Desarrollo de sistemas de diálogo mediante VoiceXML

Reconocimiento Automático del Habla

Máster de Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Curso 2009/10

David Griol Barres

ÍNDICE

Objetivos

- VoiceXML: lenguaje de acceso vocal a la información en Internet.
- Mostrar las características de esta tecnología.
- Recursos disponibles: IBM Voice Toolkit.
- Aplicación práctica: centralita en castellano.

VOICE XML: Funcionalidades

- Ventajas de las tecnologías web para el desarrollo de aplicaciones controlables mediante la voz:
  - Integración de servicios de voz y de datos.
  - Compatibilidad con otros lenguajes.
- Orígenes: Necesidad de estandarización.
  - Lenguaje estándar para generar aplicaciones de reconocimiento de voz.
  - Base: XML.
  - Reconocimiento semántico a través de gramáticas.
  - VoiceXML Forum (2000).

VOICE XML: Funcionalidades

- En los años 80 y 90, los desarrolladores de sistemas de diálogo debían programar a bajo nivel
- En años 90 surgen navegadores Web capaces de soportar voz humana
  - Diseñadores de sistemas de diálogo sólo han de concentrarse en la lógica, dejando al margen cuestiones de bajo nivel
- VoiceXML (o VXML)
  - Estándar basado en XML desarrollado por el W3C que permite acceder mediante habla a aplicaciones Web
  - Comunicación SD → Usuarios: habla sintetizada, ficheros de voz pregrabados
  - Comunicación Usuarios → SD: habla, DTMF
- Versiones de VoiceXML
  - v1.0 (2000)
  - v2.0 (2004)
  - V2.1 (2007)

http://www.w3.org/TR/2000/NOTE-voicexml-20000505/
http://www.w3.org/TR/voicexml20/
http://www.w3.org/TR/voicexml21/

VOICE XML: Funcionalidades

- VXML: lenguaje para crear sistemas de diálogo mediante de voz, utilizando:
  - Reconocimiento de voz.
  - Reconocimiento de entrada DTMF.
  - Funciones de telefonía.
  - Control del flujo del diálogo.
  - Grabación de diálogos.
  - Salida de voz sintetizada.
  - Salida de ficheros de audio.
- Aprovechar las ventajas de la tecnología web.

Sistema de Diálogo Hablado

Sistema automático capaz de emular a un ser humano en un diálogo con otra persona, con el objetivo de que el sistema cumpla con una cierta tarea:

- Suministrar información.
- Reserva de viajes, hoteles, restaurantes…
- Automatizar operaciones (domótica)
- Banca electrónica.
- Control de máquinas.
- Personas con discapacidades.
- …

Sistema de Diálogo Hablado

- **Módulo de Reconocimiento Automático del Habla**:
  - Señal vocal pronunciada por el usuario.
  - Secuencia de palabras reconocida más probable.
- **Módulo de Comprensión del Habla:**
  - Secuencia(s) de palabra(s) reconocida(s),
  - Representación semántica de su significado.
- **Gestor de Diálogo**:
  - Considera: interpretación semántica de la petición del usuario, la historia del proceso de diálogo, estado actual del diálogo…
  - Siguiente acción que debe tomar el sistema siguiendo la estrategia del diálogo.

Sistema de Diálogo Hablado

- **Módulo de Consulta a la Base de Datos de la Aplicación:**
  - Recibe peticiones de consulta a la base de datos por parte del gestor de diálogo
  - Procesa y devuelve el resultado al gestor.
- **Módulo de Generación de Respuestas:**
  - Respuesta del sistema en forma de representación formal y tiene
  - Frase, gramaticalmente correcta en lenguaje natural. Otras modalidades de información (vídeo, tablas con datos, gestos a reproducir por un avatar...).
- **Sintetizador de Texto a Voz:**
  - Respuesta del sistema (texto en lenguaje natural).
  - Correspondiente señal de audio.

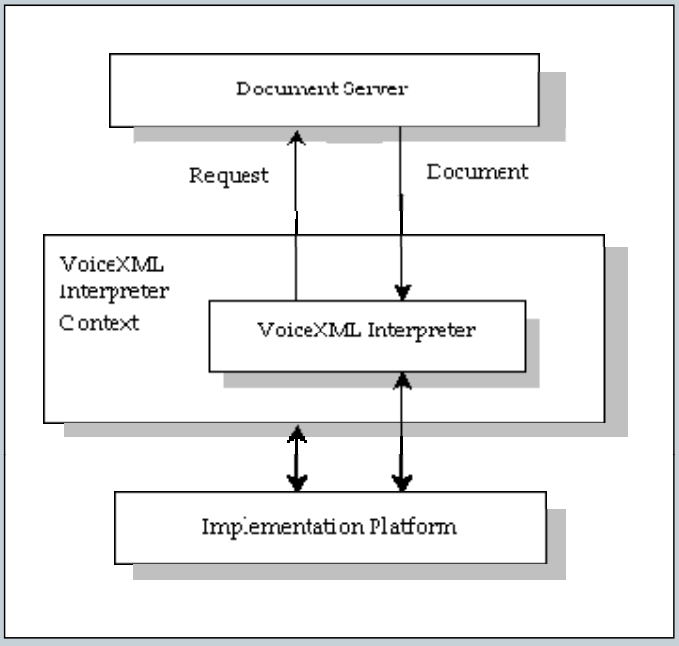
VOICE XML: Ventajas

- Ventajas:
  - Múltiples interacciones en un único documento.
  - Lenguaje alto nivel.
  - Separa los diferentes tipos de código.
  - Portabilidad de servicios.
  - Facilidad de uso en diferentes tipos de diálogos.

VOICE XML: Requisitos

- Requerimientos de la plataforma:
  - Adquisición de documentos.
  - Salida de audio.
  - Entradas de audio.
  - Transferencia.

