the World Wide Web. It is a simple, high-level dialog markup language that facilitates the authoring of system-driven and mixed-initiative voice dialogs over telephones and cell phones.

SALT targets speech applications across a whole spectrum of devices, including telephones, PDAs, tablet computers and desktop PCs. Since many devices also contain displays, multimodal interactions are a key focus. Developers use SALT with existing Web programming standards to author system-driven, user-driven and mixed-initiative voice dialogs and multimodal applications.

These differences are manifested mainly in (i) the form of the markup, (ii) the programming and execution model, and (iii) the level of the programming interface available to the developer. The following sections discuss these differences.

Scope

VoiceXML incorporates speech interface, data, and control flow, SALT focuses on the speech interface.

VoiceXML contains a large number of tags mainly because it defines a data and execution model in addition to a speech interface. It deals not only with the user interface, (eg & lt;prompt&rt;, <grammer & rt;), but also with data models (e.g. <form&rt;, <field&rt;) and procedural programming (e.g. <if&rt;, <then&rt;, <goto&rt;). This design choice was made in order to provide a standalone markup language for describing typical telephony dialogs.

SALT has only a handful of elements because it focuses on the speech interface (e.g. <prompt&rt;, <listen&rt;, etc.). Since SALT is designed to work on different devices, it runs inside different Web execution environments. So it does not define an execution model per se, but instead uses existing execution models from the web standards community, such as HTML+ECMAScript, WML, SMIL, etc., using the Document Object Model (DOM). This allows SALT to build speech applications out of existing Web applications and enables multimodal dialogs on a variety of devices. This leads to the following differences in the programming model.

Programming model

VoiceXML has a built-in, form-filling algorithm, SALT enables application developers to write customized dialog flow.

As mentioned above, VoiceXML bundles together the speech interface, data and execution models. It contains a Form Interpretation Algorithm (FIA) that applies field-filling control flow to the visiting the <fields&rt; within VoiceXML <forms&rt;, and allows a simple model of system and mixed initiative. VoiceXML makes explicit the finite-state nature of typical system-driven dialogs, which can be easily developed by novice programmers. The FIA can be manipulated and aborted with the use of conditional and procedural programming elements for more complex dialogs.

SALT has no built-in execution model to execute its elements, but instead uses the event-wiring model familiar to Web developers. In this model, prompts are played and recognitions activated on the basis of events in the Web page. These events may arise from the data (e.g. the changing of the value in a field) from the GUI (if available, e.g. a click on a button), or from other SALT elements (e.g. a miss-recognition triggers another prompt). To enable this, SALT elements expose an object interface of attributes, properties, methods and event handlers, which, as noted above, is completely integrated into the execution model of the page. This follows the well-understood Web-programming model, which means that for existing Web developers, the learning curve

for adding speech to a Web application is rapid. It also allows for a finer level of user experience control (see below) and, importantly, allows for clean integration of speech into multimodal pages.

Level of API

VoiceXML has a high-level API and SALT has a lower-level API.

VoiceXML uses <forms&rt; and <fields&rt; as its building blocks of dialog. This allows the bundling together of prompts and grammars into fields that are executed one-by-one to reflect the turn-taking control model of telephony dialogs.

SALT applies a lower level interface to offer a finer grain manipulation of speech input and output primitives. SALT leaves the customization of turn-taking and dialog flow to the application author (or to higher level tools). The level of control is exposed at a finer level, so that more sensitive interactions can be authored for both telephony and multimodal dialogs.

Other standards

VoiceXML and SALT both use W3C standards. The W3C Voice Browser Working Group is developing a suite of specifications to enable the interoperability of speech applications. This is entitled the W3C Speech Interface Framework. The framework currently includes working drafts of:

- SRGS, a grammar format, http://www.w3.org/TR/speech-grammar
- SSML, a speech output format, http://www.w3.org/TR/speech-synthesis
- NLSML, a natural language result format , http://www.w3.org/TR/nl-spec
- CCXML, and a telephony call control language, http://www.w3.org/TR/ccxml

Both VoiceXML and SALT recommend the use of SRGS and SSML as grammar and speech output formats, respectively. In addition, SALT also recommends the use of NLSML as a recognition result format, and CCXML as a telephony call control language (or a call control object closely modeled on CCXML as an alternative).

Licensing *VoiceXML may be subject to royalty payments and SALT will be royalty-free.*

IBM, Motorola, Lucent, AT&T and others have indicated that they may have patents involving intellectual property in VoiceXML and insist on a reasonable and non-discriminatory licensing policy. (http://www.w3.org/2001/09/voice-disclosures.html)

The SALT Forum intends to provide a license to the SALT specification on royalty-free terms.

The future

Although SALT and VoiceXML were developed to solve different problems, these problem spaces are beginning to converge. Some VoiceXML developers are asking for a stripped-down version of VoiceXML, without the FIA, so they can write their own turn-taking strategies for complex speech applications. Other VoiceXML developers are asking that VoiceXML be modularized so that its tags can be embedded into other languages. SALT already applies a model that fits these roles.

The paths of each language are pointing in the same direction and the authors hope that the standards bodies that develop future versions of SALT and VoiceXML will attempt to find a technical convergence path that leads eventually to a single speech markup standard.

Stephen Potter is with the Technical Working Group of the SALT Forum and can be reached at spotter@microsoft.com. Dr. Jim A. Larson is chair of the W3C Voice Browser Working Group and can be reached at jim@larson-tech.com.