

MODELADO DE LA COMUNICACIÓN ORAL PERSONA-MÁQUINA – ESTÁNDAR VOICEXML

RECONOCIMIENTO AUTOMÁTICO DEL HABLA

MÁSTER DE INTELIGENCIA ARTIFICIAL, RECONOCIMIENTO DE
FORMAS E IMAGEN DIGITAL
DSIC – UNIVERSITAT POLITÈCNICA DE VALÈNCIA



Universidad
Carlos III de Madrid

David Griol Barres
Dpto. de Informática, Universidad Carlos III de Madrid

ÍNDICE

- **1.- Introducción**
- **2.- Aplicaciones**
- **3.- Clasificación**
- **4.- Antecedentes históricos**
- **5.- Arquitectura**
- **6.- Tecnologías involucradas**
- **7.- Retos actuales**
- **8.- Estándar VoiceXML**
- **9.- Conclusiones**
- Referencias

Introducción

- Un **sistema de diálogo** conforma un sistema que acepta lenguaje natural como entrada y genera lenguaje natural como salida, posibilitando una conversación mediante la voz con sus usuarios.
- **Importancia de estos sistemas:**
 - La comunicación mediante la voz es la más natural y sencilla.
 - Nos comunicamos utilizando lenguaje natural. Interfaces como el teclado o el ratón nos fuerzan a adaptarnos a los requisitos de las máquinas:
 - Mediante el uso de la voz posibilitamos que puedan utilizarse sistemas y aplicaciones en entornos en los que los interfaces tradicionales imposibilitan su uso (ej. Automóvil).
 - Facilita el acceso a estas aplicaciones para personas ancianas o con discapacidades.

Introducción

- La complejidad de estos sistemas puede variar desde la selección de una opción simple desde un menú a la creación de “compañeros conversacionales”.
- Recientemente, se han incorporado nuevas modalidades de entrada y salida diferentes de la voz en estos sistemas multiagente (gestos, escritura manuscrita, expresiones faciales, emociones...) → **Sistemas Multimodales**.
- **Retos actuales:**
 - Fallos de entendimiento, conocimiento sobre el contexto, flexibilidad, adaptación, aprendizaje de la experiencia.

Introducción

- **Ventajas de los sistemas de diálogo**
 - ahorro económico para empresas.
 - uso del teléfono: comodidad y flexibilidad.
 - comunicación natural y variada.
 - usuarios gozan de mayor grado de expresión.
 - no limitados a responder mediante determinados comandos.
- **Algunos retos actuales**
 - mayor robustez.
 - diálogo más natural e “inteligente”.
 - interacción multimodal.
 - interacción multilingüe.

ÍNDICE

- 1.- Introducción
- 2.- **Aplicaciones**
- 3.- Clasificación
- 4.- Antecedentes históricos
- 5.- Arquitectura
- 6.- Tecnologías involucradas
- 7.- Retos actuales
- 8.- Estándar VoiceXML
- 9.- Conclusiones
- Referencias

Aplicaciones actuales de los Sistemas de Diálogo

- **Algunas aplicaciones actuales**

- Control programas de ordenador.
- Aplicaciones telefónicas.
- Sistemas de dictado.
- Control de sistemas de automóviles.
- Sistemas de reconocimiento de locutores.
- Búsqueda de información.

Aplicaciones actuales de los Sistemas de Diálogo

- **Control de programas de ordenador**

- Iniciar aplicaciones (ej. Calculadora)
- Navegar en sistemas de menús
- Edición de documentos (“cursiva”, “negrita”, “cambiar fuente”, ...)
- Manos libres
- El reconocedor sabe a priori qué posible comandos recibirá oralmente del usuario



Aplicaciones actuales de los Sistemas de Diálogo

- **Aplicaciones telefónicas**

- Inicialmente, navegación mediante teléfonos de tonos (**DTMF, Dual Tone Multi Frequency**)
 - Se usan botones del teléfono
 - Cada número asociado a una función de la aplicación
 - “Pulse 1 para aceptar, pulse 2 para rechazar y pulse 4 para finalizar”
 - Problema: usuarios pueden olvidar asignación de números a funciones de la aplicación

Aplicaciones actuales de los Sistemas de Diálogo

- **Aplicaciones telefónicas**

- Posteriormente, sistemas **IVR (Interactive Voice Response)**
 - Utilizar palabras aisladas (p. e. “**uno**”, “**dos**”, “**si**”, “**no**”) en lugar de botones.
 - Adecuado para aplicaciones con número de funciones reducido.
 - Habla constituye forma alternativa de comunicación, no aporta capacidad de expresión (interacción muy limitada).
 - Suelen utilizar mensajes pregrabados en lugar de técnicas de síntesis de habla.

Aplicaciones actuales de los Sistemas de Diálogo

- **Sistemas de dictado**

- Hablar es más rápido que escribir.
- Modos entrada: texto o ejecución de comandos.
- Suelen requerir adaptación al locutor (entrenamiento).
- Vocabulario limitado (60.000 palabras aprox.)
- Adición de nuevas palabras por usuario.
- Ejemplos:
 - ViaVoice (IBM)
 - Voice Xpress (Lernout & Hauspie)

Aplicaciones actuales de los Sistemas de Diálogo



- **Control de sistemas de automóviles**

- Control de dispositivos manos libres (p. e. teléfono).
- Proporcionan mayor seguridad en conducción.
- Reconocimiento del habla muy difícil.
 - Gran ruido.
 - Aplicación de técnicas de reconocimiento robusto.
- Ejemplo: AutoPC (Clarion).
 - Vocabulario 1.200 palabras aprox.
 - Permite cambiar emisora de radio, leer e-mail, obtener direcciones mediante navegación GPS, etc.

Aplicaciones actuales de los Sistemas de Diálogo

- **Reconocimiento de locutores**

- **Tres tecnologías relacionadas:** identificación, detección y verificación de locutores.
- **Identificación de locutor:** determinar quién es, de entre un conjunto.
- **Detección de locutor:** detectar si hay (o no) cambio de locutor.
- **Verificación locutor:** comprobar identidad locutor.
- Aplicaciones:
 - Telebanco (p. e. comprobar estado de cuentas)
 - Compras a través de teléfono
 - Acceso a bases de datos
 - Aplicaciones forenses
 - Login mediante voz (ej. iMAC, de Apple)

ÍNDICE

- 1.- Introducción
- 2.- Aplicaciones
- 3.- **Clasificación**
- 4.- Antecedentes históricos
- 5.- Arquitectura
- 6.- Tecnologías involucradas
- 7.- Retos actuales
- 8.- Estándar VoiceXML
- 9.- Conclusiones
- Referencias

Clasificación de los Sistemas de Diálogo

- **Dominio de aplicación**
 - Tarea realizada por el sistema, Ejemplos:
 - ATIS (Air Travel Information Service).
 - TRAINS.
 - etc.
- **Tipo de comunicación**
 - visual / escrita /oral.
 - unimodal / multimodal.
- **Estrategia de interacción**
 - dirigida por sistema (system-directed).
 - dirigida por usuario (user-directed).
 - mixta (mixed-initiative).
 - ventajas / desventajas.

Clasificación de los Sistemas de Diálogo

- **Tipos de discurso**
 - ¿De qué forma pueden hablar los usuarios?
 - Directamente relacionado con el tipo de reconocedor utilizado
 - Reconocimiento de **palabras aisladas** (IWR, Isolated Word Recognition)

“POR FAVOR, DIGA ‘UNO’ SI DESEA RECIBIR INFORMACIÓN METEOROLÓGICA, DIGA ‘DOS’ SI QUIERE HACER UNA RESERVA, Y DIGA ‘TRES’ SI DESEA REALIZAR OTRA OPERACIÓN”.
 - Reconocimiento de **palabras conectadas** (CWR, Connected Word Recognition). No se suelen usar en sistemas de diálogo.
 - Reconocimiento de **voz continua** (CSR, Continuous Speech Recognition).
 - Reconocimiento de **palabras-clave** (Word Spotting).

