


# **Nuevos Paradigmas de Interacción**

## **Tutorial de VoiceXML**

**Prof. Ramón López-Cózar Delgado**  
**Dpto. Lenguajes y Sistemas Informáticos**  
**ETS Informática y Telecomunicación**  
**Universidad de Granada**  
**<http://lsi.ugr.es/rlopezc>**



# Puntos a tratar

- 
- Introducción a VoiceXML
  - Arquitectura de VoiceXML
  - Conceptos sobre VoiceXML

# Introducción a VoiceXML

- 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

# Introducción a VoiceXML

- VoiceXML (o VXML)
  - Estándar basado en XML desarrollado por el W3C que permite acceder mediante habla a aplicaciones Web
  - Comunicación Sistema Diálogo → Usuarios
    - Habla sintetizada
    - Ficheros de voz pregrabados
  - Comunicación Usuarios → Sistema Diálogo
    - Habla
    - DTMF

# Introducción a VoiceXML

- Versiones de VoiceXML

**v1.0 (desarrollada en 2000)**

<http://www.w3.org/TR/voicexml>

**v2.0 (desarrollada en 2004)**

<http://www.w3.org/TR/voicexml20/>

**v2.1 (desarrollada en 2007)**

<http://www.w3.org/TR/voicexml21/>

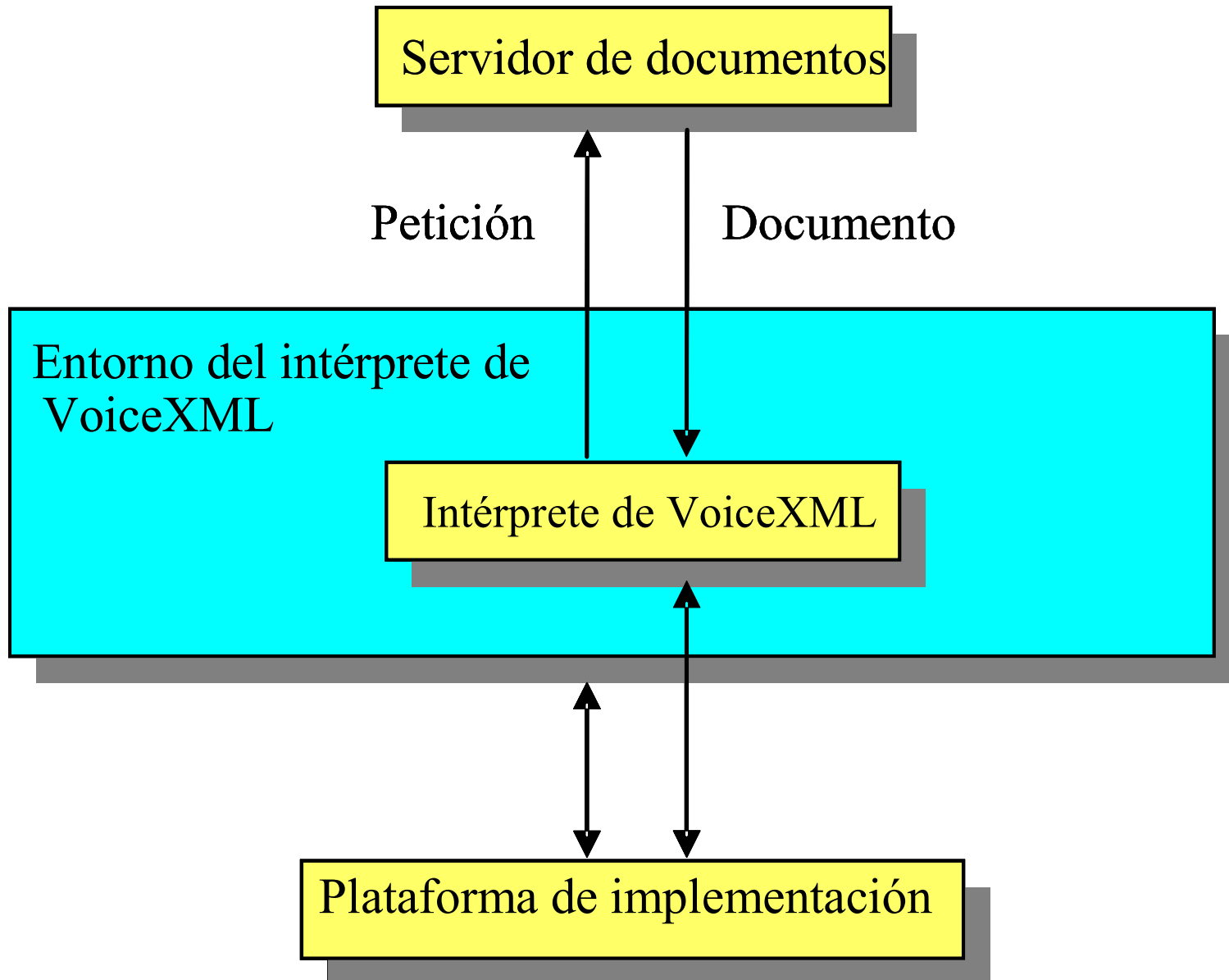
**v3.0 (desarrollada en 2010)**

<http://www.w3.org/TR/voicexml30/>

# Puntos a tratar

- Introducción a VoiceXML
- ➔ ● Arquitectura de VoiceXML
- Conceptos sobre VoiceXML

