# Realización de sistemas de diálogo en una plataforma compatible con VoiceXML: Proyecto GEMINI

**R. Córdoba, F. Fernández, V. Sama, L.F. D'Haro, R. San-Segundo, J.M. Montero, J. Macías-Guarasa, J. Ferreiros, J.M. Pardo**
Grupo de Tecnología del Habla. Departamento de Ingeniería Electrónica. UPM.
E.T.S.I. Telecomunicación. Ciudad Universitaria s/n, 28040 Madrid, Spain.
{cordoba, efhes, vsama, lfdharo, lapiz, juancho, macias, jfl, pardo}@die.upm.es

**Resumen:** En este artículo hacemos un estudio de lo que ha supuesto la utilización del estándar VoiceXML en la plataforma de sistemas de diálogo de la que ya disponíamos en el grupo. Como intérprete de VoiceXML hemos elegido a OpenVXI, solución de código abierto portable que nos permite hacer las modificaciones necesarias para adaptarnos a las características de nuestros módulos de reconocimiento y síntesis, por lo que hacemos un énfasis especial en los cambios que ha sido necesario realizar en dicho intérprete. Así mismo, hacemos una revisión de todos los módulos de nuestra plataforma y sus características, destacando la adopción de estándares en los mismos, como SSML para el conversor texto-voz y JSGF para la especificación de las gramáticas del reconocedor. Para terminar, hacemos una serie de comentarios acerca de las limitaciones que hemos detectado en VoiceXML.
**Palabras clave:** Sistemas automáticos de diálogo, reconocimiento de habla, síntesis de voz, VoiceXML, OpenVXI.

**Abstract:** In this paper we study the approach followed to use the standard VoiceXML in a dialog system platform already available in our group. As VoiceXML interpreter we have chosen OpenVXI, an open source portable solution that allows us making the modifications needed to adapt to the characteristics of our recognition and synthesis modules, so we will emphasize the changes that we have had to make in such interpreter. Besides, we review all modules in our platform and their capabilities, highlighting the use of standards in them, as SSML for the text-to-speech system and JSGF for the specification of grammars for recognition. Finally, we discuss several ideas regarding the limitations detected in VoiceXML.
**Keywords:** automatic dialog systems, speech recognition, voice synthesis, VoiceXML, OpenVXI.

## *1 Introducción*

El proyecto Gemini (Generic Environment for Multilingual Interactive Natural Interfaces, IST-2001-32343), objeto de esta comunicación (Gemini, 2003), es un proyecto de dos años (2002-2004) que acaba de terminar, financiado por la Unión Europea en el que interviene nuestro grupo representando a la UPM. El consorcio está formado por los socios siguientes: Knowledge S.A. (Grecia) como coordinador, Universidad de Patras – WCL (Grecia), TEMIC SDS (Alemania), Universidad Politécnica de Madrid – GTH (España), FAW (Alemania) y Egnatia Bank (Grecia).

El objetivo fundamental del proyecto ha sido el desarrollo de un sistema de generación de aplicaciones de diálogo de forma semiautomática partiendo de una descripción de la base de datos. El segundo objetivo fundamental ha sido la preparación de una plataforma en tiempo real en la que ejecutar los guiones obtenidos en el sistema de generación. En dicho sistema, el resultado final para una aplicación de habla es un guión del diálogo escrito en lenguaje VoiceXML 2.0.

Para poder ejecutarlo, necesitamos de un intérprete. Hemos elegido como intérprete OpenVXI 2.0.1, de SpeechWorks (SpeechWorks, 2004), por ser una solución de código abierto y que por su portabilidad, no requiere de ningún motor de reconocimiento o síntesis de voz en particular, ni es específico de una plataforma telefónica concreta. Como inconvenientes cabe destacar la escasez de una

documentación válida, lo que dificulta la introducción de los motores en el intérprete y la ausencia de una implementación de referencia con funcionalidad real.

La utilización del lenguaje VoiceXML confiere al sistema un alto nivel de estandarización gracias a la creciente adaptación de VoiceXML como lenguaje estándar de definición de aplicaciones vocales.