VOICE XML: Arquitectura

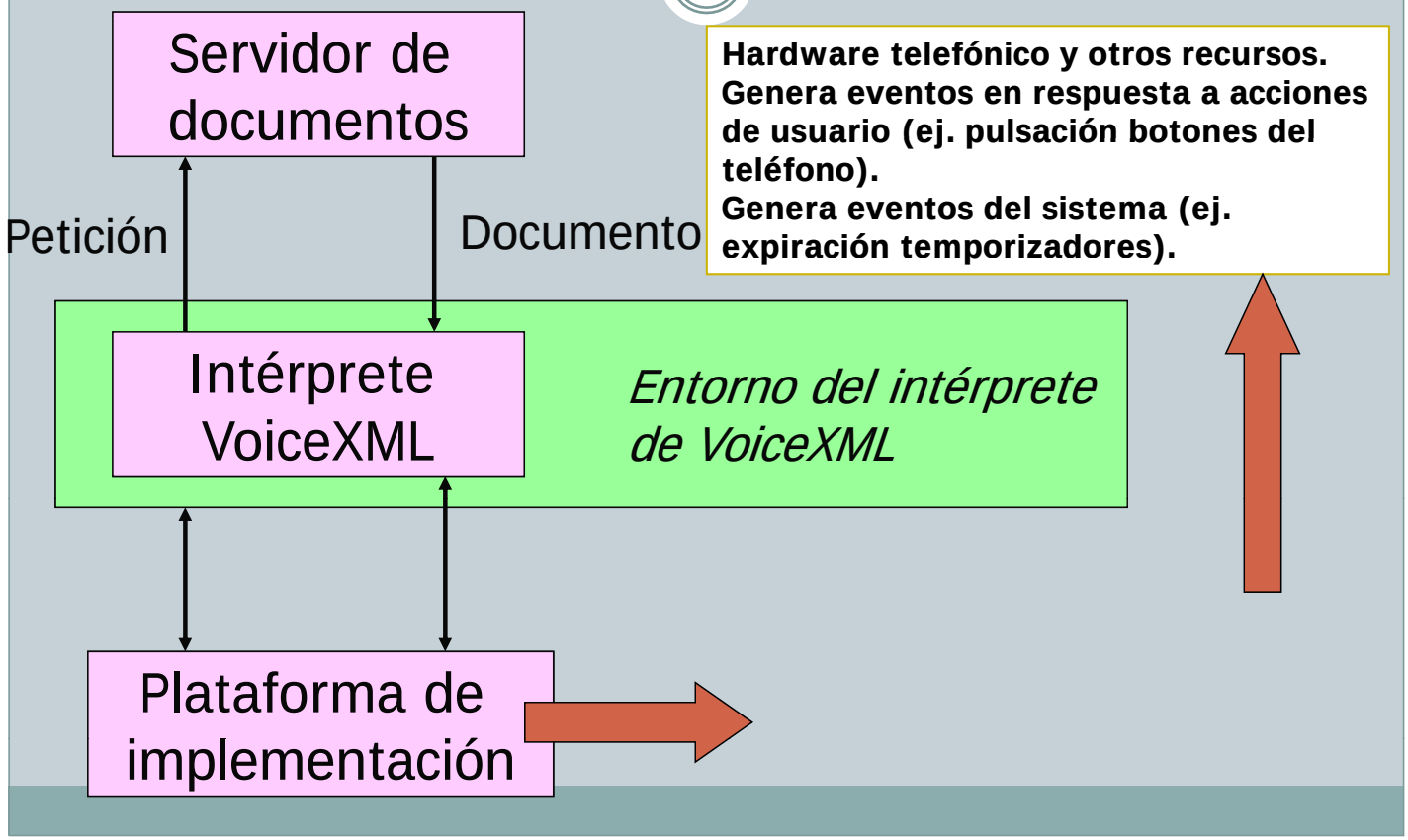


- Document Server:
  - Procesa peticiones del VoiceXML Interpreter a través del VoiceXML Interpreter Context.
  - Genera documentos que son procesados por el VoiceXML Interpreter.
- La plataforma de implementación genera eventos en respuesta a las acciones de usuario.

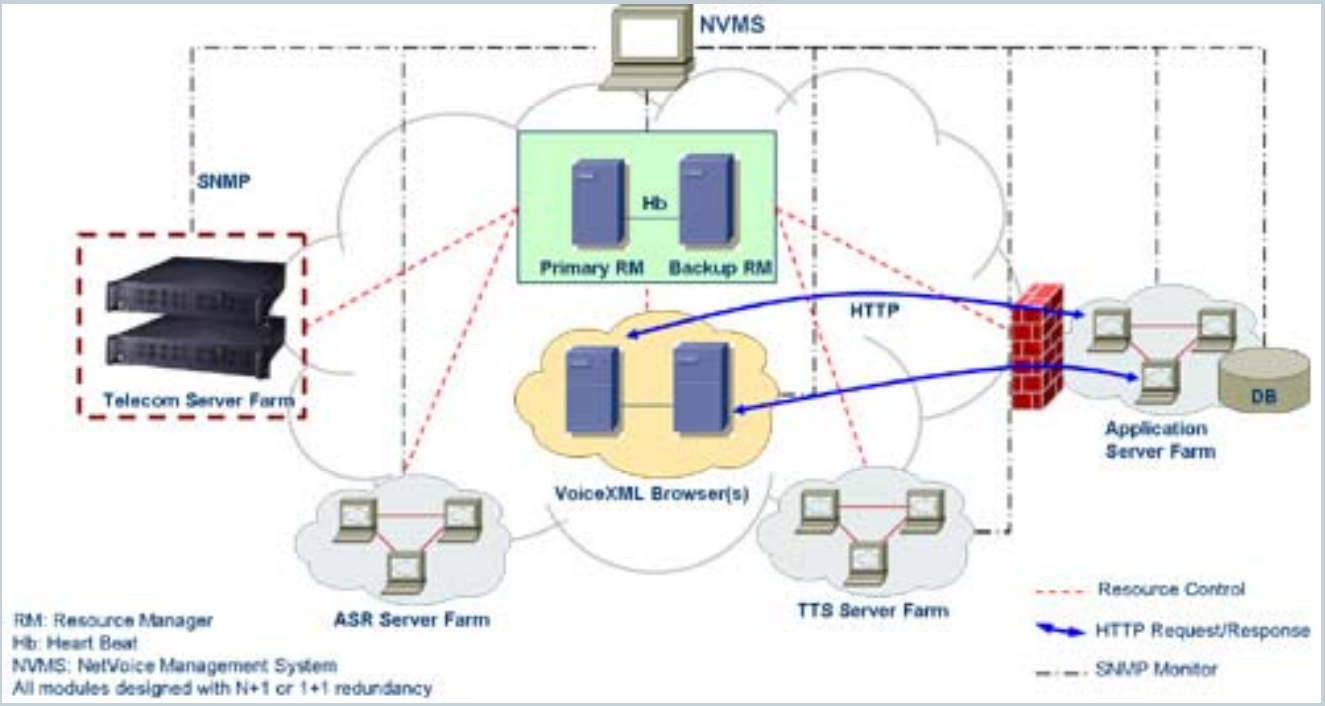
VOICE XML: Arquitectura

- Intérprete de VoiceXML *(aplicación cliente)*
  - Ejecuta lógica de aplicación
  - Genera prompts y procesa respuestas del usuario
  - Busca información en sitios Web para proporcionarla al usuario
- Servidor de documentos *(servidor Web)*
  - Procesa peticiones enviadas por intérprete de VoiceXML
  - Proporciona documentos VoiceXML
- Entorno del intérprete de VoiceXML
  - Procesa documento VoiceXML
  - Responde a llamadas de usuarios
  - Monitoriza entradas de usuario (ayuda, no respuesta, etc.)
  - y genera mensajes predefinidos