# Arquitectura de VoiceXML



# Arquitectura de VoiceXML

- **Intérprete de VoiceXML** (aplicación cliente)
  - Ejecuta lógica de aplicación
  - Genera *prompts* (turnos del sistema) y procesa respuestas del usuario
  - Busca información en sitios web para proporcionarla al usuario



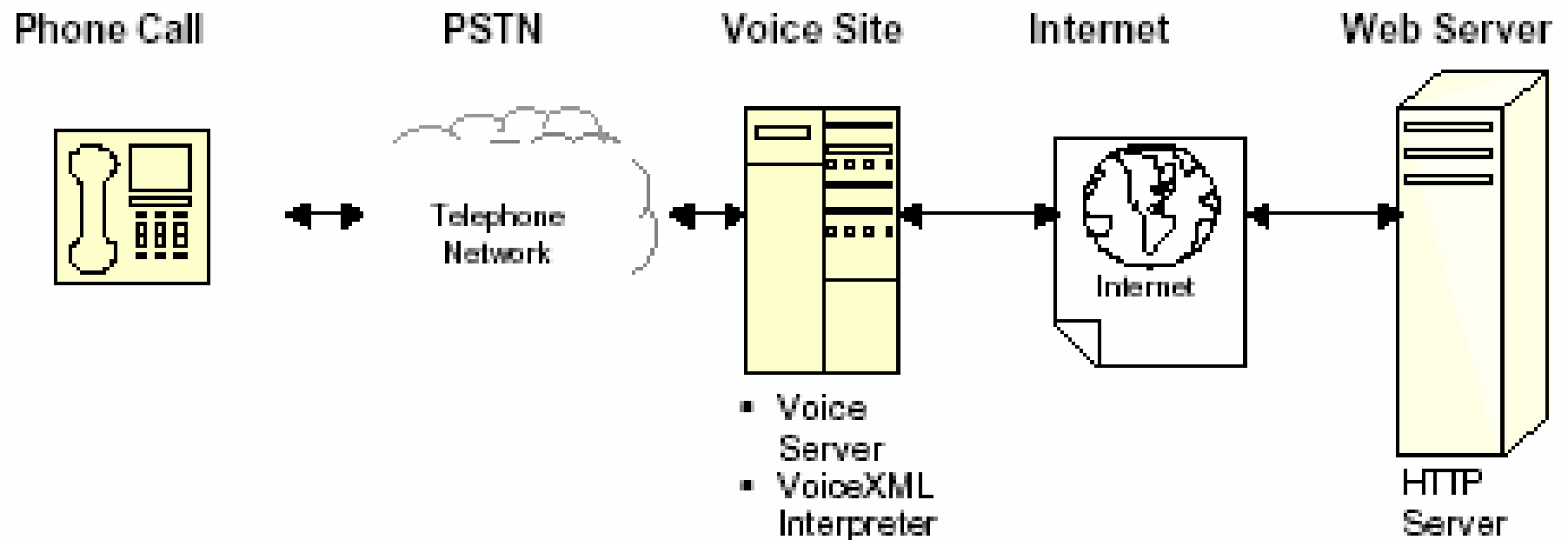
# Arquitectura de VoiceXML

- **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 llamada de usuario
  - Monitoriza entradas de usuario (ayuda, no respuesta, etc.) y genera mensajes predefinidos

# Arquitectura de VoiceXML

- **Plataforma de implementación**
  - Hardware telefónico y otros recursos
  - Genera eventos en respuesta a acciones de usuario (p.e. pulsación botones del teléfono)
  - Genera eventos del sistema (p.e. expiración temporizadores)

# Arquitectura de VoiceXML



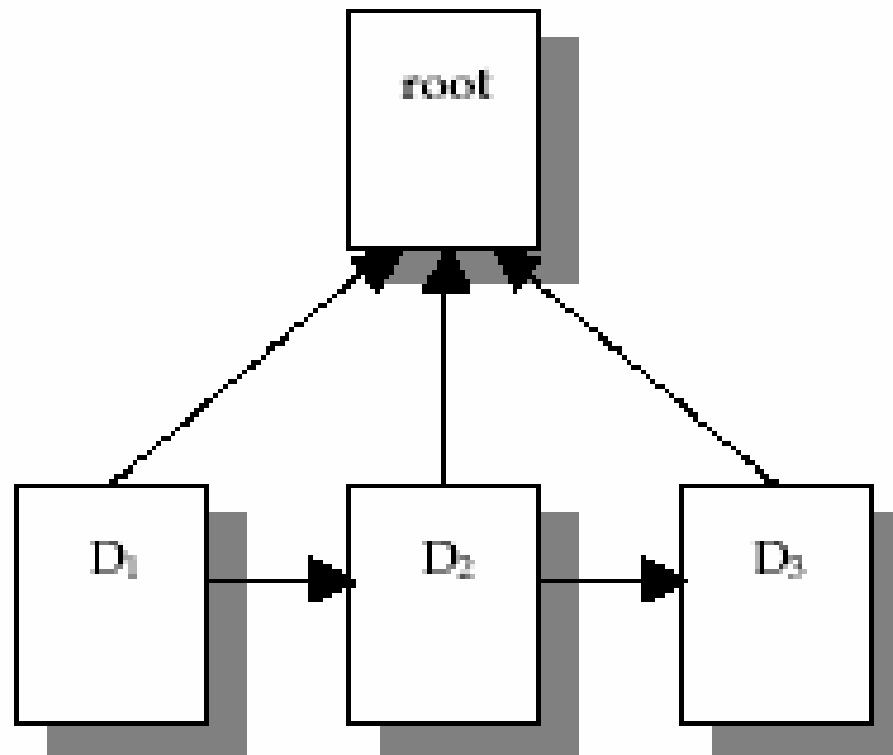
# Puntos a tratar

- Introducción a VoiceXML
- Arquitectura de VoiceXML
- ➔ ● Conceptos sobre VoiceXML