Para poder demostrar la eficacia de la plataforma, ha sido probada con una aplicación bancaria, que ofrece información general de productos del banco: créditos personales, créditos para compra de coche, hipotecas; depósitos y sus tipos de interés; tarjetas de crédito / débito, etc.; autenticación del usuario: el usuario debe introducir su número de cuenta y un PIN; consultas y transacciones para las cuentas del cliente: consulta de saldos y movimientos para las cuentas y tarjetas de crédito, realización de transferencias entre cuentas, etc.

## *2 Antecedentes*

Este proyecto es la continuación del proyecto IDAS (LE4-8315) que se han presentado en distintos congresos (Córdoba 2001, Córdoba 2002, Lehtinen 2000.)

En dicho proyecto, se desarrolló un demostrador capaz de dar un servicio de páginas blancas por teléfono en el que se proporcionaba a los usuarios números de teléfono (o fax) de particulares y de empresas. Fue evaluado con usuarios reales y sus tasas de éxito fueron muy positivas.

Como sistema de partida se ha empleado "Servivox", desarrollado íntegramente en nuestro grupo y utilizada en muchos otros proyectos como en (San-Segundo 2001), donde se aplicó a un sistema de información y reserva de trenes. Para su integración dentro de la plataforma OpenVXI fue precisa una importante labor previa de documentación y análisis que permitió hacer frente a los distintos módulos del sistema original de forma independiente. Fruto de este trabajo se han desarrollado las correspondientes interfaces de acceso a dichos módulos.

## *3 El intérprete OpenVXI*

El intérprete OpenVXI es un software de código abierto que ha sido desarrollado por SpeechWorks (SpeechWorks, 2004) y que se utiliza para ejecutar guiones en VoiceXML. Su gran ventaja es que proporciona gran parte de la funcionalidad necesaria para ejecutar aplicaciones de diálogo, como una interfaz XML que procesa el guión VoiceXML, una API JavaScript, una API de funciones de WWW y una API de registro. Para las funciones relacionadas con entrada/salida (reconocimiento, síntesis y control telefónico) proporciona unas interfaces incompletas que podemos modificar para adaptarnos a las necesidades de cada plataforma. Vamos a describir ahora el trabajo realizado en cada uno de estos módulos.

### 3.1 Módulo de reconocimiento de habla

Se ha hecho un gran esfuerzo en adaptar nuestro software de reconocimiento a OpenVXI. El sistema admite la carga dinámica y la utilización de distintos reconocedores, incluso de forma simultanea, durante la ejecución de una aplicación VoiceXML. Se han implementado mecanismos que basados en VoiceXML permiten identificar el reconocedor a utilizar en cada momento de la aplicación, si bien se ha detectado una falta de estandarización al respecto. La incorporación de esta funcionalidad nos facilita la activación y utilización de varias gramáticas en paralelo durante un determinado proceso de reconocimiento. En este sentido el sistema posibilita el tener modelos y gramáticas de reconocimiento diferentes en función de la pregunta al usuario, pudiendo adaptarlos a distintas situaciones: habla continua, habla aislada, dígitos conectados, fechas, etc.

Para conseguirlo, hemos definido una propiedad específica para nuestra plataforma que, siguiendo las directivas de la especificación VoiceXML 2.0, se llamará "es.upm.die.gth.recognizer" y cuyo valor determina el reconocedor a utilizar. De este modo si en un determinado documento VoiceXML especificamos:

<property name="es.upm.die.gth.recognizer" value="hablaAislada" />

La implementación de la plataforma cargará y posteriormente utilizará dicho reconocedor (el de habla aislada en este caso).

La implementación que hemos realizado de la interfaz de reconocimiento de OpenVXI tiene las siguientes características:

1. Gestión y registro de errores y de eventos (i.e. fin de temporización).

Se han implementado los mecanismos de contingencia ante eventos como la solicitud de

ayuda por parte del usuario, que puede estar relacionada bien con el prompt facilitado por el sistema o bien con el estado del diálogo, la solicitud de salida del sistema por parte del usuario en un momento determinado, la solicitud de repetición de la pregunta o del prompt formulado por el sistema, o la respuesta del sistema ante la no detección de habla durante un proceso de reconocimiento.