Clasificación de los Sistemas de Diálogo

- **Clasificación de sistemas de diálogo**
 - **Sistemas diálogo oral (unimodal)**: sólo usan habla.
 - **Sistemas diálogo multimodal**: usan varios canales de comunicación.
 - **Entrada**: habla, movimiento labios, gestos usuario, etc.
 - **Salida**: habla, imágenes, gráficos, etc.
 - **Sistemas diálogo multilingüe**: permiten interacción mediante varios idiomas (p. e. Castellano e Inglés).

ÍNDICE

- 1.- Introducción
- 2.- Aplicaciones
- 3.- Clasificación
- 4.- **Antecedentes históricos**
- 5.- Arquitectura
- 6.- Tecnologías involucradas
- 7.- Retos actuales
- 8.- Estándar VoiceXML
- 9.- Conclusiones
- Referencias

Antecedentes históricos

- El desarrollo de máquinas capaces de hablar con las personas ha constituido una meta muy perseguida desde hace siglos:
 - En antigüedad: estatuas de dioses supuestamente podían hablar (ej. estatuas de divinidades en la mitología griega).
 - En nuestra época: **modelos acústicos** (réplicas “mecánicas” del tracto vocal) y **electroacústicos** (sistemas eléctricos/electrónicos)
 - Los primeros intentos serios de desarrollar sistemas que hablaban se remontan a los siglos XVIII y XIX. Autómatas que imitan comportamientos humanos.

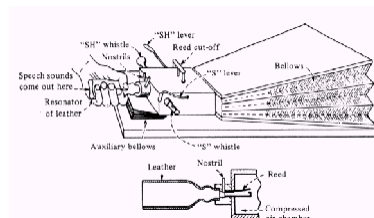


Antecedentes históricos

- **C. G. Kratzenstein (1779)**
 - Fuelle + lengüeta + cámara de resonancia
 - Sonidos vocálicos
- **W. Von Kempelen (1791)**
 - Cabeza parlante → Primer autómata capaz de pronunciar palabras.
 - Fuelles + lengüetas + cámaras de resonancia de forma variable
 - Sonidos vocálicos y consonánticos
 - Frases completas

Antecedentes históricos

- **Faber (1846)**
 - Órgano parlante capaz de imitar la producción vocal humana.



- **Sir Charles Wheatstone (1891)**
 - versión mejorada de máquina de Von Kempelen

Antecedentes históricos

- **R. Riesz (1937)**
 - lengüeta de longitud variable
 - entonación de frases
- **J. Q. Stewart (principios siglo XX)**
 - Generación de la voz mediante dispositivos eléctricos (Von Herlmholtz)
 - dos resonadores acoplados excitados mediante pulsos eléctricos
 - diferentes frecuencias permitían generar sonidos vocálicos

Antecedentes históricos

- **H. Dudley, R. Reiz y S. Watkins (1930)**
 - **vocoder** : primera máquina eléctrica capaz de generar frases
 - controlada mediante operador humano
 - teclado para controlar diversos parámetros
 - pedal para control de *pitch* (entonación)
 - mostrada en Exposición Mundial de Nueva York, 1939

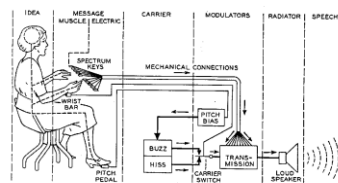


Fig. 8—Schematic circuit of the vocoder.

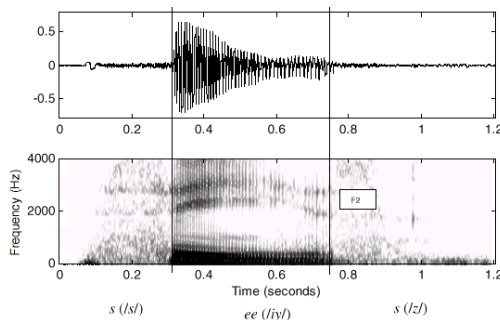


Antecedentes históricos

- **1940s**
 - Se desarrollan los primeros ordenadores durante la Segunda Guerra Mundial:
 - 1938 – 1941: Kornad Zuse, primer computador programable.
 - 1943: Colossus, Bletchley Park.
 - 1943 – 1945: Univ. Pennsylvania, ENIAC.
 - Se empieza a considerar el potencial de estas máquinas para otros propósitos:
 - Allan Turing.
 - Warren Weaver.

Antecedentes históricos

- **F. S. Cooper, A. M. Liberman y J. M. Borst (1951)**
 - espectrograma = representación visual de señal de voz



- dispositivo eléctrico: producía voz a partir de luz y rueda giratoria con 50 círculos concéntricos de densidades variables
- la máquina transformaba a voz la información codificada en el espectrograma

Antecedentes históricos

- **1950s**
 - 50 - Alan Turing – Máquinas Inteligentes.
 - 51 – Cooper, Liberman and Borst – Espectrograma.
 - 57 – Chomsky – Estructuras Sintácticas.
 - 58 – Peterson, Wang y Siversten – Difonemas.
- **1960s**
 - Primeros agentes conversacionales, e.g. Weizenbaum ELIZA → No Lingüística Computacional.
- **1970s**
 - Lingüística Computacional.
 - Basada en la teoría sobre Semántica y Lenguaje desarrollada por Chomsky, Montague, Wood.
 - Síntesis de voz basada en reglas.

Antecedentes históricos

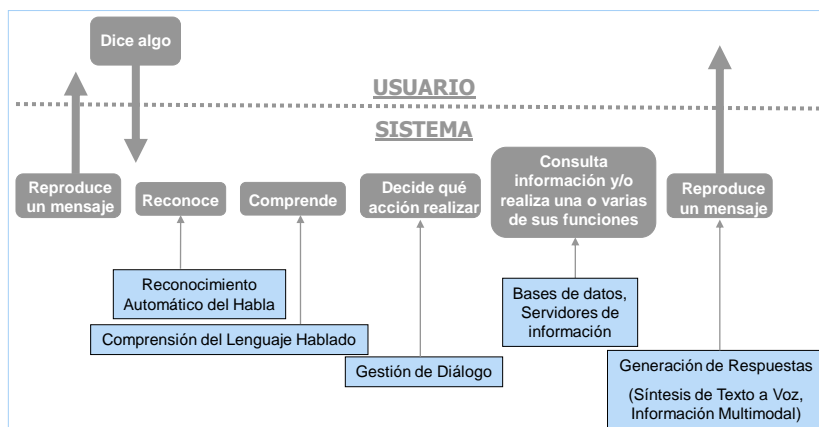
- **1980s**
 - Servicios de telefonía.
 - Sistemas de dictado.
- **1990s**
 - Desarrollos de corpus: Wordnet.
 - Reconocimiento aislado de palabras. Sistemas de diálogo telefónicos.
 - Compañías dedicadas.
- **2000s**
 - Sistemas estadísticos (reglas todavía en uso).
 - Estándares para facilitar el desarrollo (VoiceXML, XHTML+Voice).
 - Usabilidad, sistemas adaptados al usuario.

