What is a Spoken Language Dialog System?

• An SLDS is a computer system that you can talk to in order to carry out some task.

• SLDSs are typically of two kinds:
  – Information-provision systems provide information in response to a query, such as a request for timetable information or weather information.
  – Transaction-based systems allow you to undertake some transaction, such as buying or selling stocks, or reserving a seat on a plane.
The Architecture of an SLDS

- Speech Recognition
- Language Understanding
- Dialog Management
- Database
- Speech Synthesis
- Language Generation

© Macquarie University 2007
What's Involved in Building an SLDS

- **Dialog Design**
  - The process of working how the interaction between human and machine will move from stage to stage.
  - Also referred to as script writing or call flow layout.

- **Prompt Design**
  - What the system says to get the user to say something that is permitted to be said.

- **Grammar Writing**
  - Specifying what the user is permitted to say at any given state.
Software Development Life Cycle

- There are some specific tasks to speech applications:
  - dialog design (= call flow layout)
  - specifying the persona
  - coding the application (in VoiceXML)
    - grammar writing
    - error handlers
    - speech synthesis tags
# Types of Spoken Output

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prompt</td>
<td>Indicates it is time for user input, and thus serves as a turn-taking cue</td>
</tr>
<tr>
<td>Feedback</td>
<td>Presents the app state that results from user input, allowing the user to compare original intent with result</td>
</tr>
<tr>
<td>Instructions</td>
<td>Provide information about operating the user interface or understanding the task</td>
</tr>
<tr>
<td>Help</td>
<td>Help instructions often adopt a separate mode or state aimed at coaching the user</td>
</tr>
<tr>
<td>App Data</td>
<td>Information presented to the user as part of the task: eg weather, stock information, flight times</td>
</tr>
</tbody>
</table>
Components of the Speech Interface Framework
A Speech Application Fragment

```xml
<?xml version = "1.0"?>
<vxml version = "2.0" xmlns = "http://www.w3.org/2001/vxml">
...
<form id = "travel">
  <field name = "destination">
    <prompt>Do you want to fly to
    <emphasis level = "strong"> New York </emphasis> or to <emphasis level = "strong"> Washington </emphasis></prompt>
    <grammar mode = "voice" root = "destination-city">
      <rule id = "destination-city"/>
        <one-of>
          <item tag = "NEW-YORK"> New York </item>
          <item tag = "NEW-YORK"> Big Apple </item>
          <item tag = "WASHINGTON"> Washington </item>
          <item tag = "WASHINGTON"> The Capital </item>
        </one-of>
    </rule>
  </field>
</form>
</vxml>
```
VoiceXML

- VoiceXML is designed for creating audio dialogs that feature
  - recognition of spoken and DTMF key input,
  - recording of spoken input,
  - mixed initiative conversation,
  - synthesized speech,
  - recorded speech,
  - telephony.

- Its major goal is to bring the advantages of Web-based development and content delivery to interactive voice response applications.
The Speech Recognition Grammar Specification (SRGS) specifies the words and phrases which a user may speak in response to a prompt.

The syntax of the grammar format can be presented in two forms:
- Augmented BNF Form
- XML Form.

The specification makes the two representations mappable to allow automatic transformations between the two forms.

See: http://www.w3.org/TR/speech-grammar/
SSML

- The Speech Synthesis Markup Language (SSML)
  - describes how text is presented as audio to the user
  - improves the quality of synthesized content
  - offers a standard way to control aspects of speech.
- See: http://www.w3.org/TR/speech-synthesis/
The Call Control Extensible Markup Language (CCXML)
- provides call control for VoiceXML telephony platforms
- allows for controlling how phone calls are placed, answered, transferred, conferenced.

Traditionally, call control has required interaction with the telephony API's which often change from one platform to another.