2. Distintos canales de entrada

Hemos diseñado el módulo para poder realizar el reconocimiento bien a partir de muestras capturadas directamente desde un micrófono o desde el teléfono, o bien a través de muestras de voz pregrabadas y almacenadas en un fichero. El motor permite su configuración de forma que se guarden en un fichero las muestras de voz sobre las que se ha aplicado el proceso de reconocimiento, con el objeto de que estén disponibles posteriormente para tareas de reproducción o depuración del sistema.

La funcionalidad que le falta a OpenVXI y que, por tanto, ha sido necesario implementar es:

1. Un identificador del canal

Se necesita un identificador para el canal que estamos gestionando y poder asignarle de forma exclusiva un determinado dispositivo de audio de entrada. Se ha asignado un número para cada uno de los hilos que se crean para atender cada canal y se han modificado las cabeceras de funciones necesarias para conseguir disponer de este valor.

2. Carga previa del reconocedor

Para utilizar un determinado reconocedor debemos haberlo cargado previamente en el sistema, es decir, haber realizado una carga de los modelos que lo definen para no ralentizar el proceso de reconocimiento en tiempo real. Ni la especificación VoiceXML ni OpenVXI conceden importancia alguna a la necesidad de cargar previamente el reconocedor, con lo que no existe un momento o sitio específico donde realizar la carga de los modelos. Sólo hay dos sitios posibles donde hacer dicha carga: al cargar la gramática y en la llamada al reconocedor. Como la carga de la gramática es previa al reconocimiento en sí, ha sido el sitio elegido para hacerlo.

3. Asociación gramática-reconocedor

Se necesita una asociación directa entre la gramática o gramáticas activas y los modelos que queremos emplear en el reconocimiento. En OpenVXI se permite tener múltiples gramáticas activas, a la vez que diferentes modelos de reconocimiento, pero tanto VoiceXML como OpenVXI dejan este tema abierto. Para advertir al intérprete de los modelos y gramáticas concretos que queremos emplear al lanzar un proceso de reconocimiento hemos decidido aprovechar la implementación del método *LoadGrammarFromUri* de la interfaz de reconocimiento VXIrec (vacía en la distribución de OpenVXI). Este método se utiliza para cargar una determinada gramática en el sistema cuando el atributo “src” de la etiqueta <grammar> es no nulo. Este atributo indica la dirección URI del documento donde buscar la definición de la gramática a cargar. Nuestra implementación incluye la búsqueda y carga del documento que contiene la gramática. Además, permite establecer una relación entre una gramática y unos modelos específicos mediante la declaración del atributo “type” asociado a la etiqueta <grammar> empleada para declarar la gramática en cuestión.

<grammar src=“datos/bigram.dat” type=“Aislada”/>

Se indica al intérprete que los modelos que debe cargar en el reconocedor, teniendo activa la gramática referenciada en “src”, son aquellos a los que hemos asociado el nombre de “Aislada”. Así mismo, se puede establecer cualquier relación entre una gramática y unos modelos asociando un “type” distintivo a cada uno de los modelos de que dispongamos.

## 3.2 Módulo de generación de voz

Este motor ofrece todos los servicios relacionados con la generación de voz contemplados por el estándar VoiceXML, incluyendo la reproducción de ficheros de audio y la conversión texto-voz que permite ofrecer al usuario mensajes de contenido variable sin necesidad de grabarlos previamente.

El motor de generación de voz está constituido por los siguientes módulos:

- Módulo de síntesis de voz. Creado a partir de las funciones ofrecidas por el conversor texto-voz “Boris”, conversor desarrollado por el grupo GTH, ampliamente probado y distribuido comercialmente (Pardo, 1995.)
- Módulo de reproducción de muestras de voz pregrabadas. Permite reproducir muestras de audio pregrabadas especificando la ruta y el nombre del fichero que las contiene.