ÍNDICE

- 1.- Introducción
- 2.- Aplicaciones
- 3.- Clasificación
- 4.- Antecedentes históricos
- **5.- Arquitectura**
- 6.- Tecnologías involucradas
- 7.- Retos actuales
- 8.- Estándar VoiceXML
- 9.- Conclusiones
- Referencias

Arquitectura de un Sistema de Diálogo

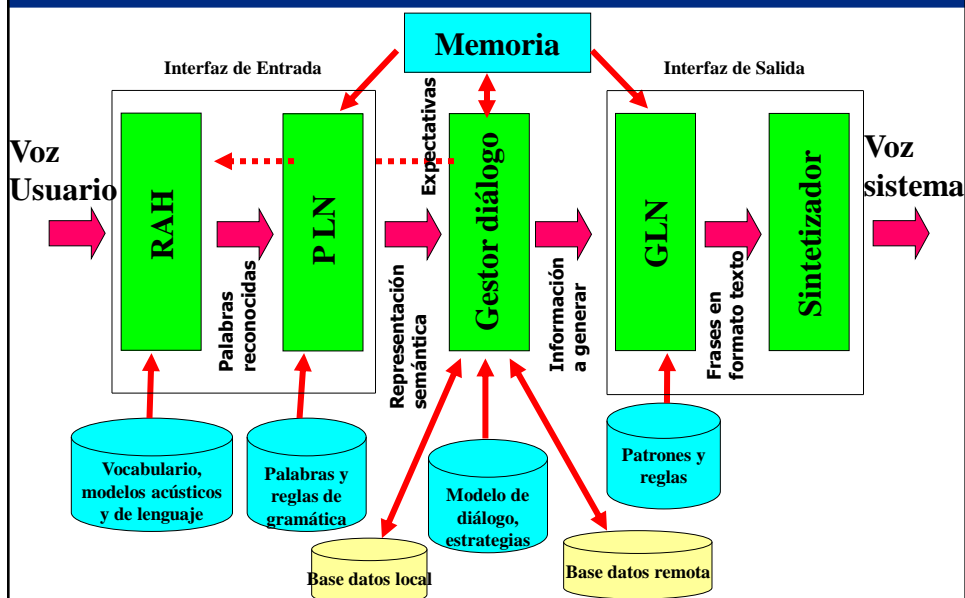
Sistema de Diálogo: Sistema multiagente capaz de emular a un ser humano en un diálogo con otra persona.



David Griol – UC3M

29

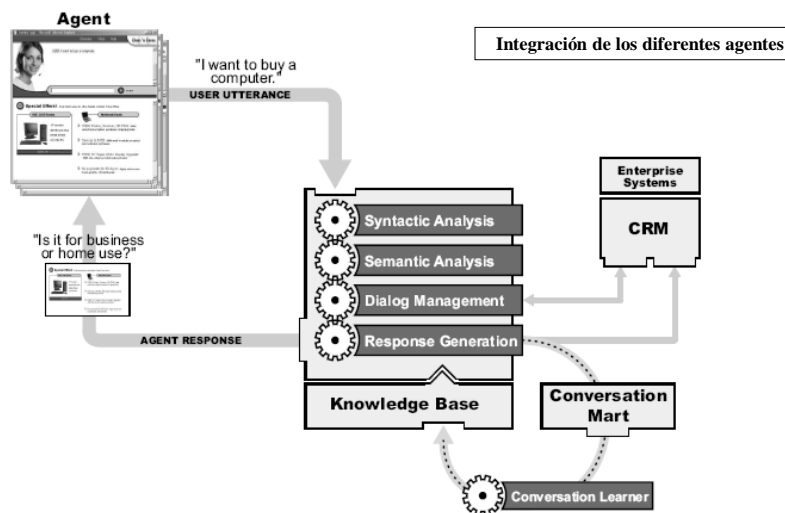
Arquitectura de un Sistema de Diálogo



David Griol – UC3M

30

Arquitectura de un Sistema de Diálogo



David Griol – UC3M

31

ÍNDICE

- 1.- Introducción
- 2.- Aplicaciones
- 3.- Clasificación
- 4.- Antecedentes históricos
- 5.- Arquitectura
- 6.- **Tecnologías involucradas**
- 7.- Retos actuales
- 8.- Estándar VoiceXML
- 9.- Conclusiones
- Referencias

David Griol – UC3M

Tecnologías involucradas

- **Tecnologías utilizadas en los agentes conversacionales actuales**
 - Reconocimiento Automático del Habla (RAH).
 - Procesamiento de Lenguaje Natural (PLN).
 - Gestión diálogo.
 - Generación de Lenguaje Natural (GLN).
 - Síntesis del habla.

Tecnologías involucradas: RAH

- **Interfaz de entrada**
 - **Reconocedor de habla (Speech Recogniser)**
 - proporciona secuencia de palabras reconocidas .
 - una o varias hipótesis de reconocimiento.
 - valores de confianza.

Tecnologías involucradas: RAH

- **Problemas relacionados con el RAH**

- **Coarticulación:** fonemas se ven afectados por fonemas vecinos.
- **Variabilidad acústica:** cada persona pronuncia los sonidos de forma diferente.
- **Variabilidad lingüística:** distintos dialectos de un mismo idioma.
- **Confusión acústica:** similitud entre palabras que suenan de forma parecida (p. e. estalagmita / estalagmita).
- **Tamaño del diccionario:** al crecer, aumenta probabilidad de error.

Tecnologías involucradas: RAH

- **Problemas relacionados con el RAH**

- **Ruidos e interferencias:** provocan errores de reconocimiento.
- **Palabras OOV (*Out-of-vocabulary*):** palabras no previstas, por no pertenecer al dominio de aplicación (no incluidas en diccionario).
- **Frases no permitidas por la gramática:** frases sin sentido, incorrectas gramaticalmente.
- **Cross-talk:** terceras personas hablando cerca del micrófono.
- **Otros factores:** estado anímico del locutor, resfriados, etc.

Tecnologías involucradas: RAH

- **Problema de la coarticulación**
 - habla = secuencia de sonidos (fonemas).
 - secuencia de fonemas forma sílabas, y éstas forman palabras.
 - cada fonema tiene características articulatorias y acústicas únicas.
 - esta información es muy importante para poder distinguir palabras ("bit" / "pit").
 - **problema**: conexión de fonemas para formar unidades mayores cambia características acústicas de dichos fonemas (es decir, los fonemas se ven afectados por los fonemas vecinos).
 - puede ocurrir
 - dentro de una palabra (ej. /t/ en "tea", "tree", "city", "steep", etc.)
 - entre palabras (ej. "this" en "this car" o "this ship" p.e.)

Tecnologías involucradas: RAH

- Retos:
 - Variabilidad:
 - Lingüística.
 - Interlocutor.
 - Intralocutor.
 - Del canal.
 - Del contexto.
 - Efecto Cocktail party.
 - Efecto Lombard.



Tecnologías involucradas: RAH

- **Posibles errores introducidos durante el RAH**
 - Como consecuencia de los problemas comentados, aparecen tres tipos de errores:
 - **Inserciones:** en frase reconocida hay más palabras que en frase pronunciada.
 - **Borrados:** en frase reconocida hay menos palabras que en frase pronunciada.
 - **Sustituciones:** en frase reconocida algunas palabras son cambiadas.

Tecnologías involucradas: RAH

- **Número de usuarios o locutores:**
 - **Monolocutor:**
 - Entrenados con la voz de un único locutor.
 - Tasas de reconocimiento altas.
 - Necesarios para locutores con problemas de dicción.
 - **Independiente del locutor:**
 - Reconocen voz de usuarios distintos a los de entrenamiento.
 - Se reducen las tasas de reconocimiento.
 - Necesarios para aplicaciones telefónicas.
 - **Adaptados al locutor:**
 - Múltiples locutores realizan el entrenamiento.
 - Adaptación posterior a un locutor (comportamiento como monolocutor).

Tecnologías involucradas: RAH

- **Estilo de habla:**
 - **Palabras aisladas (IWR, Isolated Word Recognition):**
 - Locutor realizar pausas de gran duración entre las palabras.
 - Facilidad localización y fin de las palabras.
 - **Palabra conectadas (CWR, Connected Word Recognition):**
 - Locutor debe hacer pausa breves entre palabras.
 - **Habla continua (CSR, Continuous Speech Recognition):**
 - Locutor no necesita hacer pausas, habla de forma normal.
 - Mayores problemas de coarticulación en palabras.
 - No hay conocimiento de separación entre palabras.

Tecnologías involucradas: RAH

- **Estilo del habla:**
 - **Habla leída:**
 - Locutor debe hablar como si estuviera leyendo.
 - **Habla espontánea:**
 - Modo natural de comunicación entre las personas.
 - Locutor puede hablar naturalmente.
 - Fenómenos del habla espontánea (titubeos, repeticiones, ...)
 - **Palabras clave (Word Spotting):**
 - Sistema sólo debe reconocer determinadas palabras, no todas las palabras pronunciadas.