# Conceptos sobre VoiceXML

- **Aplicación**

- conjunto de documentos VoiceXML  $D_i$  que comparten el documento raíz (root) de la aplicación



# Conceptos sobre VoiceXML

- **Aplicación**

- Siempre que usuario interactúa con un documento, su documento raíz está siempre cargado en memoria
- El documento raíz se quita de memoria cuando se realiza transición a documento que no pertenece a la aplicación
- Las variables del documento raíz son accesibles desde cualquier documento hoja de la aplicación
- Las gramáticas del documento raíz pueden estar activas en todo momento, independientemente del documento hoja de la aplicación en que esté la interacción

# 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 varias 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 actual, otro documento dentro de la aplicación, u otro documento de otra aplicación

# Conceptos sobre VoiceXML

- Documento VoiceXML
  - Compuesto por elementos de alto nivel llamados *diálogos*
  - Tipos de diálogos
    - Formularios (*forms*): presentan información y obtienen datos de usuarios mediante campos (*fields*)
      - **id**: identificativo de formulario (su nombre)
      - Definición de variables
      - Elementos de control
      - Manejadores de eventos
      - Bloques “**filled**”: código que se ejecuta cuando se rellenan determinados campos



# Conceptos sobre VoiceXML

- Formularios (*forms*):
  - **scope**: ámbito de las gramáticas del formulario
    - Dos posibles valores:
      - “**dialog**”: sólo están activas en el formulario
      - “**document**”: están activas en cualquier diálogo del documento. Si el documento es el documento raíz, están activas en cualquier diálogo de cualquier documento de la aplicación

# Conceptos sobre VoiceXML

En prompts y gramáticas no se puede usar **tildes** ni interrogación **abierta**

Gramática activa en todo el documento

```
<vxml version="2.1" xml:lang="es-ES">
  <form id="datos" scope="document">
    <field name="ciudadDestino">
      <prompt>A que ciudad deseas viajar? </prompt>
      <grammar src="ciudades.jsgf"/>
    </field>
    <filled>
      <prompt>Has dicho <value expr="ciudadDestino"/>? </prompt>
    </filled>
  </form>
</vxml>
```

# Conceptos sobre VoiceXML

- **Menús:** presentar opciones para realizar transiciones

```
<vxml version="2.1" xml:lang="es-ES">
  <menu>
    <prompt>Bienvenido a casa. Elige una opcion: </prompt>
    <enumerate/>
      <choice next="http://www.deportes.ejemplo/vxml/start.vxml">
        Deportes </choice>
      <choice next="http://www.previsiones.ejemplo/intro.vxml">
        Parte meteorologico </choice>
      <choice next="http://www.real-madrid.ejemplo/voice/start.vxml">
        Noticias real madrid </choice>
    <enumerate/>
  </menu>
</vxml>
```

...

<menu>

<property name="inputmodes" value="voice dtmf"/>

<prompt>

Para deportes pulsa 1, para parte meteorologico pulsa 2, para noticias real madrid pulsa 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 seccion de deportes

</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>

...

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

# Conceptos sobre VoiceXML

- **Ejecución de acciones en respuesta a relleno de campos**
  - **<filled>** se usa para decidir qué hacer cuando se rellenan campos mediante entrada del usuario. Dos posibilidades:
    - hijo de elemento **<form>**: la acción se ejecuta cuando se rellena uno o más campos
    - hijo de elemento **<field>**: la acción se ejecuta cuando se rellena el campo

## Valores del atributo **mode**:

**“all”** (por defecto) – la acción se ejecuta cuando todos campos rellenos, y al menos, uno relleno mediante última entrada del usuario

**“any”** – la acción se ejecuta cuando entrada del usuario rellena al menos un campo

- **<filled>**

```
<form id="obtener_ciudades_origen_destino">
  <field name="ciudad_origen">
    <grammar src="http://www.gramaticas.ejemplo/ciudad.jsgf"/>
    <prompt>Desde que ciudad deseas salir?</prompt>
  </field>
  <field name="ciudad_destino">
    <grammar src="http://www.gramaticas.ejemplo/ciudad.jsgf"/>
    <prompt>A que ciudad quieres viajar?</prompt>
  </field>
  <filled mode="all" namelist="ciudad_origen ciudad_destino">
    <if cond="ciudad_origen == ciudad_destino">
      <prompt>La ciudad de salida no puede ser igual que la de
        destino</prompt>
      <clear/>
    </if>
  </filled>
</form>
```

**<filled> hijo de form**

- **<filled>**

```
<form id="obtener_ciudad">
  <field name="ciudad">
    <grammar src="http://www.gramaticas.ejemplo/ciudades.jsgf"/>
    <prompt>Como se llama la ciudad?</prompt>
    <bfilled>
      <if cond="ciudad == 'Sevilla' ">
        <prompt>El servicio a Sevilla ha sido interrumpido</prompt>
      </if>
    </bfilled>
  </field>
</form>
```