La implementación del mismo también cubre los siguientes aspectos:

- Permite al usuario disponer de un control prosódico básico sobre la voz sintetizada según las diversas posibilidades que ofrece el lenguaje SSML. Para ello, se han desarrollado una serie de funciones que permitan interpretar el lenguaje de etiquetas SMML.
- Permite utilizar las funcionalidades de conversión texto a voz cuando la reproducción desde fichero falle.
- El procesamiento relativo a la reproducción desde fichero y la síntesis de voz se hace de forma asíncrona, no bloqueante, con el objetivo de permitir la interrupción del diálogo por parte del hablante (barge-in) según la especificación VoiceXML 2.0.
- Gestión y registro de los diferentes errores y eventos que puedan producirse.

La funcionalidad que le falta a OpenVXI y que, por tanto, ha sido necesario implementar es:

1. Un identificador del canal

Se necesita un identificador igual que para el reconocedor (ver sección 3.1.)

2. Cola FIFO

Se ha tenido que implementar una estructura tipo cola FIFO para almacenar los mensajes de voz que luego serán reproducidos en el procedimiento *Play()*.

3. Procedimiento *Play()*

La documentación oficial del procedimiento es incompleta e incorrecta para la reproducción no bloqueante. Lo que es necesario realizar es: Comprobar que existen mensajes por reproducir, si no salir; Comprobar que no existe ningún proceso de reproducción en curso (en caso afirmativo, esperar a que acabe); Bucle de reproducción bloqueante de todos los mensajes en cola excepto el último que se reproducirá de forma no bloqueante.

El estándar SSML (Speech Synthesis Markup Language) (Burnett, 2002) tiene como objetivo el proporcionar una manera estándar de controlar los elementos que entran en juego en la síntesis de voz como pueden ser la frecuencia fundamental, la pronunciación, y el volumen a través de las diferentes plataformas. Lo hace mediante una serie de etiquetas que se introducen con el texto a sintetizar. Hemos implementado la interpretación de las siguientes etiquetas en nuestro conversor:

1. “emphasis” (para dar énfasis a ciertos fragmentos), teniendo el atributo “level” los siguientes posibles valores: “strong”, “moderate”, “none” y “reduced”.
2. “break” (pausa de un tiempo definido), con el atributo “time”, que es un entero seguido de “ms”. Por ejemplo, “300ms” para hacer una pausa de 300 ms.
3. “prosody”, con los atributos “pitch” (frecuencia fundamental), “rate” (velocidad) y “volume” (volumen). Para los 3 atributos, se pueden especificar los valores de tres maneras: (a) mediante valores discretos dados por cadenas, por ejemplo “x-high” || “high”|| “medium”||..., (b) con un número flotante y la unidad correspondiente, como en “100Hz”, o (c) con un valor relativo, en forma de porcentaje positivo o negativo, i.e. “+10%”. Nosotros hemos optado por esta última opción.

Ejemplo:

¿Ha dicho que quiere transferir <break time=“100ms”/> <emphasis level=“strong”> 100 euros </emphasis> desde su cuenta <prosody volume=“+20%”> 3466 </prosody> <break time=“200ms”/> a la cuenta con el número <prosody pitch=“+40%”> 564 </prosody>?

## 3.3 Módulo de control telefónico

La norma VoiceXML sólo estandariza las acciones de desconexión y transferencia de llamadas, con lo cual el proceso de integración de los servicios telefónicos ha resultado relativamente sencillo. De nuevo para ello hemos empleado tecnología propietaria desarrollada íntegramente en el GTH. Se ha integrado con éxito la Tarjeta interfaz de línea telefónica (IFTEL). Esta tarjeta conecta el sistema a la línea telefónica y se encarga de adaptar las señales entre la línea telefónica y la tarjeta de sonido del PC. También permite conectar dispositivos auxiliares al sistema, como auriculares, equipos de grabación de voz, etc.

La principal función que realiza la interfaz de línea es realizar la conexión del sistema con la línea telefónica, realizando la conversión de 2 a 4 hilos y adaptando los niveles de señal a ambos lados de la interfaz.

Además, y debido a las necesidades del sistema, realiza otras muchas funciones, entre las que se encuentran las siguientes:

- Adaptación de señales para conectar dispositivos externos auxiliares de entrada y de salida, como altavoces, micrófonos y auriculares.
- Control de la conexión/desconexión con la línea telefónica.
- Control del dispositivo externo para la grabación de la conversación sistema/usuario.
- Detección de llamadas.
- Detección de tonos multifrecuencia (DTMF).
- Detección de paso a falta (el usuario cuelga).