Tecnologías involucradas: RAH

- **Implementación:**

- **Aproximación estadística: Modelos ocultos de Markov**

- La aproximación estadística es actualmente la más utilizada: encontrar la secuencia de palabras W pronunciadas dada una secuencia de datos acústicos A :

$$P(W|A) = \frac{P(A|W)P(W)}{P(A)}$$

- Necesidad de datos de entrenamiento.
 - RAH disponibles: HTK, Sphinx, Loquendo...

Tecnologías involucradas: PLN

- **Interfaz de entrada**

- **Procesador de lenguaje natural (Semantic Analyser)**

- obtiene significado de frases de usuario (generalmente mediante **frames**).
 - cambio en el lenguaje de representación, de lenguaje natural a un lenguaje semántico, de forma que se mantenga el significado del mensaje.

Tecnologías involucradas: PLN

- **Algunas aplicaciones:**
 - Traducción automática.
 - Extracción y procesamiento de información.
 - P.e. Question Answering, búsqueda tema.
 - Resumen y simplificación de textos.
 - P.e. Generación de resúmenes, Generación de informes.
 - Clasificación, recuperación y filtro de información.
 - P.e. Filtros de correo, web semántica.
 - Comprobación gramatical lenguaje.
 - P.e. Correctores ortográficos y gramaticales, OCR.
 - Agentes conversacionales.

Tecnologías involucradas: PLN

- **Implementación**
 - Metodologías basadas en reglas.
 - Extraen la información semántica a partir del análisis sintáctico-semántico de las frases:
 - gramáticas definidas para la tarea,
 - detección de palabras (o secuencias de palabras) clave, con significado semántico.
 - Metodologías estadísticas.
 - Definición de unidades lingüísticas con contenido semántico y obtención de modelos a partir de muestras etiquetadas.
 - Entrenamiento (aprendizaje) del modelo: capturar las correspondencias entre las entradas de texto y su representación semántica.
 - Necesario datos de entrenamiento etiquetados.

Tecnologías involucradas: PLN

- **Problemas relacionados con el PLN**
 - **Elipsis** (omisión de palabras).
 - Ej. *“uno de lomo con queso”* (¿a qué se refiere “uno”?)
 - **Anáfora**
 - Ej. *“lo quiero grande”* (¿a qué se refiere “lo”?)
 - **Ambigüedad** (léxica y estructural)
 - Ej. *“vale”* (¿qué quiere decir “vale”?)
 - Ej. *“Mi hermana vio a un niño jugando en el parque con un telescopio”* (¿cuál es el significado?)
 - **Derivados de errores de RAH**
 - inserciones, borrados y sustituciones pueden provocar errores en análisis lingüístico de las frases.

Tecnologías involucradas: PLN

- La complejidad del lenguaje natural reside en su ambigüedad:
 - **Léxico**:
 - La misma palabra puede pertenecer a categorías gramaticales distintas.
 - **Sintáctico**:
 - Frases con varias estructuras sintácticas posibles.
 - **Semántico**:
 - Palabras polisémicas.
 - Misma estructura sintáctica, distintos significados.
 - **Discurso**:
 - Ambigüedad referencial.
 - **Pragmático**:
 - Ironía, sarcasmo, doble sentido.

Tecnologías involucradas: PLN

- Y si además se trata de lenguaje natural oral:
 - Léxico y morfológico:
 - Palabras homófonas.
 - Formas coloquiales.
 - Fragmentos, repeticiones, dubitaciones.
 - Quasi-léxico (p.ej. “ujum...”).
 - Clases de palabras propias del lenguaje oral (p.ej. marcadores del discurso como “bueno”).
 - Interjecciones.
 - Expresiones multi-palabra (p.ej. “ya veo”).
 - Sintáctico:
 - Estructuras incompletas.
 - Reparaciones.
 - Repeticiones.
 - Combinaciones sintácticas.

Tecnologías involucradas

- **Interfaz de entrada**
 - Modalidades adicionales de entrada si el sistema es **multimodal**
 - Gestos del usuario
 - Movimiento de labios de usuario
 - Dirección de mirada de usuario
 - Expresiones faciales de usuario
 - Texto escrito por usuario (Ej. Usando un PDA)
 - Etc.

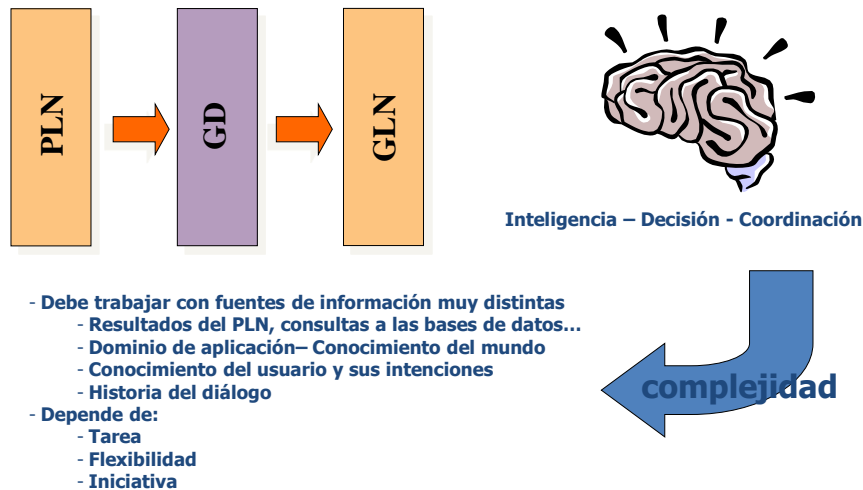
Tecnologías involucradas: Gestor del Diálogo

- **Gestión del diálogo (Dialogue manager):**
 - Decidir la siguiente respuesta del sistema:
 - ¿Qué información dar? → Historia del diálogo
 - ¿De dónde obtenerla? → Manejar diversas fuentes de información.
 - Gestionar la interacción:
 - Iniciativa del diálogo.
 - Recuperación de errores:
 - ¿Se ha entendido bien la petición del usuario?
 - ¿Cómo corregirla?
 - Estrategias de confirmación.

Tecnologías involucradas: Gestor del Diálogo

- **Gestión del diálogo (Dialogue manager):**
 - Mantener historia del diálogo:
 - ¿Qué información es necesario almacenar? ¿Cómo?
 - Aspectos avanzados:
 - Aprender de la experiencia.
 - Estrategias adaptativas.
 - Estrategias predictivas.

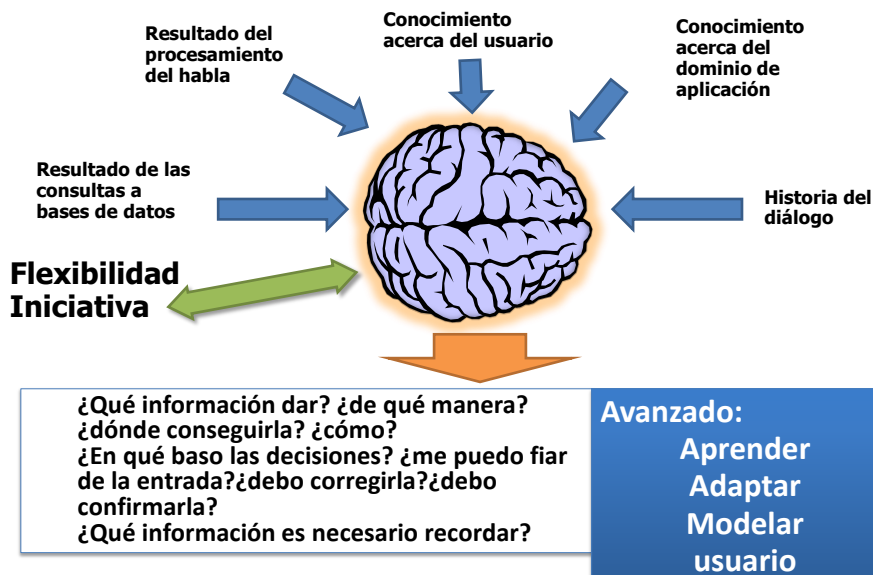
Tecnologías involucradas: Gestor del Diálogo



David Griol – UC3M

53

Tecnologías involucradas: Gestor del Diálogo



David Griol – UC3M

54

Tecnologías involucradas: Gestor del Diálogo

- No existe un consenso sobre las tareas que debe llevar a cabo. Un listado breve lo componen:
 - Decidir el próximo turno de sistema y qué información proporcionar.
 - ¿Qué información debe proporcionarse/requerirse? ¿Cómo?
 - ¿De dónde obtener esta información? Múltiples fuentes
 - ¿Cómo tomar esta decisión? ¿A partir del último turno de usuario? ¿A partir de toda la historia del diálogo?
 - Gestionar la interacción
 - ¿Quién tiene la iniciativa del diálogo?
 - Detectar/corregir errores de reconocimiento/compreensión
 - ¿La entrada al gestor del diálogo es correcta?
 - Si no lo es, ¿cómo corregirla?