VOICE XML: Arquitectura



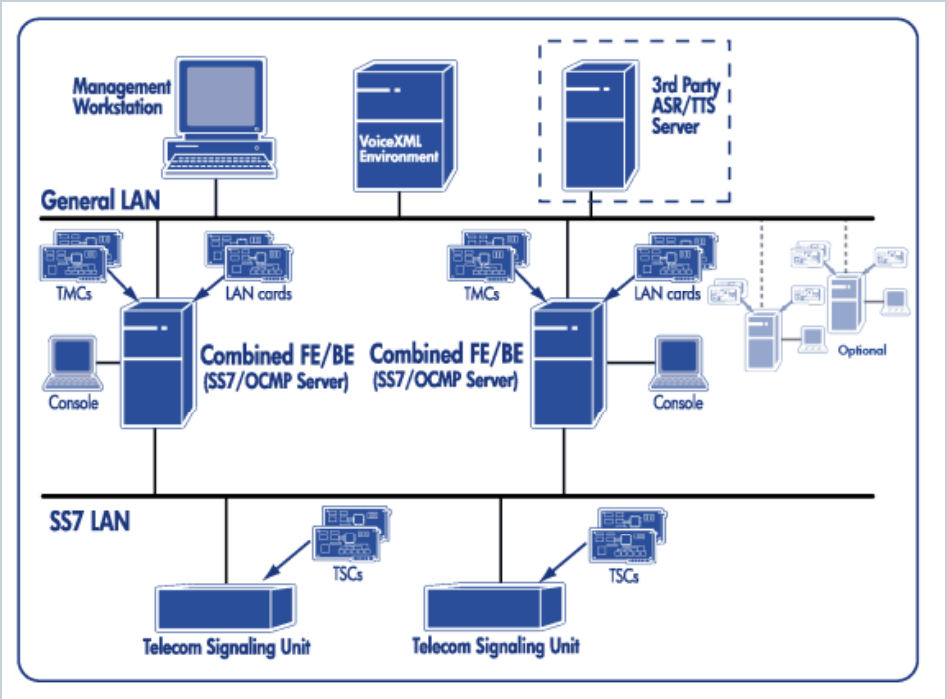
VOICE XML: Arquitectura



http://www.ewingstech.com/ewings/user/web/solutions.php?as=NETVOICE

VOICE XML: Arquitectura

http://docs.hp.com/en/J7170-90018/ch01s03.html



VOICE XML: Conceptos básicos

- Documento VXML = máquina de estados.
- Diálogos:
  - FORMS: Recoge los valores de una serie de campos.
  - MENUS: Conjunto de opciones y transiciones a otros diálogos.
- Subdiálogos = llamadas a funciones.
- Aplicaciones, Sesiones, Gramáticas, Eventos, Enlaces.

Conceptos sobre VoiceXML

- Sesión: una sesión comienza cuando usuario empieza a interactuar con el intérprete de VoiceXML
- Gramática: vocabulario y frases permitidas en cada estado. Un estado puede tener una o más gramáticas asociadas
- Eventos: pueden ser generados por la plataforma por varias razones (p. e. usuario no responde, no responde correctamente, solicita ayuda, existen errores en documento, etc.)
- Enlaces: especifican transiciones a otros puntos del documento, otro documento dentro de la aplicación, u otro documento de otra aplicación

VOICE XML: Elementos

Elemento	Funcionalidad
<assign>	Asigna un valor a una variable
<audio>	Reproduce un clip de audio con un prompt
<block>	Contenedor (no interactivo) de código ejecutable.
<catch>	Captura un evento
<choice>	Define un ítem del menú.
<clear>	Borra una o varias variables del form
<disconnect>	Desconecta una sesión
<else>	Estructuras <if>
<elseif>	Estructuras <if>
<enumerate>	Abreviatura para enumerar las opciones de un menú
<error>	Captura un evento error
<exit>	Finaliza la sesión
<field>	Declara un campo de entrada en el form

VOICE XML: Elementos

Elemento	Funcionalidad
<filled>	Acción a ejecutar cuando se completen los campos
<form>	Diálogo para presentar información y recoger datos
<goto>	Ir a otro diálogo en el mismo o diferente documento
<grammar>	Especifica el reconocimiento de voz o la grammar DTMF
<help>	Captura un evento ayuda
<if>	Logica condicional
<initial>	Declara código inicial antes de entrar en el form
<link>	Especifica una transición válida para todos los diálogos en su alcance
<log>	Genera un mensaje de depuración.
<menu>	Diálogo para seleccionar entre varias alternativas
<meta>	Define un ítem de metadata ítem en formato nombre / valor
<metadata>	Define información metadata usando el esquema metadata
<noinput>	Captura el evento no-entrada

VOICE XML: Elementos

Elemento	Funcionalidad
<nomatch>	Captura el evento no-coincidencia
<object>	Definir extensiones a medida
<option>	Especifica una opción en un <field>
<param>	Parámetro en un <object> o <subdialog>
<prompt>	Salida de audio para el usuario
<property>	Propiedades de la plataforma de implementación
<record>	Graba una muestra de audio
<reprompt>	Reproduce un prompt cuando es revisitado tras un evento
<return>	Retorno desde un subdiálogo
<script>	Bloque de código ECMAScript
<subdialog>	Invoca a un diálogo como subdiálogo del actual.
<submit>	Suministra valores a otro documento del servidor
<throw>	Lanza un evento

VOICE XML: Elementos

Elemento	Funcionalidad
<transfer>	Transfiere la llamada a otro destino
<value>	Inserta el valor de una expresión en un prompt.
<var>	Declara una variable
<vxml>	Elemento de mayor jerarquía en un documento VoiceXML

Conceptos sobre VoiceXML - Variables

- Declaración

```
<var name="telefono"/>
<var name="telefono" expr="6305551212"/>
<var name="y" expr="document.z+1"/>
<var name="ciudad" expr="'Valencia"/>
```

Tiene el valor especial **undefined**

- Asignación

```
<assign name="flavor" expr="'chocolate'"/>
<assign name="document.mycost" expr="document.mycost+14"/>
```

- Liberar valor de variables

```
<clear namelist="city state zip"/>
```

Si no se especifica ningún campo, se liberan todos los campos del formulario

VOICE XML: Estructura y Ejecución

```
<?xml version="1.0" encoding="UTF-8"?>
 <vxml xmlns="http://www.w3.org/2001/vxml"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.w3.org/2001/vxml
   http://www.w3.org/TR/voicexml20/vxml.xsd" version="2.0">
<form>
<field name="drink">
<prompt>¿Le gustaría tomar café, té, leche o nada?</prompt>
<grammar src="bebidas.grxml" type="application/srgs+xml"/>
</field>
<block>
<submit next="http://www.drink.example.com/bebida2.asp"/>
</block>
</form>
</vxml>
```

DOCUMENTO

C (computer): ¿Le gustaría tomar café, té, leche o nada?