See: http://www.w3.org/TR/2007/WD-ccxml-20070119/
EMMA

- The Extensible MultiModal Annotation markup language is used for providing semantic interpretations for a variety of input modes:
  - speech,
  - natural language text,
  - graphical user interface,
  - and electronic ink input.
- The markup will be used as a standard data interchange format.
- The markup will be automatically generated by interpretation components to represent the semantics of users' inputs.
Developing Speech Interfaces

• Speech interfaces can be developed using
  – general-purpose languages (e.g. C++, Java, Python)
  – special-purpose languages (e.g. VoiceXML, SALT).

• A special-purpose language can
  – simplify application development
  – separate interaction code from application logic code
  – reduce network traffic
  – provide portability and simplicity
  – support prototyping and refinement.
VoiceXML Architecture

- regular phone
- wireless phone
- soft phone

- telephony interface
- voice browser
- automated speech recognition
- text-to-speech synthesis
- touchtone
- audio play/record

- VoiceXML documents
- audio files
- service logic (CGI)
- transaction processing
- database interface
Form Interpretation Algorithm

• VoiceXML forms are interpreted by an implicit form interpretation algorithm.

• The form interpretation algorithm has a main loop that
  – selects a form item
  – and then visits it.

• The selected form item is
  – the first in document order
  – whose guard condition is not satisfied.
Form Interpretation Algorithm

• The form interpretation algorithm ends
  – when it interprets a transfer of control statement (e.g. <goto> or <submit>)
  – when no form item remains to select (implied <exit>).
Customizing the Form Interpretation Algorithm

• The FIA can be customized in several ways:
  – assigning a value to a form item variable (<assign>)
  – setting a form item variable to undefined (<clear>)
  – specifying the next form item to visit (<goto>).
Working with Multi-Document Applications

• Application root document (app-root.vxml)

```xml
<?xml version = "1.0" encoding="UTF-8"?>
<vxml version = "2.0">
...
</vxml>
```

• Leaf document (leaf.vxml)

```xml
<?xml version = "1.0" encoding="UTF-8"?>
<vxml version = "2.0" application = "app-root.vxml">
...
</vxml>
```
Grammar Standard

• No standard SR grammar format was available for VoiceXML 1.0.
• Voice browser developers had to define the grammar and format.
• This problem was rectified with the Speech Recognition Grammar Specification introduced with VoiceXML 2.0.
• See: http://www.w3.org/TR/speech-grammar/
Grammar Formats

- The Speech Recognition Grammar Specification provides two formats: XML and ABNF (a plain text representation).
- VoiceXML 2.0 platforms must support the XML format.
- VoiceXML 2.0 platforms should also support the ABNF format.
- Both grammars have the power of a Context Free Grammar\(^1\).
- The two formats can be converted into each other.

\(^1\) A grammar processor that does not support recursive grammars has the expressive power of a Finite State Machine or regular language.
Grammar Formats

- VoiceXML 2.0 platforms may support vendor-dependent formats:
  - Nuance Grammar Specification Language (GSL),
  - Java Speech Grammar Format (JSGF).
- OptimTalk supports the XML grammar format.
- Tellme Studio and BeVocal support the GSL grammar format.
N-gram Grammars

- Most VoiceXML interpreters use conversational SR engines.
- They are constructed to use XML or ABNF grammars.
- Some interpreters support dictation SR engines.
- Dictation SR engines use N-gram grammars.
- N-grams are a representation of
  the probabilities of sequences of \( n \) words
  within a large collection of text
  about a specific domain.
What is JavaScript?

- JavaScript is an object-oriented scripting language.
- Client-side JavaScript
  - is an implementation of ECMAScript
  - is usually embedded directly in HTML pages
  - is interpreted.
- Server-side JavaScript
  - is used with Web servers such as Apache (mod_javascript)
  - can access the file system and connect to relational databases
  - is compiled.
```
... <rule id = "toppings" scope = "public">
  <tag> $ = new Array(); </tag>
  <item repeat = "1-"> <rule ref uri = "#topping"/>
    <tag> $.push($topping); </tag>
  </item>
</rule>

<rule id = "topping" scope = "public">
  <one-of>
    <item> cheese </item> 
    <item> ham </item> 
    <item> pepperoni </item> 
    <item> mushrooms </item>
  </one-of>
</rule>
```
What is Voice over IP?