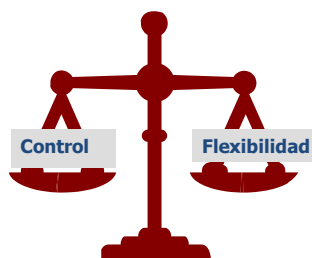
Tecnologías involucradas: Gestor del Diálogo

- No existe un consenso sobre las tareas que debe llevar a cabo. Un listado breve lo componen:
 - Gestionar las confirmaciones
 - ¿Qué información es necesario confirmar?
 - ¿Cómo?
 - Gestionar la historia del diálogo
 - ¿Qué información es necesario almacenar?
 - ¿Cómo?
 - Avanzado: Aprender de la experiencia, adaptar estrategias, construir modelos de usuario, incorporar nuevas fuentes de información...

Tecnologías involucradas: Gestor del Diálogo



- La iniciativa puede ser:
 - Dirigida por el sistema
 - Dirigida por el usuario (pionero: How may I help you? AT&T)
 - Mixta
 - Estática – Siempre el mismo tipo
 - Dinámica – Varía durante la interacción
- Estrategias de confirmación:
 - Ímplicita
 - Explícita
 - Mixta



Tecnologías involucradas: Gestor del Diálogo

- **Posibles metodologías para su implementación**
 - Técnicas basadas en reglas:
 - Definir cada una de las reglas que compondrán la estrategia y aplicarlas según se presenten las condiciones requeridas (estándar VoiceXML).
 - Metodologías estadísticas:
 - Aprendizaje de un modelo a partir de los casos disponibles en un corpus de diálogos previamente etiquetado.

Tecnologías involucradas: Gestor del Diálogo

Gestión del diálogo



Enrutamiento de llamadas
Consulta de información
Reservas
Transacciones bancarias
Control remoto
Tutorización
Soporte al diseño
Negociación
"Companions"

Tecnologías involucradas: Almacenamiento de la Información

- **Memoria (Memory Module)**
 - Almacena datos de interacciones de usuarios.
 - Proporciona historia del diálogo al gestor del diálogo.
- **Bases de datos (Database)**
 - Contienen información solicitada por usuarios.

Tecnologías involucradas: GLN

- **Interfaz de salida**
 - **Generador de lenguaje natural (Natural Language Generator o Response Generator)**
 - Genera respuestas en formato de texto (plantillas).
 - **Sintetizador de voz (Speech Synthesizer)**
 - Concatenación de palabras. Poco flexible, gran inteligibilidad.
 - Concatenación de unidades menores que las palabras (p. e. difonemas). Gran flexibilidad, menos inteligibilidad.

Tecnologías involucradas: GLN

Generación del lenguaje natural

- . **Organización del contenido.**
- . **Distribución en frases.**
- . **Lexicalización.**
- . **Generación de referencias.**
- . **Realización lingüística.**

Id-mensaje: 000
Relación: PROYECCIÓN
Argumentos:

Película: VÉRTIGO
Sala: 2
Sesión: 18:30



- . *Incluir conocimiento del dominio de aplicación.*
- . *Riqueza gramatical y léxica.*
- . *Organización del discurso (retórica, argumentación)*
- . *Conocimiento del usuario.*

Tecnologías involucradas: GLN

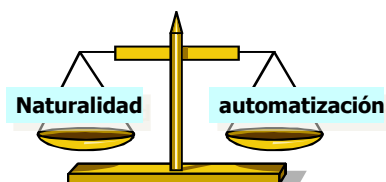
- Síntesis del habla:

- **Técnicas paramétricas:**

- Modelan parámetros fisiológicos relativos al tracto vocal para producir sonidos de forma artificial.
 - Basadas en formantes → Se fundamenta en variables (formantes) que modelan la vibración de las cuerdas vocales.

- **Técnicas sin fundamento físico:**

- Síntesis concatenativa → Unir trozos pregrabados de voz humana. ¿unidades?



Tecnologías involucradas: GLN

- Síntesis del habla:

- Basada en producción humana:

- Paramétrica y basada en formantes.



Silabización
Acentuación
Entonación
Cadencia

- Sin base fisiológica:

- Síntesis concatenativa.



Tecnologías involucradas: GLN

– Problemas relacionados con la GLN

- Generar frases sintáctica y semánticamente correctas
 - ¿Quieres *uno* bocadillo de lomo y *una* coca-cola?
 - ¿Has dicho que quieres una cerveza *de limón*?
- Elegir las palabras y expresiones adecuadas para un contexto dado
 - *Anáfora*: ¿Has dicho que *la* quieres de tamaño grande?
 - *Elipsis*: El bocadillo cantábrico cuesta 2 euros y el (...) de lomo cuesta 2 euros con 20 céntimos

Tecnologías involucradas: GLN

• Problemas relacionados con la síntesis de habla

- Lograr gran *inteligibilidad*
 - Las frases deben ser fácilmente entendibles
- Lograr gran *naturalidad*
 - La voz sintética debe tener una gran semejanza con la voz real
- Elección de unidades lingüísticas a usar
 - *Fonemas*: menor naturalidad, más complejo, más flexible
 - *Difonemas*: menor naturalidad, más complejo, más flexible
 - *Trifonemas*: menor naturalidad, más complejo, más flexible
 - *Palabras*: gran naturalidad, simple, poco flexible
 - *Frases*: gran naturalidad, simple, muy poco flexible

Tecnologías involucradas: GLN

- **Problemas relacionados con el tiempo de respuesta**
 - usuarios **no aceptan** sistemas lentos, especialmente en aplicaciones telefónicas
- **Problemas relacionados con expectativas de usuarios**
 - usuarios no suelen ser conscientes de limitaciones técnicas
 - esperan que los sistemas funcionen casi perfectamente

Tecnologías involucradas: Multimodalidad

- **Interfaz de salida**
 - Modalidades adicionales de salida si el sistema es **multimodal**
 - gráficos
 - agente animado (cara o cuerpo), movimientos sincronizados
 - otros sonidos
 - etc.

Tecnologías involucradas : Multimodalidad

- **Sistema de diálogo multimodal**

- Información adicional en interfaz de entrada

- Movimiento de labios de usuario, dirección mirada usuario,
 - Gestos de usuario, texto escrito por usuario, etc.