Para todas estas funciones de la tarjeta se han implementado los correspondientes mecanismos de control para el intérprete OpenVXI siempre ciñéndonos a las especificaciones de la norma VoiceXML 2.0.

Por otra parte, se ha introducido una funcionalidad adicional de gran utilidad para el desarrollo y prueba de las diferentes aplicaciones generadas: la simulación de la línea telefónica mediante un micrófono y un altavoz. De este modo, es posible la comunicación oral con el sistema a través de estos dispositivos sin necesidad de hacerlo forzosamente a través de la línea telefónica, bastando una simple modificación de la configuración del sistema.

## *4 Los módulos de la plataforma*

Vamos a comentar ahora brevemente los módulos de los que disponemos en el Grupo para la creación de plataformas de diálogo y que se han utilizado en el sistema en tiempo real preparado para el proyecto Gemini y la aplicación bancaria que se ha desarrollado en él.

### 4.1 Módulo de reconocimiento de habla

En la actualidad, nuestro sistema utiliza modelos ocultos de Markov (HMM) continuos entrenados con la base de datos SpeechDat, de 4.000 locutores, que dispone de unas 46 horas de grabación de habla continua. El sistema utiliza modelos contextuales agrupados mediante un algoritmo de árboles de decisión. El sistema tiene 1807 estados diferentes, con 6 gaussianas por estado. La tasa de error, para una tarea de habla continua por teléfono de 3.065 palabras, es de únicamente un 4.2%.

Además, en nuestro sistema utilizamos un reconocedor de habla aislada de características similares al anterior para los casos en que se espere que el usuario responda con una palabra aislada. Lógicamente, la tasa de error es menor cuando se tienen que detectar palabras aisladas. También utilizamos en nuestro sistema reconocedores adaptados al reconocimiento de fechas, reconocimiento de dígitos y deletreo (se pasa a deletreo cuando hay errores en el reconocimiento.)

Así mismo, disponemos de un módulo para el cálculo de medidas de confianza. Dado que en un sistema de diálogo existen multitud de fuentes de error (errores de reconocimiento, respuestas inesperadas, etc.) resulta de gran utilidad disponer de herramientas para detectar de forma fiable qué tipo de información ha sido pronunciada. En función del valor de confianza del resultado, cambia el tipo de confirmación que hace el sistema. Si es baja, la confirmación será explícita; si es alta, será implícita o incluso sin confirmación. Hemos incluido un módulo de obtención de medidas de confianza que utiliza parámetros a nivel de palabra y de frase con una alta fiabilidad.

### 4.2 Módulo de comprension

El módulo de comprensión se encarga de extraer los conceptos que posteriormente tomará el sistema OpenVXI para rellenar la estructura en la que se depositan los resultados del reconocimiento.

Hemos adaptado este módulo a la aplicación bancaria desarrollada en el proyecto. La comprensión está basada en reglas dependientes de contexto, utilizándose un único diccionario para todos los *prompts* del sistema (16 en total contando el *prompt* de confirmación “Sí-No”). Este diccionario está etiquetado semánticamente en función de la información que se desea extraer. A las palabras que no aportan información de utilidad se les asigna la categoría “basura”, al igual que a las posibles palabras fuera de vocabulario con las que se pudiera encontrar el sistema, a pesar de ello pueden ser utilizadas en alguna de las reglas si se estima necesario.

A la hora de etiquetar el diccionario se ha tenido en cuenta la posibilidad de que una misma palabra puede aparecer en distintos *prompts.* Se ha llegado a una posible solución al respecto estableciendo que una palabra puede tener múltiples categorías, siendo una de ellas la de “basura” para que en caso de necesitar utilizar esa palabra pueda ser transformada por las reglas de comprensión antes de la

eliminación de los elementos basura. Así se podrían utilizar las palabras que se necesitasen y en caso de no ser necesarias se perderían tras el mencionado proceso de eliminación.

Creemos que esto no ocasiona problemas ya que las reglas de comprensión cambian en cada *prompt*, son reglas especificas para cada uno de ellos y solamente se llama a uno de los *prompts*, ya que el sistema sabe en qué momento de la conversación se encuentra, por lo que activa sólo aquel *prompt* que necesita.