• Voice over IP is a generic term for all types of voice communication using the Internet protocol IP (TCP/IP) instead of traditional circuit switched technology.

• This includes packet technologies by telecommunication companies to carry voice at the core of their networks in ways that are not controlled by an end user.
What is Internet Telephony?

• Internet telephony is a service that end users decide to use.
• It is a specialised form of Voice over IP.
• Regular voice telephone calls are transmitted via the Internet.
• All or part of the public switched telephone networks is bypassed.
• Internet telephony can occur
  – between computers
  – between a computer and a phone
  – between phones.
About SIP

• SIP (Session Initiation Protocol)
  – is an application layer-protocol,
  – can establish, manage and terminate voice and video sessions,
  – sessions involve one or more participant,
  – can use unicast or multicast communication,
  – is being developed by the SIP Working Group,
  – has the status of a proposed standard.
Dialog Styles

<table>
<thead>
<tr>
<th>Mixed Initiative</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Application Directed</strong></td>
</tr>
<tr>
<td>Forms</td>
</tr>
<tr>
<td>DTMF Menus</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Form Filling Dialog Model

• In a form filling dialog model
  the application typically prompts the caller
  for discrete pieces of information
  in a pre-determined order.
• In this model, the VoiceXML application mainly
  consists of a number of call states
  that collect input from the caller.
Problems with Form Filling

- For callers, "form filling" can become quite cumbersome.
- Especially when callers are accustomed to provide multiple pieces of information
  - in succession without interruption of intermediary prompts
  - in a different order than specified by the application.
- For example:
  - \((A + B + C)\)
  - \((B + A) + C\)
Mixed Initiative Dialog Model

• In a mixed initiative dialog model
  – the call flow
    can be directed by the caller or by the application
  – the application
    collects pieces of information in a single call state.
• VoiceXML does not allow for true free-form utterances.
• However, it is possible to approximate this dialog style.
Implementing Mixed-Initiative

• The following things need to be done:
  – define subgrammars to collect each piece of information
  – define a form level grammar that uses the subgrammars to collect the information
  – define a mixed initiative dialog that collects input from the caller.
• The mixed initiative dialog can be built on top of a form-filling dialog.
SALT

• SALT (= Speech Application Language Tags)
  – is an extension of HTML
  – consists of a small set of XML elements (tags)
  – adds a powerful speech interface to Web pages.

• SALT can be used for both
  – voice-only browsers
  – multimodal browsers.
SALT: Multimodality

- Multimodal
  - adding SALT to a visual page (HTML, cHTML, WML)
  - speech-enabling controls
    - for push-to-talk form-filling scenarios
    - for more complex mixed initiative capabilities
SALT: Multimodality

- Recognition may be started by a browser event (clicking on button).
- Activates a grammar of an input field.
- Binds the recognition result into that field.

```html
<!-- HTML -->
  ... 
  <input name="txtBoxCity" type="text" />
  <input name="buttonCityListen" type="button" onclick="listenCity.Start();" />
  ...

  <!-- SALT -->
  <salt:listen id="listenCity">
    <salt:grammar name="g_city" src="/city.grxml" />
    <salt:bind targetElement="txtBoxCity" 
      value="/city" />
  </salt:listen>
</html>
```
SALT: Telephony

• For SALT applications without a visual display, the application drives the interaction with the user by prompting for required information.
• HTML scripting and the event model performs this function.
• Using scripting and the event model, the full control is available to developers for the management of prompt playing grammar activation and processing of recognition results.
SALT: Telephony

• Implementations of SALT are expected to provide scriptlets.
• Scriptlets handle common dialog processing tasks.
• For example, the \texttt{RunAsk()} function of the next example
  — activates prompts and recognition
  — until the values of the input fields are filled.
<body onload="RunAsk()">
<form id="travelForm">
  <input name="txtBoxOriginCity" type="text" />
  <input name="txtBoxDestCity" type="text" />
</form>