- Información adicional en interfaz de salida

- Gráficos
 - Otros sonidos
 - Agente animado
 - Expresiones faciales
 - gestos y movimiento de labios
 - sincronización con respuesta oral



Tecnologías involucradas: Multilingüe

- **Sistema de diálogo multilingüe**

- Estrategias

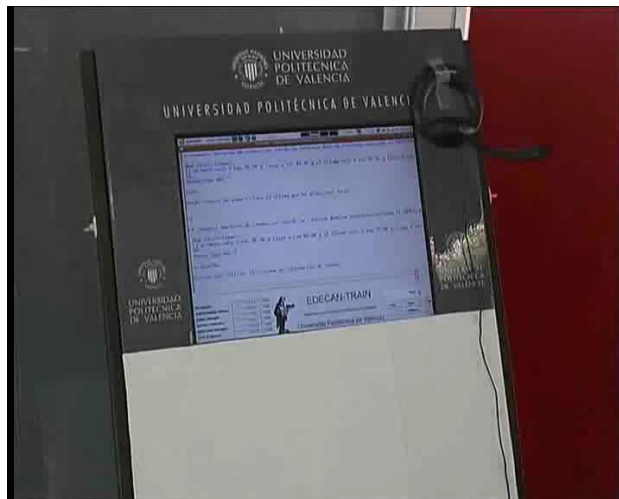
- Elección idioma al inicio del diálogo
 - Detección automática del idioma

- Ideas fundamentales

- Extracción de **misma representación semántica** independientemente del idioma utilizado (*interlingua*)
 - Crear módulos que sean **independientes del idioma**
 - Información dependiente de idioma almacenada externamente
 - Estrategias de interacción no cambian
 - Usar sistema TTS multilingüe (Ej. Festival)

Proyecto DIHANA

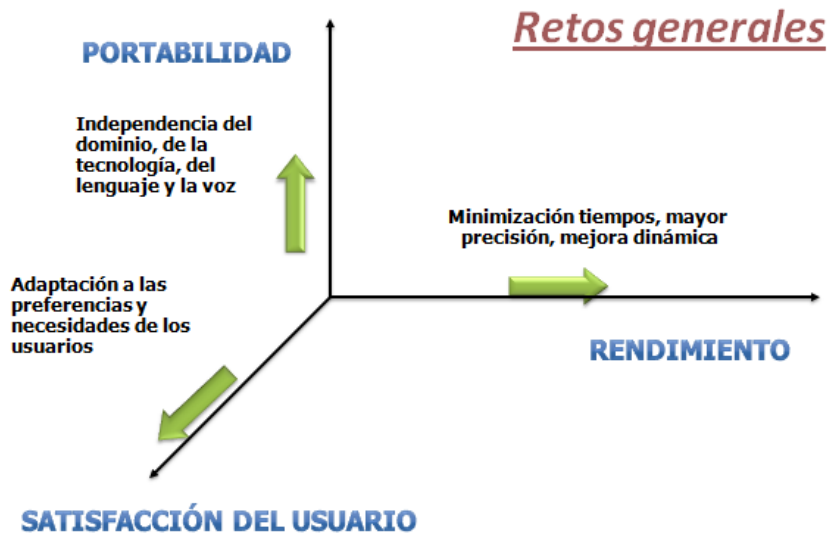
- **Desarrollo de la totalidad de módulos utilizando metodologías estadísticas** (Griol et al, 2008)



ÍNDICE

- **1.-** Introduccción
- **2.-** Aplicaciones
- **3.-** Clasificación
- **4.-** Antecedentes Históricos
- **5.-** Arquitectura
- **6.-** Tecnologías involucradas
- **7.- Retos actuales**
- **8.-** Estándar VoiceXML
- **9.-** Conclusiones
- Referencias

Retos Actuales



Retos Actuales

Proactividad

- Sistemas que actúan de forma autónoma.
- Clave para:
 - Sistemas ubicuos.
 - Agentes conversacionales.
- El sistema debe ser capaz de:
 - Registrar diálogos.
 - Capturar el contexto del diálogo.
 - Extraer información relevante.
 - Determinar cuándo participar.
- Debemos dotarlo de:
 - Capacidad de resolución de problemas.
 - Sensibilidad al contexto.



Smart Shopping Assistant

Comparison Chart

	Carlini	Pasta Superiore
Product Name	Carlini	Pasta Superiore
Price (EUR)	2.49	4.29
Weight (g)	375	500
Ecological produced	NO	YES
Calories (kcal/100g)	153	364

Smart Shopping Assistant

Do you want to prepare a pasta dish?

If yes, the following products may be useful:

- Pasta Sauce Deluxe
- Pasta Sauce Vegetarian
- Dry Wine
- Milk
- Cheese
- Salt

Retos Actuales

Adaptación

- Niveles de adaptación al usuario:
 - Perfiles personales → Opciones estáticas.
 - Adaptación a destrezas de los usuarios (p.ej. Noveles vs. Expertos).
 - Adaptación a necesidades especiales:
 - No nativos.
 - Discapacitados.
 - Niños y ancianos.
- Sistemas multilingües.

Retos Actuales

Adaptación


- Inteligencia emocional:
 - Sistemas capaces de:
 - Detectar el estado emocional de los usuarios.
 - Modificar su comportamiento de acuerdo al mismo.
 - Dar respuestas emocionales.
 - Los sistemas empáticos contribuyen a una valoración más positiva de los sistemas y aportan fluidez a la interacción usuario-máquina.

ÍNDICE

- 1.- Introducción
- 2.- Aplicaciones
- 3.- Clasificación
- 4.- Antecedentes Históricos
- 5.- Arquitectura
- 6.- Tecnologías involucradas
- 7.- Retos actuales
- 8.- **Estándar VoiceXML**
- 9.- Conclusiones
- Referencias

David Griol – UC3M

Puntos a tratar

- Estándares
 - VoiceXML 
 - CCXML
 - CalXML
- Herramientas de desarrollo
 - CSLU Toolkit
 - IBM WebSphere Voice Toolkit
 - **Voxeo Designer**
 - Otras herramientas


David Griol – UC3M

78

VOICE XML: Objetivos

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

VoiceXML - Índice

- Introducción 
- Arquitectura
- Conceptos
- Construcción del diálogo
- Control de flujo
- Contenido ejecutable
- Entrada del usuario
- Salida del sistema
- Aspectos avanzados

VoiceXML: Introducción

- 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

- 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: 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: Inconvenientes

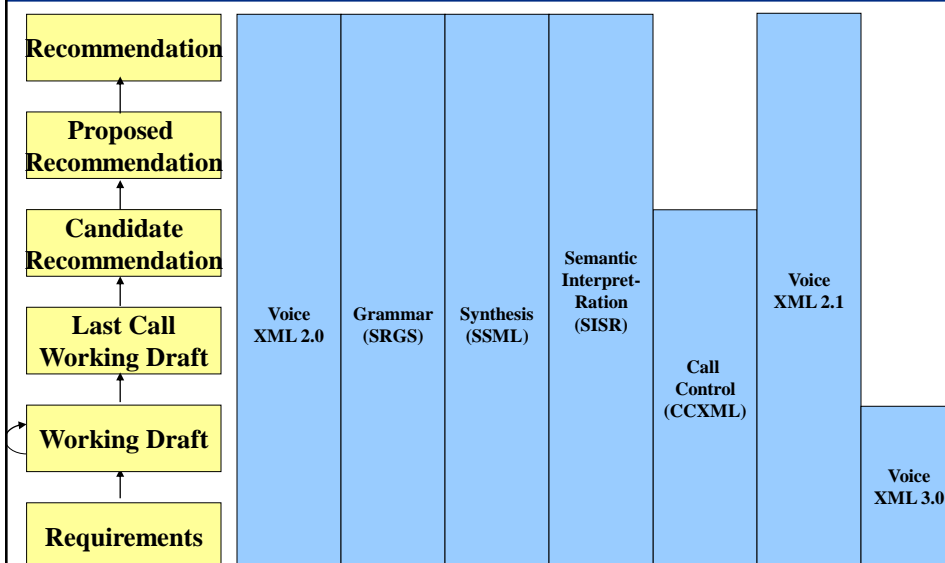
- Inconvenientes:
 - Acceso telefónico
 - No adecuado para realizar análisis de la voz: tratamiento de enfermedades, lectura de textos, etc.
 - Implementaciones parciales de la norma.

XML

- XML = eXtensible Markup Language
- Elementos entre etiquetas:
`<prompt>Bienvenido al sistema</prompt>`
- Elementos normalmente pareados
`<prompt>`
 Bienvenido a nuestra tienda

 tenemos las mejores ofertas
`</prompt>`
- Elementos quizás con atributos
`<choice next="#boat">`
`<grammar type="application/grammar+xml" version="1.0"`
`root = "by_boat" src = "boat.grxml">`
- Dado que "<", ">", y "&" tienen significados distintos:
 "<," en vez de "<"
 ">," en vez de ">"
 "&," en vez de "&".