Respecto a la secuencia que sigue este módulo para procesar una intervención del usuario, podríamos resumirla en los siguientes pasos: 1) preprocesado de la cadena de entrada procedente del reconocedor (incluye asignación de categorías, procesado de dígitos, eliminación de posibles interjecciones, etc.), este paso es común para todas las reglas, 2) reglas antes de eliminar las palabras que tiene la categoría "basura", 3) eliminación de palabras con categoría "basura", 4) reglas situadas tras la eliminación de "basuras", 5) escritura de los conceptos resultantes del proceso de comprensión.

Un posible ejemplo sería:

1) Preprocesado

2) reescribe3 ("ID_change", "basura", "ID_moneda", "cambio_moneda");

3) Eliminación de "basura"

4) reescribe2 ("ID_quiero", "cambio_moneda", "SLOT_cambio_moneda");

5) Obtención de los resultados

Como se puede observar se utilizan reglas de reescritura en las que se indican las categorías de origen al inicio de la regla y el elemento final sería la categoría de destino del resultado de esa regla. Así, en la regla del paso dos se van a utilizar tres palabras que se encuentran en el orden indicado en la regla (las tres primeras) y van a dar como resultado un elemento con la categoría "cambio_moneda". Esta regla recogería principalmente expresiones del tipo "cambio de moneda". Posteriormente, y tras la eliminación de las palabras con categoría "basura", vemos como en el paso 4 se utilizan dos elementos que darán como resultado el SLOT_cambio_moneda que finalmente será el resultado de la comprensión en el paso cinco.

### 4.3 Módulo de modelado de lenguaje

Disponemos de los habituales modelos de lenguaje estocásticos. Además, para este proyecto hemos desarrollado gramáticas de tipo JSGF (Java Speech Grammar Format).

Este tipo de gramáticas son independientes de plataforma y están basadas en el concepto de gramáticas de reglas (*rule grammar*) adoptando determinadas convenciones de Java a la forma de las gramáticas tradicionales. Se han utilizado para representar cómo suelen responder los usuarios a las preguntas del sistema. Al igual que en el módulo de comprensión, tenemos una gramática *JSGF* para cada *prompt* del sistema, además de una especifica para números. Al tratarse de un sistema multilingüe existen diferentes gramáticas para los idiomas disponibles en la aplicación.

Un posible ejemplo de gramática JSGF para un prompt del castellano podría ser:

```
#JSGF V1.0 ISO8859-1;
grammar GeneralInformationCathegory;
public <infocathegory> =
    <Deposit_Products> {this.$value="DepositProducts"}|
    <Cards>            {this.$value = "Cards"} |
    <EBanking>         {this.$value = "EBanking"} |
    <Exchange_Rates>   {this.$value="ExchangeRates"};

<Deposit_Products>= [infórmeme (sobre | de)] depósitos
            [<Polite>];
<Cards>= [<I_want>] tarjetas ;
<EBanking>= [(a cerca de)] banca [electrónica];
<Exchange_Rates>= [<I_want>] cambio [<Monedas>] ;
<Monedas> = de (moneda | divisas)
<I_want> = [(quiero | deseo)] información (de | sobre);
<Polite> = (por favor);
```

Está gramática corresponde al prompt en el que el usuario debe elegir entre diversos servicios bancarios (depósitos, tarjetas de crédito, cambio de moneda o información sobre banca electrónica) y para su escritura nos hemos basado en una captura de textos realizada durante el desarrollo de este sistema. En ella podemos ver cómo los elementos opcionales aparecen entre corchetes y los paréntesis se han utilizado para agrupar o desambiguar expresiones. A su vez, entre llaves encontramos etiquetas de información específica de la aplicación y las barras verticales indican una alternativa, siendo posible que aparezca un elemento u otro. Así, sí tomamos por ejemplo la regla llamada <Exchange_Rates> vemos que el elemento <I_want> es opcional y puede o no aparecer, dentro de este elemento hay diferentes alternativas ("quiero" o "deseo", etc.), irá seguido de la palabra "cambio" y posteriormente podrá aparecer el elemento <Monedas>.