<!-- Speech Application Language Tags -->
<salt:prompt id="askOriginCity"> Where would you like to leave from? </salt:prompt>
<salt:prompt id="askDestCity"> Where would you like to go to? </salt:prompt>

<salt:listen id="recoOriginCity" onreco="procOriginCity()">
  <salt:grammar src="city.xml" />
</salt:listen>

<salt:listen id="recoDestCity" onreco="procDestCity()">
  <salt:grammar src="city.xml" />
</salt:listen>
<!-- scripted dialog flow -->
<script>
    function RunAsk() {
        if (travelForm.txtBoxOriginCity.value==") {
            askOriginCity.Start();
            recoOriginCity.Start();
        } else if (travelForm.txtBoxDestCity.value==") {
            askDestCity.Start();
            recoDestCity.Start();
        }
    }
    function procOriginCity() {
        travelForm.txtBoxOriginCity.value = recoOriginCity.text;
        RunAsk();
    }
    function procDestCity() {
        travelForm.txtBoxDestCity.value = recoDestCity.text;
        travelForm.submit();
    }
</script>
</body>
</html>
VoiceXML versus SALT

• VoiceXML and SALT are both
  – markup languages
  – that describe speech interfaces.
• VoiceXML is designed for telephony applications:
  – interactive voice response applications are the focus.
• SALT targets speech application across a whole spectrum:
  – multimodal interactions are the focus.
VoiceXML versus SALT

- The differences between VoiceXML and SALT are manifested in:
  - the form of the markup
  - the programming and execution model
  - the level of programming interface available for developers.
VoiceXML 2.1

• VoiceXML 2.1 is not a replacement for VoiceXML 2.0.
• It only discusses 8 elements.
• Two of these 8 elements are new ones.
• 6 elements are enhancements to existing VoiceXML elements.
• VoiceXML 2.1 is a W3C Recommendation since 19 June 2007.
• See: http://www.w3.org/TR/voicexml21/
### VoiceXML 2.1 — New and Enhanced Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Purpose</th>
<th>Section</th>
<th>New/Enhanced</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;data&gt;</code></td>
<td>Fetches arbitrary XML data from a document server.</td>
<td>5</td>
<td>New</td>
</tr>
<tr>
<td><code>&lt;disconnect&gt;</code></td>
<td>Disconnects a session.</td>
<td>8</td>
<td>Enhanced</td>
</tr>
<tr>
<td><code>&lt;grammar&gt;</code></td>
<td>References a speech recognition or DTMF grammar.</td>
<td>2</td>
<td>Enhanced</td>
</tr>
<tr>
<td><code>&lt;foreach&gt;</code></td>
<td>Iterates through an ECMAScript array.</td>
<td>6</td>
<td>New</td>
</tr>
<tr>
<td><code>&lt;mark&gt;</code></td>
<td>Declares a bookmark in a sequence of prompts.</td>
<td>4</td>
<td>Enhanced</td>
</tr>
<tr>
<td><code>&lt;property&gt;</code></td>
<td>Controls platform settings.</td>
<td>5.1, 7</td>
<td>Enhanced</td>
</tr>
<tr>
<td><code>&lt;script&gt;</code></td>
<td>References a document containing client-side ECMAScript.</td>
<td>3</td>
<td>Enhanced</td>
</tr>
<tr>
<td><code>&lt;transfer&gt;</code></td>
<td>Transfers the user to another destination.</td>
<td>9</td>
<td>Enhanced</td>
</tr>
</tbody>
</table>
The Future of VoiceXML

VoiceXML 3.0 is the next major release of VoiceXML. Its purpose is to provide powerful dialog capabilities that can be used to build advanced speech applications, and to provide these capabilities in a form that can be easily and cleanly integrated with other W3C languages. It will provide enhancements to existing dialog and media control, as well as major new features (e.g. modularization, a cleaner separation between data/flow/dialog, and asynchronous external eventing) to facilitate interoperability with external applications and media components.

http://www.w3.org/Voice/