**<filled> hijo de field**

# Conceptos sobre VoiceXML: FIA

- **Algoritmo de interpretación de formularios (FIA)**
  - Los formularios son interpretados **de forma implícita**
  - El FIA usa un bucle para seleccionar siguiente ítem del formulario y visitarlo
  - Ítem seleccionado es el primero cuya **condición de guarda** (p.e. que el campo tenga un valor asignado) no se haya satisfecho
  - Así, si el formulario sólo tiene campos (**<field>**), el usuario será preguntado reiterativamente hasta que todos los campos sean rellenados



# Conceptos sobre VoiceXML: FIA

- **Algoritmo de interpretación de formularios (FIA)**
  - Interpretar un formulario conlleva generalmente lo siguiente
    - Seleccionar ítems y generar prompts
    - Obtener entradas del usuario (que pueden rellenar uno o más campos) o generar eventos (p.e. cuando usuario solicita ayuda)
    - Interpretar ítems **<filled>** cuando se rellenan nuevos campos

# Conceptos sobre VoiceXML: FIA

- **Algoritmo de interpretación de formularios (FIA)**
  - Finalización del FIA cuando
    - Transferencia de control (p.e. **<goto>** a otro documento o **<submit>** a un servidor de documentos)
    - Cuando no se encuentra ningún ítem que visitar en el formulario

# Conceptos sobre VoiceXML: FIA

- El FIA puede ser controlado de diversas formas para alterar orden de visita de campos del formulario
  - **Asignar valor a la variable del ítem** (así el ítem no es seleccionado)
    - Ej. `<assign name="ciudad_origen" expr="true"/>`
  - **Usar `<clear>`** (pone ítem como **undefined**, forzando que sea visitado)
    - Ej. `<clear namelist="ciudad_origen"/>`
  - **Usar `<goto ...>`** (especifica explícitamente el siguiente ítem a visitar)
    - Ej. `<goto nextitem="confirmar_salida"/>`

# Conceptos sobre VoiceXML

- Ej. Control en orden de visita de ítems de formulario
  - Ver siguientes diapositivas ...

**<link event="exit">** <grammar>adios|terminar|finalizar</grammar> </link>

**<form id=" analisis\_04\_02\_2004">**

**<catch event="exit">**

<goto nextitem="confirmar\_salida"/>

</catch>

<block>

<prompt>Hola, has sido elegido aleatoriamente para contestar a las preguntas de una encuesta</prompt>

</block>

<field name="p1" type="boolean">

<prompt>Estás de acuerdo con la postura del gobierno respecto a la guerra en Irak?</prompt>

</field>

<field name="p2" type="boolean">

<prompt>Crees que realmente existia una amenaza de armas de destruccion masiva en Irak?</prompt>

</field>

<block>

**<submit next="miServidor.miDomnio.es" namelist="p1 p2"/>**

</block>

**(Continúa ...)**

Cuando el usuario pronuncie alguna de estas palabras, se genera el evento **"exit"**

El evento "exit" se captura aquí, realizándose un goto a "confirmar\_salida"

Las respuestas se envían a un servidor de documentos

...

...

```
<field name="confirmar_salida" type="boolean">
  <prompt>Seguro que deseas terminar la encuesta?</prompt>
  <filled>
    <if cond="confirmar_salida">
      De acuerdo, adios.
      <exit/>
    <else/>
      De acuerdo, continuemos por donde nos quedamos.
      <clear namelist="confirmar_salida"/>
    </if>
  </filled>
</field>
</form>
```

Al hacer este clear, **confirmar\_salida** puede volver a ser visitado. El FIA vuelve a buscar el siguiente ítem a visitar

# Conceptos sobre VoiceXML

- Estrategias 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

# Conceptos sobre VoiceXML

- Estrategias de interacción
  - 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



```
<form id="informacion_meteorologica">
  <block>Bienvenido a este servicio de informacion meteorologica</block>
  <field name="provincia">
    <prompt>En que provincia?</prompt>
    <grammar src="provincia.jsgf"/>
    <catch event="help">
      Di el nombre de la provincia, por ejemplo, Granada
    </catch>
  </field>
  <field name="ciudad">
    <prompt>En que ciudad?</prompt>
    <grammar src="ciudad.jsgf"/>
    <catch event="help">
      Di el nombre de la ciudad, por ejemplo, Loja
    </catch>
  </field>
  <block>
    <submit next="/servlet/prevision_meteorologica" namelist="ciudad
      provincia"/>
  </block>
</form>
```

**Ej. Iniciativa dirigida por sistema**

# Conceptos sobre VoiceXML

## Ej. Iniciativa dirigida por sistema

S: Bienvenido a este servicio de informacion meteorologica.  
En que provincia?

U: ayuda

S: Di el nombre de la provincia, por ejemplo, Granada

U: Granada

S: En que ciudad?

U: Madrid

S: No he comprendido. En que ciudad?

U: Loja

S: El tiempo en Loja es soleado a las 12 AM ...