### 4.4 Módulo de modelado de usuario

Consiste en introducir alternativas en el diálogo básico para adaptarse a la experiencia del usuario con el sistema. Se definen una serie de niveles de experiencia, y en función de ellos, el sistema ofrecerá más o menos información.

Por ejemplo, si el usuario es novato, se le proporcionará una ayuda más detallada, se le harán preguntas más concretas, se intentará confirmar lo que dice, etc.

Para decidir a qué nivel debe pertenecer un usuario, dado que en una llamada telefónica no se le suele identificar, hay que tener en cuenta una serie de factores como: rapidez en las respuestas, tasas de acierto del reconocimiento, solicitudes de ayuda, etc. (San-Segundo, 2001).

### 4.5 Módulo de reconocimiento de idioma

Uno de los objetivos del sistema es ser multilingüe. Para ello, debemos detectar el idioma del usuario actual utilizando un pequeño fragmento de lo que dice inicialmente, tras lo cual conmutamos al sistema de reconocimiento del idioma detectado.

Se pueden utilizar distintas técnicas. En el grupo hemos utilizado una llamada PPRLM, que se basa en modelar información de la secuencia de fonemas que se obtiene para cada uno de los idiomas utilizando un reconocedor de fonemas muy simple. Se pueden consultar los resultados en la discriminación entre inglés y castellano en (Córdoba, 2003.)

## *5 Limitaciones de VoiceXML*

La naturaleza del trabajo desarrollado nos ha obligado a acercarnos a la recomendación VoiceXML desde el punto de vista del desarrollador. Quizá por ello hemos visto el lado más "feo" de la especificación VoiceXML, lado que queda oculto al simple programador de aplicaciones que no tiene que enfrentarse a los problemas que subyacen.

VoiceXML es un lenguaje de gestión de diálogos muy potente, que ofrece una gran variedad de posibilidades de control sobre los servicios de reconocimiento y generación de voz. Sin embargo, el abanico de posibilidades que ofrece VoiceXML es tan grande que obliga a realizar un diseño muy fino de la implementación de la plataforma para conseguir admitir todos los servicios ofrecidos. Prueba de ello es que la gran mayoría de las plataformas VoiceXML que actualmente se pueden encontrar en el mercado (BeVOCAL café, Hey Anita, etc.) no cubren todas las funcionalidades que permite el lenguaje.

Sin embargo, y a pesar de la gran potencialidad de VoiceXML, se han detectado una serie de limitaciones en la definición de dicho lenguaje que merece la pena comentar (consultar también (Hamerich, 2003)):

- Hay una carencia o falta de estandarización en VoiceXML en la manera de lanzar los distintos reconocedores activos. Este tema se puede resolver de una manera bastante confusa, mediante la definición libre de propiedades (<property>). Si revisamos las propiedades generales de plataforma para el reconocimiento (Sharma, 2002), encontramos una serie de propiedades que deben ser admitidas por la plataforma pero no se especifica ningún mecanismo ni propiedad relativa a la carga y uso de un determinado reconocedor de habla durante la ejecución. Podemos ver en la Sección 3.1 cómo se ha utilizado esta propiedad.
- Tampoco hay estandarización en los aspectos relativos al control telefónico de llamadas. Se trata de un tema que aún esta un poco "verde", y que por lo tanto, actualmente no tiene mucho impacto sobre las implementaciones de VoiceXML. Esto hace, que la mayoría de desarrolladores de plataformas VoiceXML opten por no implementar la parte telefónica de la plataforma (Eberman, 2002). Así mismo, en el foro de discusión sobre VoiceXML y OpenVXI (VXIDiscuss, 2004), hay muy poco interés en los aspectos telefónicos de VoiceXML.
- Una de las grandes dificultades en VoiceXML está asociada al modelado del retorno en el flujo de diálogo. VoiceXML ofrece un mecanismo para gestionarlo pero desafortunadamente sólo está permitido dentro de un <form>. Consideramos que las llamadas a subdiálogo dentro de <fields> o <blocks> son claramente un requisito indispensable cuya inclusión se propone para futuras versiones de VoiceXML.
- Comparado con otros lenguajes de programación, VoiceXML carece de las herramientas comunes para implementar la lógica de un programa, en particular los bucles (i.e., while y for). Generalmente se recomienda emplear en su lugar guiones ECMA. Pero esta solución no sirve de nada

cuando se necesitan llevar a cabo operaciones entre fields o diálogos.