H (human): Zumo de naranja.

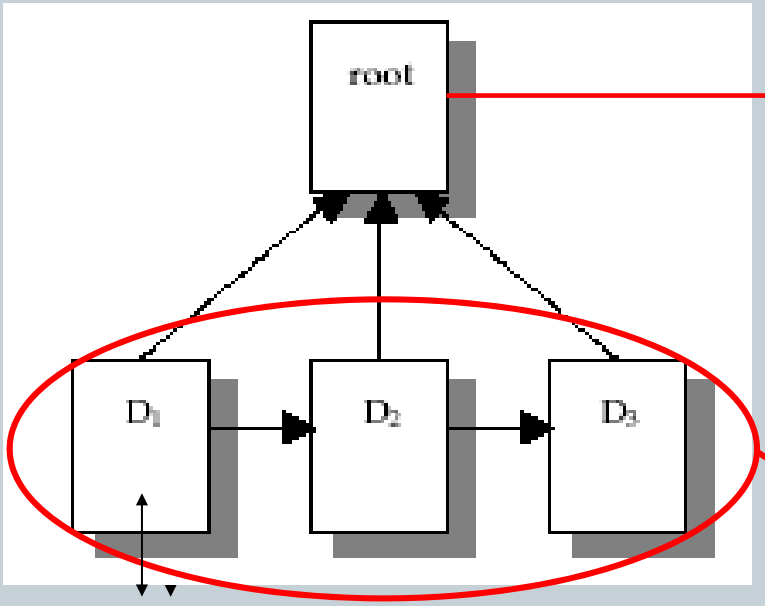
C: Lo siento, no le he entendido. (mensaje específico del sistema.)

C: ¿Le gustaría tomar café, té, leche o nada??

H: Té.

C: (continua en el documento bebida2.asp)

VOICE XML: Estructura y Ejecución



- Sus variables son accesibles desde cualquier documento.
- Sus gramáticas pueden estar activas en todo momento.

Comparten el documento raíz

Siempre que el usuario interactúa con un documento, su documento raíz está cargado en memoria

VOICE XML: Estructura y Ejecución

ROOT
app-root.vxml

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml xmlns="http://www.w3.org/2001/vxml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/vxml
   http://www.w3.org/TR/voicexml20/vxml.xsd"
   version="2.0">
 <var name="bye" expr="'Ciao'"/>
 <link next="operator_xfer.vxml">
   <grammar type="application/srgs+xml" root="root" version="1.0">
     <rule id="root" scope="public">operator</rule>
  </grammar>
 </link>
</vxml>
```

LEAF
leaf.vxml

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml xmlns="http://www.w3.org/2001/vxml"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.w3.org/2001/vxml
   http://www.w3.org/TR/voicexml20/vxml.xsd"
   version="2.0" application="app-root.vxml">
 <form id="say_goodbye">
  <field name="answer">
     <grammar type="application/srgs+xml" src="/grammars/boolean.grxml"/>
```

APLICACIÓN

VOICE XML: Estructura y Ejecución

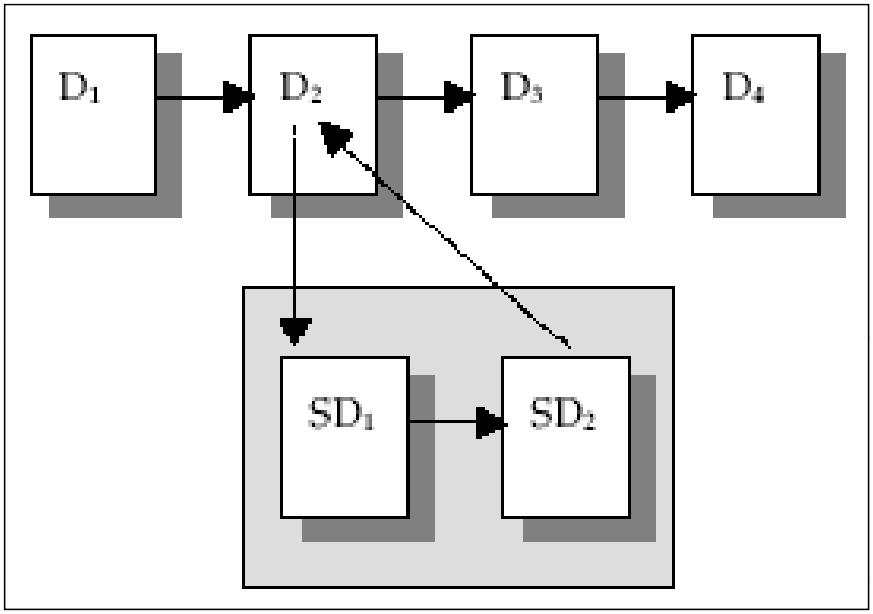
LEAF
leaf.vxml

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml xmlns="http://www.w3.org/2001/vxml"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.w3.org/2001/vxml
   http://www.w3.org/TR/voicexml20/vxml.xsd"
   version="2.0" application="app-root.vxml">
 <form id="say_goodbye">
  <field name="answer">
     <grammar type="application/srgs+xml" src="/grammars/boolean.grxml"/>
     <prompt>Shall we say <value expr="application.bye"/>?</prompt>
     <filled>
       <if cond="answer">
        <exit/>
       </if>
       <clear namelist="answer"/>
     </filled>
  </field>
 </form>
</vxml>
```

APLICACIÓN

Formularios - Subdiálogos

- SUBDIALOG ≈ llamadas a funciones. Regreso al punto de llamada



```
<!– formulario que realiza llamada a subdiálogo -->
<form>
…
  <subdialog name="resultado" src="#obtener_carnet_conducir">
    <param name=“dia_nacimiento" expr="'02-10-2000'"/>
    <filled>
      <submit next=http://mi_servicio.ejemplo/cgi-bin/proceso namelist=“carnet status”/>
    </filled>
  </subdialog>
</form>