# VoiceXML: Formularios de iniciativa mixta

- Sistema de diálogo y usuario pueden tomar iniciativa 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

# VoiceXML: Formularios de iniciativa mixta

- 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
  - El usuario pueden proporcionar información de reserva de hotel cuando el sistema le solicita información de alquiler del coche

```
<form id="viajar_de_a">
```

```
  <grammar src="http://www.direcciones.ejemplo/viajar_de_a.jsfgf"/>
```

```
  <block>
```

```
    <prompt bargain="false">
```

Permite p.e. "de Granada a Loja"

Bienvenido a nuestro sistema automatico de informacion...

```
    </prompt>
```

```
  </block>
```

```
  <initial name="prompt_inicial">
```

```
    <prompt>Desde que ciudad a que ciudad deseas viajar?</prompt>
```

```
    <nomatch count="1">
```

Por ejemplo, di desde Granada a Cordoba

```
    </nomatch>
```

```
    <nomatch count="2">
```

Lo siento, sigo sin comprender lo que dices. Voy a solicitarte la informacion por partes

```
      <assign name="prompt_inicial" expr="true">
```

```
      <reprompt>
```

```
    </nomatch>
```

```
  </initial>
```

(*continúa en  
siguiente  
diapositiva ...*)

Necesario para que se escuche el siguiente prompt

**Ej. Iniciativa mixta**

El mensaje inicial no puede ser interrumpido por usuario

La variable asociada al campo del formulario **prompt\_inicial** se pone a **true** para que no vuelva a ser visitado por el FIA

...  
...  
...

Permite p.e. "Granada"

```
<field name="ciudadOrigen">  
  <grammar src="http://www.direcciones.ejemplo/ciudad.jsgf"/>  
  <prompt>Desde que ciudad deseas salir?</prompt>  
  ... etc. ...  
</field>  
... etc. ...  
</form>
```

# VoiceXML: Formularios de iniciativa mixta

- Enlace: `<link>`
  - Tiene una o más gramáticas asociadas
  - Se activa cuando la entrada del usuario es aceptada por alguna gramática
  - Ámbitos de un enlace
    - Si es hijo de `<vxml>` entonces gramáticas activas en todo el documento
    - Si es hijo de `<form>` entonces gramáticas activas en el formulario
    - Si está en documento raíz a nivel de documento entonces gramáticas activas en cualquier documento de la aplicación

# VoiceXML: Formularios de iniciativa mixta

- Enlace: **<link>**
  - Permite
    - realizar transiciones a un nuevo documento o diálogo (como **<goto>**)

```
<link next="http://www.voicexml.org/books/main.vxml">  
  <grammar type="application/x-jsgf"> libros | libros de VoiceXML </grammar>  
  <dtmf> 2 </dtmf>  
</link>
```

Este enlace se activa al pronunciar las frases “libros” o “libros de VoiceXML”, o al pulsar el botón del número “**2**” en el teléfono



# VoiceXML: Formularios de iniciativa mixta

- Enlace: **<link>**
  - Permite
    - generar un evento (como **<throw>**)

```
<link event="help">  
  <grammar type="application/x-jsgf">  
    no lo entiendo | ayuda | puedo tener ayuda | necesito ayuda | explicamelo  
  </grammar>  
</link>
```

Al pronunciar estas frases se genera el evento **“help”**

# Conceptos sobre VoiceXML: Variables

- Declaración

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

Tiene el valor  
especial **undefined**

- Asignación

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

# Conceptos sobre VoiceXML: Variables

- Liberar valor de variables

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

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

# Conceptos sobre VoiceXML: Gramáticas

- El elemento **<grammar>** se usa para especificar gramática que determina conjunto de frases que usuario puede pronunciar para realizar acción o proporcionar información
- La gramática puede proporcionar
  - Un solo valor mediante una cadena de caracteres (**gramática a nivel de campo**)
  - Un par atributo-valor (**gramática a nivel de formulario**)

# Conceptos sobre VoiceXML: Gramáticas

- Dos tipos de gramáticas
  - Interna

```
<link next="#exit">  
  <grammar>adios|terminar|finalizar</grammar>  
</link>
```

- Dos tipos de gramáticas
  - Externa

```
<form id="gestion_informacion">  
  <grammar src="viajar_de_a.jsgf"/>
```

Gramática **externa** a nivel de formulario

```
  <initial>  
    <prompt>Como puedo ayudarte?</prompt>  
  </initial>
```

```
  <!-- obtencion ciudad destino -->  
  <field name="ciudadDestino">  
    <prompt>A que ciudad quieres viajar?</prompt>  
    <grammar src="ciudades.jsgf"/>  
  </field>
```

Gramática **externa** a nivel de campo

```
  ...  
</form>
```

- Ejs. Gramáticas JSGF (Java Speech Grammar Format)

#JSGF V1.0;

**Gramática a nivel de campo**

grammar ciudades;

public <ciudades> = jaen | cordoba | sevilla | huelva | cadiz |  
malaga | granada | almeria;

#JSGF V1.0;

grammar viajar\_de\_a;

public <viajar\_de\_a> =  
    [<deseo>]

    [<destino> <ciudad> {this.ciudadDestino=\$ciudad} ]

    [<procedencia> <ciudad> {this.ciudadOrigen=\$ciudad} ];

Si en la frase aparece  
“<destino> <ciudad>”  
esa ciudad se asigna al  
campo **ciudadDestino**

<deseo> = quiero | me gustaria |  
[yo] queria | [yo] necesito | [yo] tengo que;

<destino> = [ir | viajar] a;

<ciudad> = jaen | cordoba | sevilla |  
huelva | cadiz | malaga | granada |  
almeria;

Si en la frase aparece  
“<procedencia>  
<ciudad>” esa ciudad  
se asigna al campo  
**ciudadOrigen**

**Gramática a nivel de formulario**

<procedencia> = de | desde | salir desde | saliendo desde ;



# Conceptos sobre VoiceXML: Gramáticas

- Ámbito de las gramáticas (scope)
  - **Gramática de campo**: sólo están activas cuando el FIA visita el campo. No tienen atributo scope
  - **Gramática de enlace**: tiene el ámbito correspondiente al elemento que contiene el enlace. No tienen atributo scope