Lenguajes relativos al Habla del W3C




Enlaces importantes

- Especificaciones VoiceXML:
 - <http://www.w3.org/TR/voicexml20/>
- Ejemplos:
 - <http://www.vxml.org/>
 - <http://www.optimsys.cz/en/optimsys/voicexml-examples>
- Plataforma Voxeo Evolution:
 - <http://evolution.voxeo.com/>

Ejercicio 1

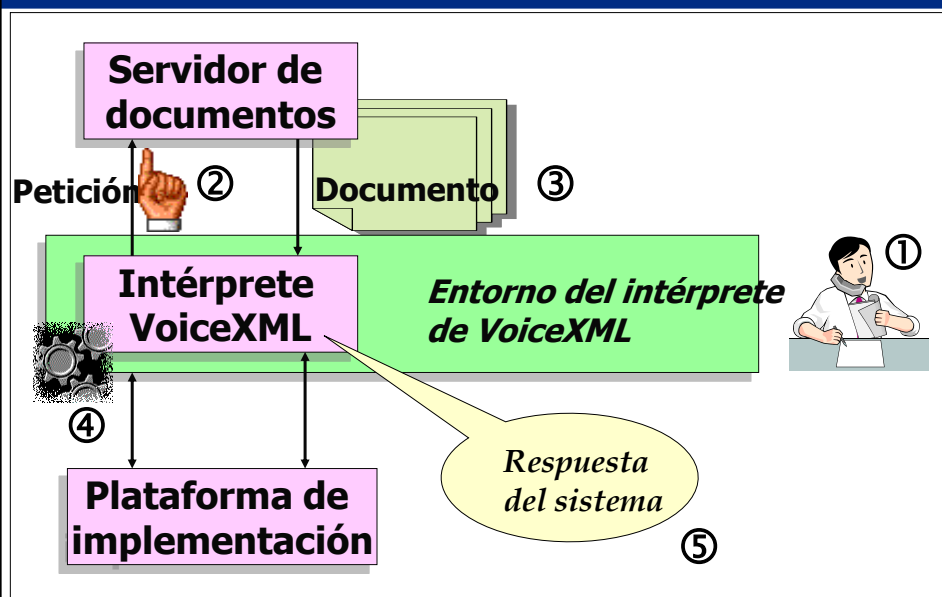
- Describe una aplicación de las tecnologías del habla que utilizarías en tu trabajo.
- Describe una aplicación de las tecnologías del habla que tú o tu familia utilizaríais en casa.

VoiceXML - Índice

- Introducción
- Arquitectura 
- Conceptos
- Construcción del diálogo
- Control de flujo
- Contenido ejecutable
- Entrada del usuario
- Salida del sistema
- Aspectos avanzados