…

<!– subdiálogo para obtener el carnet de conducir -->
<form id=“obtener_carnet_conducir">
  <var name=“dia_nacimiento"/>

  <field name=“carnet">
    <grammar src="http://grammarlib/carnet_conducir.gram" type="application/x-jsgf"/>
    <prompt> Por favor, diga el número de su carnet de conducir. </prompt>
    <filled>
      <if cond=“carnet_valido(carnet,dia_nacimiento)">
        <var name="status" expr="true"/>
      <else/>
        <var name="status" expr="false"/>
      </if>
      <return namelist=“carnet status"/>
    </filled>
  </field>
</form>
```

Paso de parámetro al subdiálogo

Definición de variable de entrada al subdiálogo

Devolución de variables al documento que realiza la llamada

VOICE XML: Estructura y Ejecución

PRINCIPAL
app.vxml

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml xmlns="http://www.w3.org/2001/vxml"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.w3.org/2001/vxml
   http://www.w3.org/TR/voicexml20/vxml.xsd"
   version="2.0">
  <form id="billing_adjustment">
    <var name="account_number"/>
    <var name="home_phone"/>
    <subdialog name="accountinfo" src="acct_info.vxml#basic">
      <filled>
        <!-- Note the variable defined by "accountinfo" is
          returned as an ECMAScript object and it contains two
          properties defined by the variables specified in the
          "return" element of the subdialog. -->

        <assign name="account_number" expr="accountinfo.acctnum"/>
        <assign name="home_phone" expr="accountinfo.acctphone"/>
      </filled>
    </subdialog>

    <field name="adjustment_amount">
     <grammar type="application/srgs+xml" src="/grammars/currency.grxml"/>
      <prompt>
        What is the value of your account adjustment?
      </prompt>
      <filled>
        <submit  next="/cgi-bin/updateaccount"/>
      </filled>
    </field>
  </form>
</vxml>
```

SUBDIÁLOGO

VOICE XML: Estructura y Ejecución

SUBDIALOG
acct_info.vxml

SUBDIÁLOGO

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml xmlns="http://www.w3.org/2001/vxml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.w3.org/2001/vxml
   http://www.w3.org/TR/voicexml20/vxml.xsd"
   version="2.0">
  <form id="basic">
    <field name="acctnum">
      <grammar type="application/srgs+xml" src="/grammars/digits.grxml"/>
      <prompt> What is your account number? </prompt>
    </field>
    <field name="acctphone">
      <grammar type="application/srgs+xml" src="/grammars/phone_numbers.grxml"/>
      <prompt> What is your home telephone number? </prompt>
      <filled>
        <!-- The values obtained by the two fields are supplied
          to the calling dialog by the "return" element. -->
        <return namelist="acctnum acctphone"/>
      </filled>
    </field>
  </form>
</vxml>
```

VOICE XML: Forms

- Componente fundamental de los documentos VXML.
- Contienen:
  - Campos de entrada (items) y de control.
  - Declaración de variables.
  - Tratamiento de eventos.
  - Acciones a ejecutar cuando se completen determinados campos.
- Asociación con variables.
- Algoritmo de interpretación.
- Directos y mixtos.

VOICE XML: Forms Items

- Campos de entrada:
  - **FIELD:** Campos de entrada (items) y de control.
  - **FILLED:** Acción a ejecutar cuando se completan campos.
  - **TRANSFER:** Conectar al usuario a otra entidad.
  - **RECORD**: Almacenar grabaciones del usuario.
- Campos de control:
  - **BLOCK:** Contenido del sistema para presentar los campos.
  - **INITIAL:** Presentación del Form.

VOICE XML: Forms

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/vxml
   http://www.w3.org/TR/voicexml20/vxml.xsd">
<form id="weather_info">
 <block>Welcome to the weather information service.</block>
 <field name="state">
  <prompt>What state?</prompt>
  <grammar src="state.grxml"  type="application/srgs+xml"/>
  <catch event="help">
     Please speak the state for which you want the weather.
  </catch>
 </field>
 <field name="city">
  <prompt>What city?</prompt>
  <grammar src="city.grxml" type="application/srgs+xml"/>
  <catch event="help">
     Please speak the city for which you want the weather.
  </catch>
 </field>
 <block>
  <submit next="/servlet/weather" namelist="city state"/>
 </block>
</form>
</vxml>
```

C (computer): Welcome to the weather information service. What state?
H (human): Help
C: Please speak the state for which you want the weather.
H: Georgia
C: What city?
H: Tblisi
C: I did not understand what you said. What city?
H: Macon
C: The conditions in Macon Georgia are sunny and clear at 11 AM ...

Formularios – Variables ocultas

- Variables “escondidas” del nombre de un campo:
  - **name$.confidence**: valor de confianza en el reconocimiento del campo: **0.0 – 1.0** (0.0 es el menor valor, 1.0 es el mayor valor)
  - **name$.utterance**: cadena de palabras reconocidas (en el formato proporcionado por el usuario)
  - **name$.inputmode**: modo en que fue proporcionada la entrada del usuario (*dtmf* o *voice*)

Formularios – Ej. Variables ocultas

```
<field name="numero_telefono" type="phone">
   <prompt> ¿Cuál es su número de teléfono? </prompt>
</field>

<field name="confirmacion_telef" type="boolean" cond="
numero_telefono$.confidence < 0.6">

   <prompt> Ha dicho <value expr="numero_telefono" /> ?
   </prompt>

   <prompt> Ha dicho <value expr="numero_telefono$.utterance"/> ?
   </prompt>

   <filled>
      <if cond="!confirmacion_telef">
         <clear namelist="numero_telefono"/>
      </if>
   </filled>
</field>
```

La confirmación del nº de teléfono se realiza sólo si el valor de confianza obtenido es < 0.6

El nº de teléfono introducido se reproduce dígito a dígito (p. e. **9 5 8 1 2 3 4 5 6**)

Si usuario no confirma, nº de teléfono se le solicita de nuevo

El nº de teléfono introducido se reproduce respectando formato usado por usuario (p. e. **9 5 8 12 34 56**)

Form Interpretation Algorithm (FIA)

1. Fase iniciación
   1.1 Inicialización contadores de prompts de variables (puestos a 1)
   1.2 Cada variable se inicializa a **undefined** o al valor del atributo **expr**

2. Bucle principal
   2.1 Fase de selección: seleccionar siguiente ítem a visitar
   2.2 Fase de obtención: obtener entrada del usuario
   2.3 Fase de procesamiento: procesar entrada (o evento generado durante fase de obtención, p. e. el usuario no dijo nada, no se entendió lo que dijo, solicitó ayuda, etc.)