# Conceptos sobre VoiceXML: Gramáticas

- Ámbito de las gramáticas (scope)
  - Gramática de formulario:
    - Por defecto, tiene como ámbito *dialog* (sólo está activa cuando usuario está en formulario)  
**scope="dialog"**
    - Si tiene ámbito *document* (está activa en cualquier diálogo del documento) **scope="document"**
    - Si **scope="document"** y el documento es el raíz de la aplicación, está activa cuando interacción está en cualquier diálogo de cualquier documento de la aplicación

# Conceptos sobre VoiceXML: Gramáticas

- Ámbito de las gramáticas (scope)
  - **Gramática de menú**: por defecto, tiene como ámbito **dialog**. Sólo está activa cuando usuario está en menú

# Conceptos sobre VoiceXML: Gramáticas

- 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*)

```
<field name="numero_telefono" type="phone">  
  <prompt>Cual es tu numero de telefono?</prompt>  
</field>
```

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

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

```
<prompt>Has dicho <value expr="numero_telefono"/> ?  
</prompt>
```

Nº teléfono se reproduce dígito a dígito (pe. **9 5 8 1 2 3 4 5 6**)

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

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

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

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

# Conceptos sobre VoiceXML: Eventos

- Gestión de eventos
  - Generados por plataforma (p.e. usuario no responde, solicita ayuda, etc.)
  - Generados por intérprete (por existencia errores en documento o al encontrar un elemento **<throw>**)

# Conceptos sobre VoiceXML: Eventos

- Gestión de eventos
  - Elementos relacionados con eventos: `<throw>`, `<catch>`, `<error>`, `<help>`, `<noinput>`, `<nomatch>`
  - `<throw>` se usa para generar un evento

```
<throw event="nomatch"/>  
<throw event="noinput"/>  
<throw event="help"/>
```

# Conceptos sobre VoiceXML: Eventos

- Gestión de eventos
  - `<catch>` se usa para responder a eventos
    - Contiene código ejecutable
    - Ejemplo en siguiente diapositiva ...



```
<form id="lanzamiento_misiles">
  <field name="id_usuario" type="digits">
    <prompt>Nombre de usuario?</prompt>
  </field>
  <field name="clave">
    <prompt>Cual es la clave?</prompt>
    <grammar>lechuga</grammar>
    <help>Es el nombre de un vegetal</help>
    <catch event="nomatch noinput" count="3">
      <prompt>Violacion de seguridad!</prompt>
      <submit next=http://www.ejemplo.com/intruso.vxml
        namelist="id_usuario"/>
    </catch>
  </field>
</block>
  <goto next="#obtener_ciudad"/>
</block>
</form>
```

La tercera vez  
que se produce  
alguno de los  
eventos se  
ejecuta el código  
de gestión  
correspondiente

# Conceptos sobre VoiceXML: Eventos

- Gestión de eventos
  - Notación abreviada de **<catch>**

**<error>**

Se ha producido un error. Por favor, llama de nuevo mas tarde

**<exit/>**

**</error>**

**<help>**No hay ayuda disponible**</help>**

**<noinput>**

No he escuchado nada. Por favor, intentalo de nuevo

**</noinput>**

**<nomatch>**

He oido algo, pero no se trata de una ciudad conocida

**</nomatch>**

forma abreviada de  
**<catch event="error">**  
(análogo en los demás casos)

# Conceptos sobre VoiceXML: Eventos

- Otros eventos
  - Eventos predefinidos (normales)
    - **telephone.disconnect.hangup** → usuario cuelga teléfono
    - **telephone.disconnect.transfer** → llamada transferida a otra línea, sin que exista retorno
  - Eventos predefinidos (de error)
    - **error.badfetch** → p.e. falta documento, URI mal escrita, error de comunicación en proceso de acceso a recurso, etc.
    - **error.semantic** → p.e. división por cero, referencia a variable no definida, etc.

# Conceptos sobre VoiceXML

- Contenido ejecutable
  - Bloque de lógica procedural que puede estar en:
    - Bloques (**<block>**) de un formulario
    - Acciones (**<filled>**) en formulario o en campos del formulario
    - Manejadores de eventos (p.e. **<catch>**, **<help>**, etc.)

# Conceptos sobre VoiceXML

- **Contenido ejecutable**
  - Elementos que pueden estar en un bloque ejecutable
    - `<var ...>`
    - `<assign ...>`
    - `<clear ...>`
    - `<if ...> ... <elseif ...> ... <else> ...`
    - `<prompt ...>`
    - `<reprompt ...>`
    - `<goto ...>`
    - `<submit ...>`
    - `<exit>`
    - `<return>`
    - `<disconnect>`
    - `<script> ...`

# Conceptos sobre VoiceXML

- Lógica condicional
  - **<if> ... <elseif> ... <else>** se usa para crear secciones de lógica condicional en el documento
  - **<elseif>** y **<else>** son opcionales

```
<if cond="total > 1000">  
  <prompt>Se ha gastado demasiado dinero</prompt>  
</if>
```

```
<if cond="cantidad < 29.95">  
  <assign name="x" expr="cantidad"/>  
<else/>  
  <assign name="x" expr="29.95"/>  
</if>
```

```
<if cond="sabor == 'vainilla' ">  
  <assign name="codigo_sabor" expr=" 'v' "/>  
<elseif cond="sabor == 'chocolate' "/>  
  <assign name="codigo_sabor" expr=" 'c' "/>  
<elseif cond="sabor == 'fresa'"/>  
  <assign name="codigo_sabor" expr=" 'f' "/>  
<else/>  
  <assign name="codigo_sabor" expr=" '?' "/>  
</if>
```