- Flujo de datos entre los guiones VoiceXML y bases de datos externas. El acceso a recursos externos, como por ejemplo bases de datos, solamente es posible a través de guiones CGI.
- La importación de datos podría mejorarse haciendo posibles las peticiones HTTP de forma explícita extendiendo las posibilidades de <subdialog> y <data>.
- Para facilitar la generación automática de guiones VoiceXML y para simplificar la reutilización en diferentes idiomas sería de gran utilidad establecer una representación externa de los prompts a partir de conceptos asociados a los mismos.

## *6 Conclusiones*

Se ha desarrollado una plataforma completa para sistemas de diálogo capaz de ejecutar guiones en VoiceXML aprovechando la funcionalidad ofrecida por OpenVXI. Se han presentado todos los detalles necesarios para conseguir la adaptación de un sistema existente a dicho entorno, destacando una serie de carencias tanto en OpenVXI como en VoiceXML que pueden ser aprovechados por los responsables del estándar.

Se han presentado los módulos que componen dicho sistema, destacando sus aspectos más novedosos y la utilización de estándares, como SSML y JSGF.

Durante el congreso podemos hacer una demostración del sistema.

## *7 Referencias*


Burnett, D. C., M. R. Walker, A. Hunt. 2002. "Speech Synthesis Markup Language Version 1.0". W3C Working Draft, http://www.w3.org/TR/speech-synthesis.

Córdoba, R., R. San-Segundo, J.M. Montero, J. Colás, J. Ferreiros, J. Macías-Guarasa, J.M. Pardo. 2001. "An Interactive Directory Assistance Service for Spanish with Large-Vocabulary Recognition", EUROSPEECH, pp. 1279-1282.

Córdoba, R., J. Macías-Guarasa, J. Ferreiros, J.M. Montero, J.M. Pardo. 2002. "State Clustering Improvements for Continuous HMMs in a Spanish Large Vocabulary Recognition System", ICSLP, pp. 677-680.

Córdoba, R., G. Prime, J. Macías-Guarasa, J.M. Montero, J. Ferreiros, J.M. Pardo. 2003. "PPRLM Optimization for Language Identification in Air Traffic Control Tasks", EUROSPEECH, pp. 2685-2688.

Eberman, B. 2002. "OpenVXI: Fostering VoiceXML via Open Source". WWW2002, USA. http://www.voicexmlreview.org/Mar2003/features.

Gemini. 2003. Página web www.gemini-project.org.

Hamerich, S. W., Y.F.H. Wang, V. Schubert, V. Schless, S. Igel. 2003. "XML-Based Dialogue Descriptions in the GEMINI Project". Proceedings of the "Berliner XML-Tage 2003", Berlín, Germany, pp. 404-412.

Lehtinen, G., S. Safra, …, J.M. Pardo, R. Córdoba, R. San-Segundo, et al. 2000. "IDAS: Interactive Directory Assistance Service", VOTS-2000 Workshop, Belgium.

Pardo, J.M., et al. 1995. "Spanish text to speech: from prosody to acoustic" International Conference on Acoustic, Vol. III.

San-Segundo, R., J.M. Montero, J. Colás, J. Gutiérrez, J.M. Ramos, J.M. Pardo. 2001. "Methodology for Dialogue Design in Telephone-Based Spoken Dialogue Systems: a Spanish Train Information System", EUROSPEECH, pp. 2165-2168.

Sharma, C., J. Kunins. 2002. "Strategies and techniques for effective voice application development with VoiceXML 2.0". Ed. Wiley.

SpeechWorks. 2004. Página web de la empresa SpeechWorks International, Inc.: http://www.speechworks.com/.

VXIDiscuss. 2004. Foro de discusión sobre VoiceXML y todos los aspectos relacionados con el intérprete OpenVXI 2.0: http://www.speechinfo.org/vxi-discuss/.