Form Interpretation Algorithm (2)

- El FIA puede ser controlado de diversas formas para alterar orden de visita de campos del formulario:
  - **Asignar** valor al ítem de la variable – el ítem no será seleccionado
    - *Ej. <assign name="ciudad_origen" expr="true"/>*
  - Usar **<clear>** – pone ítem como **undefined**, forzando que éste pueda ser visitado
    - *Ej. <clear namelist="ciudad_origen ciudad_destino"/>*
  - Usar **<goto ...>** – especifica explícitamente el siguiente ítem a visitar
    - *Ej. <goto nextitem="confirmar_salida"/>*
  - Condiciones de **guarda**:

```
<!-- obtencion ciudad destino -->
    <field name="ciudadDestinoR" cond=" ciudadDestinoG == undefined">
    <prompt>Diga el nombre de la ciudad a la que quiere viajar</prompt>
    <grammar src="ciudades.jsgf"/>
    </field>
```

Este campo sólo puede ser visitado si se cumple la condición **ciudadDestinoG = undefined**

Estrategia de interacción

- Dirigida por sistema
  - La más simple: campos del formulario visitados de uno en uno, en orden secuencial (sólo se rellena un campo en cada interacción)
  - Gramáticas de voz y/o DTMF sólo activas en estado visitado
- Mixta
  - Gramáticas de determinados estados pueden estar activas cuando interacción está en otro estado del documento o de la aplicación
  - Si usuario pronuncia frase permitida por otra gramática, ejecución continúa en el otro estado
  - Gran flexibilidad ...

Estrategia de interacción

```
<form id="informacion_meteorologica">
  <block>Bienvenido a este servicio automático de información
         meteorológica.</block>
  <field name="provincia">
    <prompt>¿En qué provincia?</prompt>
    <grammar src="provincia.gram" type="application/x-
jsgf"/>
    <catch event="help">
      Por favor, diga el nombre de la provincia en la que desea
      conocer el estado del tiempo.
    </catch>
  </field>
  <field name="ciudad">
    <prompt>¿En qué ciudad?</prompt>
    <grammar src="ciudad.gram" type="application/x-jsgf"/>
    <catch event="help">
      Por favor, diga el nombre de la ciudad en que desea
      conocer el estado del tiempo.
    </catch>
  </field>
  <block>
    <submit next="/servlet/prevision_meteorologica"
     namelist="ciudad provincia"/>
  </block>
</form>
```

S: Bienvenido a este servicio automático ...
¿En qué
provincia?
U: ayuda
S: Por favor, diga el nombre de la provincia en la ...
U: Granada
S: ¿En qué ciudad?
U: Madrid
S: No he comprendido. ¿En qué ciudad?
U: Loja
S: El tiempo en Loja es soleado a las 12 AM
...

Formularios de iniciativa mixta

- Sistema de diálogo y usuario pueden dirigir conversación
- Debe haber una o más etiquetas **<initial>**, y una o más gramáticas a nivel de form
- Si hay gramáticas a nivel de form:
  - Los campos pueden ser rellenados en cualquier orden
  - Mediante una misma frase se puede rellenar más de un campo
- Gramáticas del form pueden estar activas cuando usuario está en otros diálogos
- Ejemplo:
  - Un documento tiene dos forms: alquiler coche y reserva hotel
  - Ambos forms tienen gramáticas activas para el documento
  - Usuario pueden proporcionar información de reserva hotel cuando sistema solicita información alquiler coche

Filled

<filled> se usa para decidir qué hacer cuando se rellenan campos mediante entrada del usuario. Dos posibilidades:

- ***hijo de elemento <form>***: acción cuando se rellena uno o más campos
- ***hijo de elemento <field>***: acción cuando se rellena el campo

Filled

```
<form id="obtener_ciudades_origen_destino">
  <field name="ciudad_origen">
    <grammar src="http://www.gramaticas.ejemplo/voicexml/ciuda
    <prompt>¿Desde qué ciudad desea salir?</prompt>
  </field>
  <field name="ciudad_destino">
    <grammar src="http://www.gramaticas.ejemplo/voicexml/ciuda
    <prompt>¿A qué ciudad quiere viajar?</prompt>
  </field>
  <filled mode="all" namelist="ciudad_origen ciudad_destino">
    <if cond="ciudad_origen == ciudad_
      <prompt>La ciudad de salida no p
      ser igual que la de destino.</prom
      <clear/>
    </if>
  </filled>
</form>
```

<filled> hijo de form

Valores del atributo **mode**:
"all" (por defecto) – acción se ejecuta cuando todos campos rellenos, y al menos, uno relleno mediante última entrada del usuario
"any" – acción se ejecuta cuando entrada rellena al menos un campo

```
<form id="obtener_ciudad">
  <field name="ciudad">
    <grammar
src="http://www.gramaticas.ejemplo/ciudades.gram"/>
    <prompt>¿Cómo se llama la ciudad?</prompt>
    <filled>
      <if cond="ciudad == 'Loja'">
        <prompt> El servicio a Loja ha sido
interrumpido.</prompt>
      </if>
    </filled>
  </field>
</form>
```

<filled> hijo de field

VOICE XML: Menús

- Opciones y transición.
- Mismos elementos que el FORM.
- Opciones → <choice>.
- Grammars: propiedades *exact* y *approximate.*
- Enumerar opciones con <enumerate>.

VOICE XML: Menús

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/vxml
   http://www.w3.org/TR/voicexml20/vxml.xsd">
<menu>
  <prompt>
    Welcome home. Say one of: <enumerate/>
  </prompt>
  <choice next="http://www.sports.example.com/vxml/start.vxml">
     Sports
  </choice>
  <choice next="http://www.weather.example.com/intro.vxml">
     Weather
  </choice>
  <choice next="http://www.stargazer.example.com/voice/astronews.vxml">
     Stargazer astrophysics news
  </choice>
  <noinput>Please say one of <enumerate/></noinput>
</menu>
</vxml>
```

C: Welcome home. Say one of: sports; weather; Stargazer astrophysics news.
H: Astrology.
C: I did not understand what you said. (a platform-specific default message.)
C: Welcome home. Say one of: sports; weather; Stargazer astrophysics news.
H: sports.
C: (proceeds to http://www.sports.example.com/vxml/start.vxml)

```
<menu>
  <prompt>Bienvenido a casa. Elige una de las siguientes opciones:
<enumerate/></prompt>
  <choice next="http://www.deportes.ejemplo/vxml/start.vxml">
    Deportes </choice>
  <choice next="http://www.previsiones.ejemplo/intro.vxml">
    Parte meteorológico </choice>
  <choice next="http://www.llevant.ejemplo/voice/start.vxml">
    Noticias Llevant UE</choice>

</menu>
```

El usuario puede decir cualquier subconjunto de las palabras, en el mismo orden, p.e.: “Noticias”, “Noticias Real”, “Real Madrid”, etc.