# Conceptos sobre VoiceXML

- Generación de prompts

```
<nomatch count="1">
```

Para abrir la puerta di claramente tu clave

```
</nomatch>
```

```
<nomatch count="2">
```

<prompt>Este es tu **<emp>** ultimo **</emp>** intento</prompt>

```
</nomatch>
```

```
<nomatch count="3">
```

Entrada denegada

```
<exit/>
```

```
</nomatch>
```

**Tapered prompts:** el mensaje cambia en función del valor del contador

**<prompt> ... </prompt>** necesarios si el mensaje contiene etiquetas, p.e. **<emp>**



# Conceptos sobre VoiceXML

- Generación de prompts

Pronunciar texto con un estilo determinado (no igualmente soportado por todas las plataformas)

```
<help>
  <prompt>Estas llamando al numero
    <value expr="num_telefono" class="phone"/>
  </prompt>
  <prompt>Estas llamando al numero
    <sayas class="phone">312-555-1212</sayas> </prompt>
</help>
```

Texto alternativo a generar mediante TTS en caso de que no esté disponible **bienvenida.wav**

```
<block>
  <prompt> <audio src="bienvenida.wav"><emp> Bienvenido </emp>
  a este portal de voz</audio> </prompt>
</block>
```

# Conceptos sobre VoiceXML

- Generación de prompts

Generación del mensaje no interrumpida si usuario comienza a hablar antes de su finalización

```
<prompt bargein="false"><audio src="aviso_legal.wav"/></prompt>
```

# Conceptos sobre VoiceXML

- Generación de prompts

Obtención número aleatorio

```
<form id="otro_chiste">
  <var name="r" expr="Math.random()"/>
  <field name="otro" type="boolean">
    <prompt cond="r < .50">
      Quieres escuchar otro chiste?
    </prompt>
    <prompt cond="r >= .50">
      Si quieres escuchar otro chiste, di si. Para salir, di no
    </prompt>
    <filled>
      <if cond="otro">
        <goto next="#seleccionar_chiste"/>
      </if>
    </filled>
  </field>
</form>
```

**Prompt condicional:**  
se ejecuta si  $r < .50$

**Prompt condicional:**  
se ejecuta si  $r \geq .50$

# Conceptos sobre VoiceXML

- Generación de prompts

El usuario tiene 120 s  
para responder

```
<prompt count="1">Elige un color para tu nuevo Modelo T</prompt>  
<prompt count="2" timeout="120s">
```

Por favor, elige el color de tu nuevo Modelo T 19 24. Algunos posibles colores son los siguientes: negro, negro o negro. Por favor, elige con tranquilidad

```
</prompt>
```

# Conceptos sobre VoiceXML

- **Reprompt**
  - El algoritmo FIA generalmente no reproduce los prompts en la reiteración **tras la ejecución de un elemento <catch>**
  - **<reprompt>** indica al FIA que reproduzca el prompt

```
<field name="helado_para_postre" type="boolean">  
  <prompt>Quieres helado de postre?</prompt>  
  <prompt count="2">  
    Si quieres helado, di si. Si no quieres, di no  
  </prompt>  
  <noinput>  
    No he oído nada  
    <reprompt/>  
  </noinput>  
</field>
```

# Conceptos sobre VoiceXML

- Reprompt

Nuevos Paradigmas de Interacción

S: Quieres helado de postre?

U: (*silencio*)

S: No he oído nada. Si quieres helado, di sí. Si no quieres, di no

U: (*silencio*)

S: No he oído nada. Si quieres helado, di sí. Si no quieres, di no

U: No

Usando reprompt

S: Quieres helado de postre?

U: (*silencio*)

S: No he oído nada

U: (*silencio*)

S: No he oído nada

U: No

Sin usar reprompt

# Conceptos sobre VoiceXML

- **<goto>** realizar transiciones a:
  - Otro ítem del mismo formulario

```
<goto nextitem="confirmacion_sn"/>
```

- Otro formulario del mismo documento

```
<goto next="#otro_dialogo"/>  
<goto expr="'#' + 'otro_dialogo'"/>
```

- Otro documento

```
<goto next="http://ejemplo.vuelo/reserva_asiento"/>  
<goto next="./almuerzo_especial/#vegetariano"/>
```

# Conceptos sobre VoiceXML

- **SUBMIT**
  - **<submit>** permite enviar lista de variables a un servidor de documentos mediante peticiones HTTP Get o Post

```
<submit next="www.miservidor.ugr.es" method="post"  
namelist="nombre rango numero_serie"  
fetchtimeout="100s" fetchaudio="audio/brahms2.wav"/>
```

Tiempo espera  
respuesta del servidor

Fichero de audio a usar  
mientras llega respuesta



# Conceptos sobre VoiceXML

- **EXIT**
  - **<exit/>** devuelve el control al intérprete, el cual decide qué hacer a continuación, p.e.:
    - Ejecutar menú de nivel superior
    - Finalizar la llamada
    - Transferir llamada a un operador
    - Etc.

# Conceptos sobre VoiceXML

- **DISCONNECT**

- **<disconnect/>** fuerza al intérprete a desconectar la llamada del usuario, generando el evento **telephone.disconnected.hangup**

- **SCRIPT**

- **<script>** se usa para especificar código del lado del servidor que realiza una determinada función (análogo a **<script>** de HTML)
- Puede estar dentro de elemento **<vxml>** o en código ejecutable

# Conceptos sobre VoiceXML

- **SCRIPT**

```
...  
<block>  
  <script>  
    var f = new Date();  
    horas = f.getHours();  
    minutos = f.getMinutes();  
    segundos = f.getSeconds();  
  </script>  
</block>  
...  
<prompt> Hora actual, <value expr="horas"/> horas, <value  
expr="minutos"/> minutos y <value expr="segundos"/> segundos  
</prompt>
```

# Conceptos sobre VoiceXML

- Ejecución en múltiples documentos

```
<vxml version="2.1" xml:lang="es-ES">  
  <var name="despedida" expr="adios"/>  
  <link next="operador_xfer.vxml"> <grammar> operador  
</grammar> </link>  
</vxml>
```

Documento raíz:  
**app-root.vxml**

El documento hoja especifica  
URI de documento raíz

```
<vxml version="2.1" xml:lang="es-ES" application="app-root.vxml">
  <form id="decir_adios">
    <field name="respuesta" type="boolean">
      <prompt>Nos decimos <value expr="application.despedida"/>?
    </prompt>
    <filled>
      <if cond="respuesta">
        <exit/>
      </if>
      <clear namelist="respuesta"/>
    </filled>
  </field>
</form>
</vxml>
```

Documento hoja:  
**main.vxml**

# Conceptos sobre VoiceXML: Recursos

- **Búsqueda de recursos (resource fetching)**
  - Acceso a recursos en una URI gobernado por varios atributos
  - **caching**
    - **“safe”**: acceder a versión más reciente
    - **“fast”**: usar versión en caché del recurso

# Conceptos sobre VoiceXML: Recursos

- **Búsqueda de recursos (resource fetching)**
  - **fetchhint** → especifica cuándo el entorno del intérprete debe obtener un recurso del servidor
    - **“prefetch”**: descargar recurso cuando se carga el documento
    - **“safe”**: descargar recurso cuando es realmente necesario
    - **“stream”**: usado para recursos grandes. Comenzar a procesar recurso conforme va llegando, sin esperar a su llegada completa

# Conceptos sobre VoiceXML: Recursos

- Búsqueda de recursos (resource fetching)

```
<property name="caching" value="fast"/>
<form id="test">
  <block>
    <!-- El mensaje de bienvenida raramente cambia, así que caching
    fast va bien -->
    <audio src="http://www.weather4U.example/vxml/welcome.wav"/>
    <!-- Otros mensajes cambian frecuentemente, así que se usa
    caching safe -->
    <audio caching="safe"
      src="http://www.adiciones_online.ejemplo/prevision/ad17"/>
  </block>
</form>
```

Los recursos de este documento  
usarán por defecto  
**caching="fast"**



# Conceptos sobre VoiceXML: Otros

- Grabación de mensajes
  - `<record>` se usa para grabar mensajes del usuario
  - Estos mensajes pueden ser reproducidos o enviados a algún servidor

```
<vxml version="2.1" xml:lang="es">
  <form>
```

Se emite un pitido de  
comienzo de grabación

Al pulsar botón del  
teléfono se detiene  
grabación mensaje

```
    <record name="saludo" beep="true" maxtime="10s"
finalsilence="4000ms" dtmfterm="true" type="audio/wav">
      <prompt>Graba tu mensaje tras escuchar el tono</prompt>
      <noinput>No he oido nada, intentalo de nuevo</noinput>
    </record>
    <field name="confirmacion" type="boolean">
      <prompt>Tu mensaje es <value expr="saludo"/></prompt>
      <prompt>Para mantenerlo, di si. Para descartarlo, di no</prompt>
      <filled>
        <if cond="confirmacion">
          <submit next="guardar_saludo.pl" method="post" namelist="saludo"/>
        </if>
        <clear/>
      </filled>
    </field>
  </form>
</vxml>
```

El mensaje grabado se  
envía a un servidor

# Conceptos sobre VoiceXML: Otros

- Especificación de propiedades de la plataforma
  - **<property>** se usa para especificar valores que afectan a funcionamiento de la plataforma (p.e. proceso de RAH, expiración de temporizadores, política de caché, etc.)
    - Definibles a distintos niveles: documento, diálogo, ítem de formulario
    - Propiedades en documento raíz representan valores por defecto para propiedades en documentos de la aplicación
    - Propiedad definida a nivel inferior tiene prioridad sobre definición en nivel superior

# Conceptos sobre VoiceXML

- Especificación de propiedades de la plataforma
  - Ejemplo

```
<form id="no_bargein_form">  
  <property name="bargein" value="false"/>  
  <block>  
    <prompt>Este prompt introductorio no permite barge-in</prompt>  
    <prompt>Y este tampoco</prompt>  
    <prompt bargein="true">Pero este <emp>si</emp> permite  
    barge-in</prompt>  
  </block>  
  ...  
</form>
```

Deshabilita **barge-in** para todos los prompts del diálogo

Tiene prioridad sobre el valor por defecto

La palabra “**sí**” se pronuncia con énfasis

# Referencias

- VoiceXML 1.0
  - <http://www.w3.org/TR/voicexml>
- VoiceXML 2.0
  - <http://www.w3.org/TR/voicexml20/>
- VoiceXML 2.1
  - <https://www.w3.org/TR/voicexml21/>
- VoiceXML 3.0
  - <http://www.w3.org/TR/voicexml30/>