```
  <prompt>
    Para deportes pulse 1, para parte meteorológico pulse 2, para noticias real
madrid pulse 3.
  </prompt>
  <choice dtmf="1" next="http://www.deportes.ejemplo/vxml/start.vxml"/>
    <prompt>
      <audio=“http://www.deportes.ejemplo/voice/bienvenida_deportes.wav”>
          Bienvenido a la <emp> sección de deportes </emp>
      </audio>
    </prompt>
  </choice>
  <choice dtmf="2" next="http://www.previsiones.ejemplo/intro.vxml"/>
  <choice dtmf="3" next="http://www.real-madrid.ejemplo/voice/start.vxml"/>
</menu>
```

Énfasis

Si no se puede reproducir el mensaje del fichero .wav, se reproduce el mensaje en texto

VOICE XML: Grammar

- Elemento <grammar>.
- De voz y de DTMF.
- Sus funciones son:
  - Pronunciaciones que el usuario debe mencionar.
  - Interpretación semántica correspondiente.
- SRGS XML, ABNF.
- Internas y externas.
- Jerarquía a través de pesos.
- Diferentes ámbitos.

VOICE XML: Grammar

```
<grammar mode="voice" xml:lang="en-US" version="1.0" root="command">
  <!-- Command is an action on an object -->
  <!-- e.g. "open a window" -->
  <rule id="command" scope="public">
    <ruleref uri="#action"/> <ruleref uri="#object"/>
  </rule>

  <rule id="action">
    <one-of>
      <item> open </item>
      <item> close </item>
      <item> delete </item>
      <item> move </item>
    </one-of>
  </rule>

  <rule id="object">
   <item repeat="0-1">
      <one-of> <item> the </item> <item> a </item> </one-of>
    </item>
    <one-of>
      <item> window </item>
      <item> file </item>
      <item> menu </item>
    </one-of>
  </rule>
</grammar>
```

VOICE XML: Grammar

```
<grammar mode="voice" type="application/srgs">
#ABNF 1.0;
language en-US;
mode voice;
root $command;
  public $command = $action $object;
  $action = open | close | delete | move;
  $object = [the | a] (window | file | menu);
</grammar>
```

GRAMMAR
SRGS - ABNF

VOICE XML: Salida del Sistema

- **PROMPTS:**
  - Voz sintetizada.
  - Archivos de audio.
  - Propiedades: bargein, paragraph, phoneme, phrase, currency, value, time-out.

VOICE XML: Control de flujo

- **Variables:**
  - Equivalencia con ECMAScript.
  - Alcance según la declaración.
- **Tratamiento de eventos.**
  - Throw y Catch.
  - Selección del capturador
- **Contenido ejecutable.**
  - VAR, ASSIGN, CLEAR, IF, ELSEIF, ELSE, GOTO, RETURN, DISCONNECT, SCRIPT, LOG.

VOICE XML: Contexto y Recursos

- **Búsqueda de recursos:**
  - Parámetros fundamentales.
  - Caché de archivos.
  - Prebúsquedas.
- **Información METADATA .**
  - Etiquetas <META>, <METADATA>.
- **Propiedades:**
  - Etiqueta <property>.
  - Propiedades específicas del reconocedor de voz.
  - Propiedades específicas del reconocedor de DTMF.
  - Prompt.
- **Parámetros:**
  - Etiqueta <param>.
  - Subdiálogos.

VOICE XML: Herramientas

- **Implementación de la norma por las empresas del consorcio.**
- **Servidores de hospedaje gratuito:**
  - BeVocal Café.
  - VoiceGenie.
  - TellMe.
  - SALT.
- **Entornos de escritorio (IDE):**
  - IBM Voice WebSphere Toolkit.
  - Proyectos experimentales (ELVIRA).

VOICE XML: Herramientas

Be Vocal Café

- N mejores resultados.
- Múltiples intervenciones.
- Dependiente del locutor.

Nuance Voice Platform

- Independiente locutor.
- Ambientes ruidosos.
- Experiencia del locutor.
- Detección idioma.
- Entrenamiento local.

Loquendo Café

- Independiente del locutor.
- Ambientes ruidosos.
- Vocabulario abierto.
- Optimizado línea telefónica.
- Modelos acústicos entrenados con grandes corpus.
- Modelos Markov y redes neuronales.
- Variabilidad regional.

IBM Voice ToolKit

- Prerrequisitos:
  - Windows 2000 con Service Pack 3.
  - WebSphere Studio Site Developer v5.0.1, WebSphere Studio Application Developer v5.0.1, WebSphere Studio Enterprise Developer v5.0.1, WebSphere Application Server – Express v5.0.1.
  - Pentium 500 MHz, 768 MB RAM, 1 GB disco duro.

- Funcionalidades:
  - Editor y depurador de Voice XML.
  - Editor y herramienta de test de grammars (XML y ASBN).
  - Reconocedor de voz.
  - Creación de pronunciaciones.
  - Conversor de texto a voz.
  - Grabación de voz.

IBM Voice ToolKit

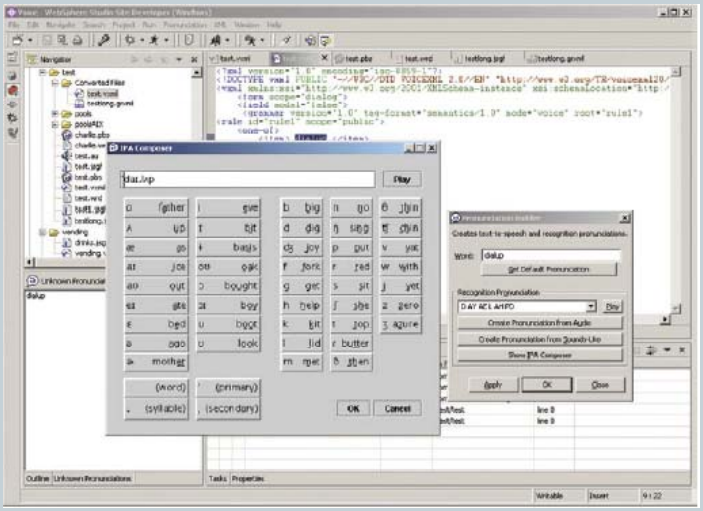
- Prerrequisitos
  - Mínimo Service Pack 2.
  - Máquina virtual de Java (J2RE).

- Software requerido
  - WebSphere Studio Site Developer v5.0.1, WebSphere Studio Application Developer v5.0.1.
  - IBM Server SDK 1.5: Voice browser.
    - Selección de idioma
    - TTS y CTTS.

- Instalación
  - Instalar en el orden correcto.
  - Comprobar instalación paso a paso.
  - Configurar rutas de acceso.

IBM Voice ToolKit

- Paquetes descargables (free download).
- Soporte en español.
- Reconocedor de voz y de CTTS en español.
- Componentes:
  - Editor de Voice XML.
  - Editor de Grammars.
  - Constructor de pronunciaciones.
  - Grabadora de audio.
  - RDC WIZARD.
  - Componentes de diálogo reutilizables (español).
  - Depurador de aplicaciones
  - Recursos y ayuda.
  - Herramientas de análisis.



IBM Voice ToolKit

- Dependiente del usuario.
- Modelos acústicos adaptados línea telefónica.
- Grandes diccionarios de palabras.
- Especificaciones de VoiceXML.
- Herramientas disponibles:
  - Creación de pronunciaciones para vocabulario desconocido.
  - Añadir entradas a diccionarios o crear propios.
  - Codificación en base a fonemas IPA.

IBM Voice ToolKit

Constructor de pronunciaciones

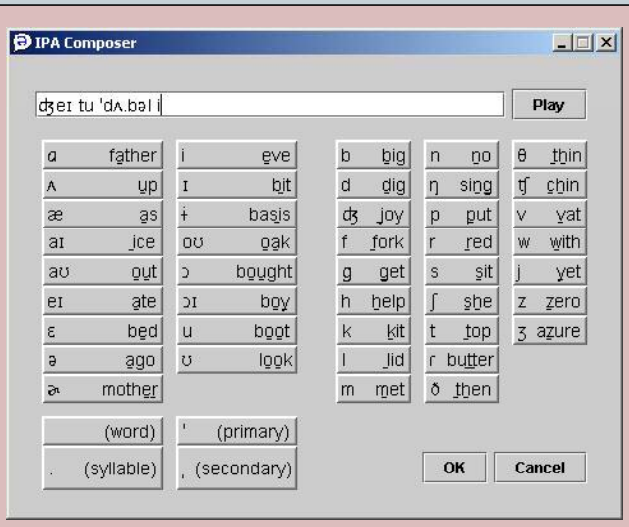


Tabla de fonemas de la IPA

Aplicación práctica: Centralita Telefónica

GRAMÁTICA PRINCIPAL

```
#JSGF V1.0;
grammar centralita;
public <centralita> ="por favor" <peticion> | <peticion> ["por favor"];
 <nombremasc> = pepe | juan | josé | luis | víctor | andrés | guillermo | paulo | antonio | pablo | raul | germán | david | miguel
     | carlos | mario | felipe | manuel | manolo | mariano | nacho | ignacio | jorge;
 <nombrefem> = maría | ana | luisa | isabel | paz | antonia | lidón | lledó;
 <nombre> = (<nombremasc> | <nombrefem>);
 <apellido> = martínez | lópez | jímenez | marzal | vilá | peris | castellanos | aibar | prat | castaño | valls | amengual |
     montoliu | sanz | gómez | aliaga | fabregat | porcar | varó | pelayo | toledo | lobo | climent | ventura | garcía | ibáñez |
     palomar | llorens | vilar | zarco | yagüe | llopis | espósito | badenas | monfort | granada | vizcaíno | iborra;
 <persona> = ([el señor] (<nombremasc> [<apellido>] | [<nombremasc>] <apellido>)) | ([la señora] (<nombrefem>
     [<apellido>] | [<nombrefem>] <apellido>));
 <digito> = cero | uno | dos | tres | cuatro | cinco | seis | siete | ocho | nueve;
 <numero> = <digito> [<digito> [<digito> [<digito>]]];
 <extension> = ( el número | la extensión | el | el teléfono ) <numero>;
 <pedirhablar> = (deseo | desaría | me gustaría | quisiera | quiero | quería) hablar;
 <pedirpasar> = póngame | me pone | me pasa | páseme | puede ponerme | puede pasarme | me pasaría;
 <pedir> = <pedirhablar> | <pedirpasar>;
 <pedirextension> = <pedir> con <extension>;
 <pedirpersona> = <pedir> con <persona> | se puede poner <persona> | <pedir> con el secretario de <persona> | <pedir>
     con la secretaria de <persona> | <pedir> con el jefe de <persona> | <pedir> con la jefa de <persona> | <pedir> con el
     despacho de <persona> | <pedirpasar> con la extensión de <persona>;
 <peticion> = <pedirextension> | <pedirpersona>;
```

Aplicación práctica: Centralita Telefónica

GRAMÁTICA AFIRMACIONES Y NEGACIONES

```
#JSGF V1.0;
 grammar confirma;
 public <confirma> = <si> | <no>;
 <si>= si | vale | "es correcto" | bien | correcto;
 <no>= no | mal | "está mal" | incorrecto;
```

PROGRAMA PRINCIPAL

```
 <?xml version="1.0" encoding="iso-8859-1"?>
 <vxml version="1.0" xml:lang="es-ES"  xmlns="http://www.w3.org/2001/vxml">
 <meta name="GENERATOR" content="Voice Toolkit for WebSphere Studio"/>
 <form>
    <var name="horas"/>
    <var name="minutos"/>
    <var name="saludo"/>
    <block><script>
       var d = new Date();
       horas = d.getHours();
       minutos = d.getMinutes();
       saludo="Buenos días.";
       if ('horas /> 14') saludo="Buenas tardes.";
       if ('horas />20') saludo="Buenas noches.";
      </script></block>
```

Aplicación práctica: Centralita Telefónica

PROGRAMA PRINCIPAL

```
    <field name="frase">
    <grammar src="centralita.jsgf"></grammar>
    <filled>
       <if cond="frase =='salir'">
         <prompt>Gracias por utilizar la centralita de nuestra empresa. Adiós.</prompt>
         <exit></exit>
       </if>
       <if cond="frase =='ayuda'">
         <prompt>Por favor, indíqueme con que persona o extensión desea hablar</prompt>
       </if>  </filled>   </field>
   <field name="confirma">
      <prompt> He entendido: <value expr="frase"/> ¿es correcto?</prompt>
      <grammar src="confirma.jsgf"></grammar>
      <filled>
        <if cond="confirma =='correcto'">     <audio src="audio.wav"></audio>    </if>
       <if cond="confirma =='no'">
         <prompt>Por favor, indíqueme con que persona o extensión desea hablar</prompt>
         <clear namelist="frase"></clear>
         <clear namelist="confirma"></clear>
       </if>  </filled>  </field>  </form>
 </vxml>
```