David Griol – UC3M

Arquitectura VoiceXML



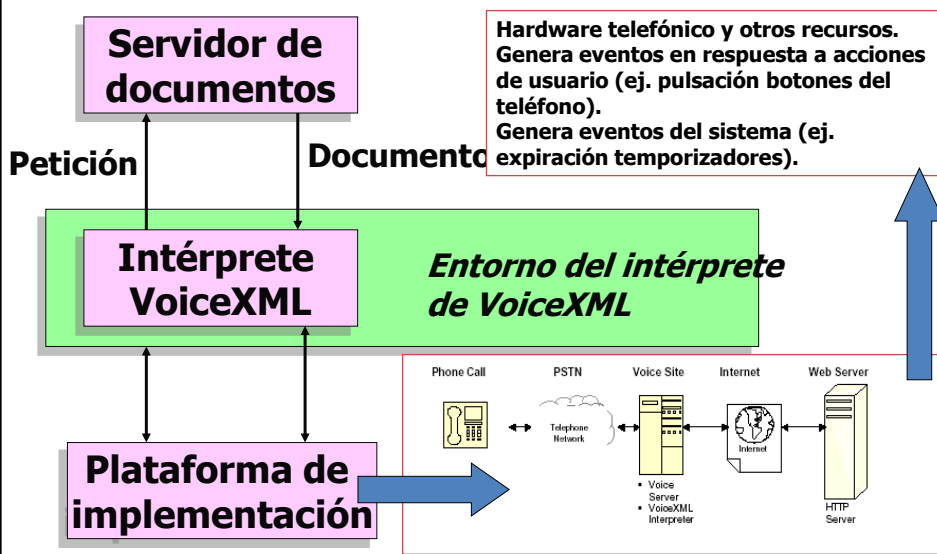
David Griol – UC3M

90

Arquitectura VoiceXML

- **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

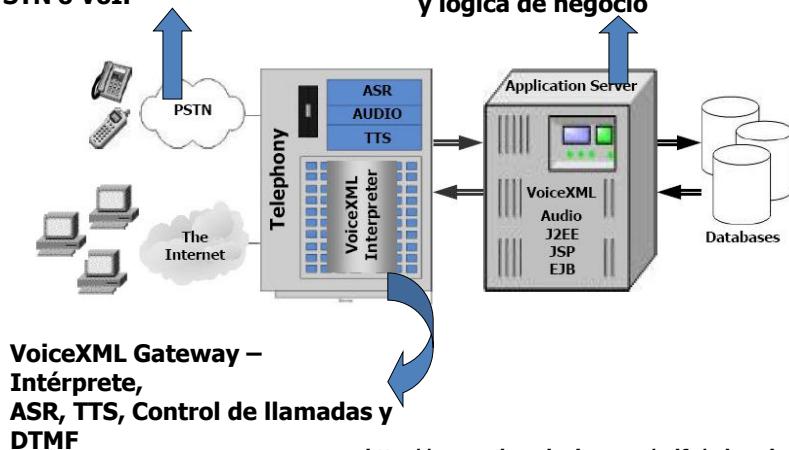
Arquitectura VoiceXML



VoiceXML Distribuido – Ejemplo 1

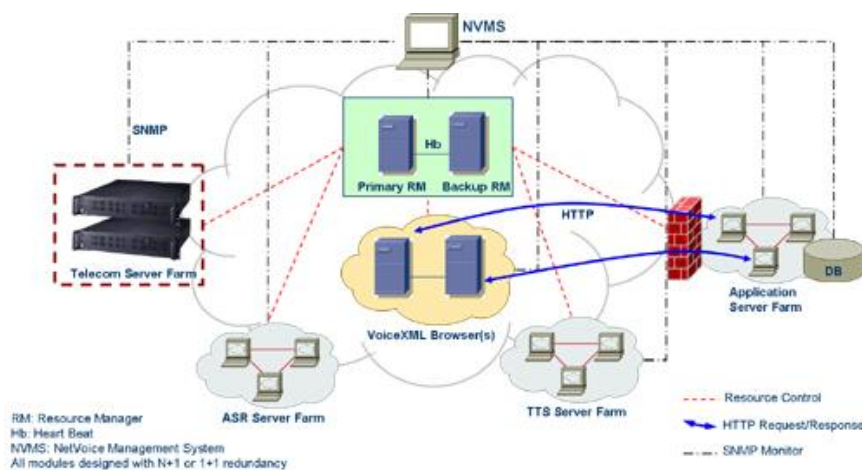
Red telefónica
– Puede ser
PSTN o VoIP

Servidor de aplicaciones
– Documentos VoiceXML
y lógica de negocio



<http://www.phonologies.com/pdfs/whyvoicexml.pdf>

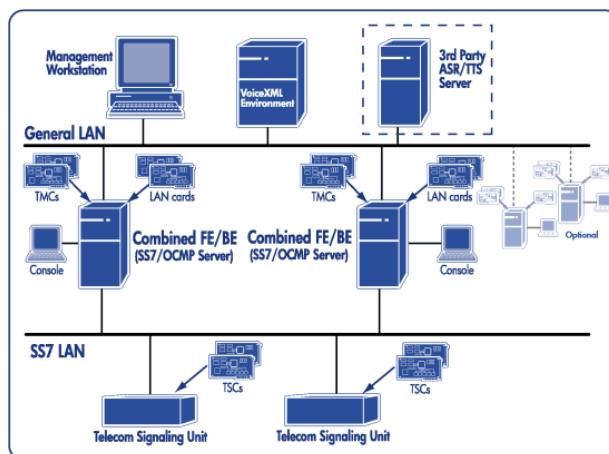
VoiceXML Distribuido – Ejemplo 2



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

VoiceXML Distribuido – Ejemplo 3


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



David Griol – UC3M

95

VoiceXML - Índice

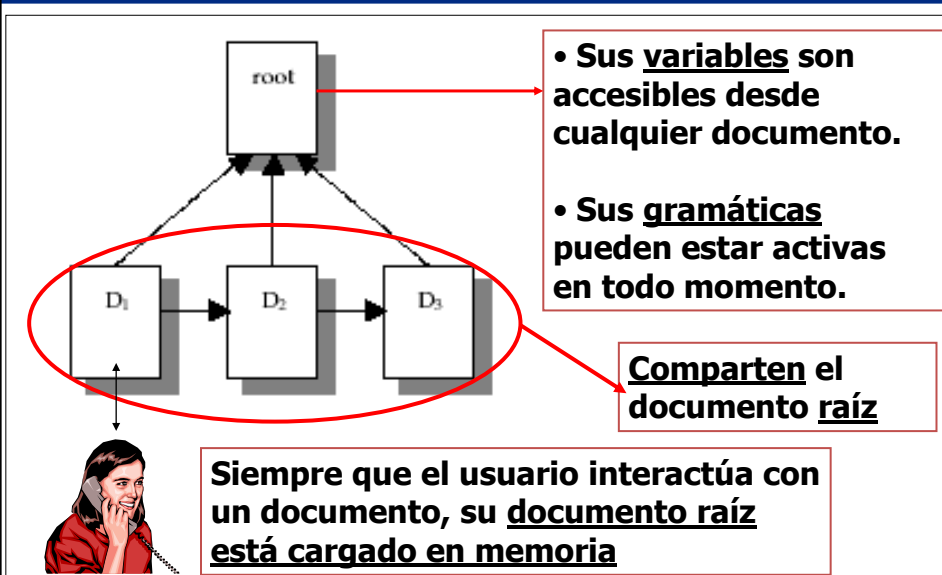
- Introducción
- Arquitectura
- Conceptos 
- Construcción del diálogo
- Control de flujo
- Contenido ejecutable
- Entrada del usuario
- Salida del sistema
- Aspectos avanzados

David Griol – UC3M

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.

Aplicación VXML



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

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: Elementos

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

VOICE XML: Elementos

Elemento	Funcionalidad
<code><filled></code>	Acción a ejecutar cuando se completan los campos
<code><form></code>	Diálogo para presentar información y recoger datos
<code><goto></code>	Ir a otro diálogo en el mismo o diferente documento
<code><grammar></code>	Especifica el reconocimiento de voz o la grammar DTMF
<code><help></code>	Captura un evento ayuda
<code><if></code>	Logica condicional
<code><initial></code>	Declara código inicial antes de entrar en el form
<code><link></code>	Especifica una transición válida para todos los diálogos en su alcance
<code><log></code>	Genera un mensaje de depuración.
<code><menu></code>	Diálogo para seleccionar entre varias alternativas
<code><meta></code>	Define un ítem de metadata ítem en formato nombre / valor
<code><metadata></code>	Define información metadata usando el esquema metadata
<code><noinput></code>	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

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)

VoiceXML - Índice

- Introducción
- Arquitectura
- Conceptos
- Construcción del diálogo
- Control de flujo
- Contenido ejecutable
- Entrada del usuario
- Salida del sistema
- Aspectos avanzados



Construcción del diálogo

- El diálogo se construye en base a cuatro elementos principales de VoiceXML:
 - Formularios
 - Menús
 - Elementos “Filled”
 - Enlaces

Formularios - Introducción

- Los elementos clave de VoiceXML son los formularios (**forms**), que presentan información y obtienen datos de los usuarios mediante campos (fields).
- Sus atributos son: **id** (nombre) y **scope** (ámbito)

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 – Ejemplo

Gramática activa en todo el documento

```
<?xml version="1.0"?>
<vxml version="1.0">
  <form id="datos" scope="document">
    <field name="ciudadDestino">
      <prompt> ¿A qué ciudad desea viajar?
    </prompt>
    <grammar src="ciudad_destino"/>
    </field>
    <filled>
      <prompt> ¿Ha dicho <value
        expr="ciudadDestino"/>? </prompt>
    </filled>
    ...
```

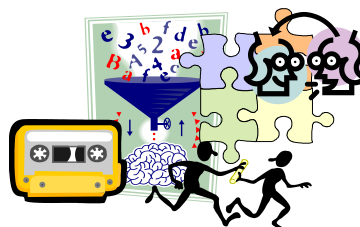
Ejercicio 2

Preguntar fecha de nacimiento

```
<form>
  <block>
    <prompt> _____ </prompt>
  </block>
  <field name = "month">
    <prompt> _____ </prompt>
    <grammar src="http://www.ajax.com/month.grxml"/>
  </field>
  <field name = "day">
    <prompt> _____ </prompt>
    <grammar src="http://www.ajax.com/day.grxml"/>
  </field>
  <field name = "year">
    <prompt> _____ </prompt>
    <grammar src="http://www.ajax.com/year.grxml"/>
  </field>
</form>
```

Formularios - Elementos

- Elementos de un formulario:
 - Entrada
 - Field, record, transfer, object, subdialog
 - Control
 - Block, initial



Formularios - Field

- Elemento **FIELD** (campo)
 - Especifica un elemento de entrada que hay que recoger del usuario.
 - Sus atributos son:
 - *Name* → Debe ser único entre los nombres de elementos del formulario.
 - *Expr* → Valor inicial, por defecto “undefined”.
 - *Cond* → Condición que debe evaluarse a cierta para que el campo sea visitado.
 - Otros: *type*, *slot*, *modal*.

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>
```

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="
numero_telefono$.confidence < 0.6">
```

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

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

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

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)

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

115

David Griol – UC3M

115

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.

David Griol – UC3M

116

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: 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">
  <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>
```

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" 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>
      </filled>
    </field>
  </form>
</vxml>
```

LEAF
leaf.vxml

A
P
L
I
C
A
C
I
O
N

VOICE XML: Estructura y Ejecución

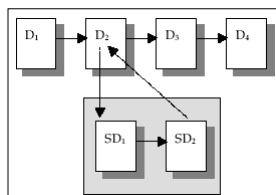
```
<?xml version="1.0" encoding="UTF-8"?>
<vxml xmlns="http://www.w3.org/2001/vxml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/vxml
    http://www.w3.org/TR/voicexml20/vxml.xsd"
  version="2.0" application="app-root.vxml">
  <form id="say_goodbye">
    <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>
```

LEAF
leaf.vxml

**A
P
L
I
C
A
C
I
O
N**

Formularios - Subdiálogos

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



Formularios - Subdiálogos

```

<!-- 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://grammariib/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"/>
      </if>
      <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

Formularios - Subdiálogos

```

<?xml version="1.0"?>
<vxml version="1.0">
  <form id="ajuste_cuenta">
    <var name="numero_cuenta"/>
    <var name="numero_telefono"/>
    <subdialog name="info_cuenta"
src="acct_info.vxml#datos">
      <filled>
        <assign name="numero_cuenta"
expr="info_cuenta.num_cuenta"/>
        <assign name="numero_telefono"
expr="info_cuenta.num_telefono"/>
      </filled>
    </subdialog>
  </form>
</vxml>

```

El subdiálogo está en documento acct_info.vxml

Valores devueltos por el subdiálogo

Documento app.vxml

Formularios - Subdiálogos

```
<?xml version="1.0"?>
<vxml version="1.0">
  <form id="datos">
    <field name="num_cuenta" type="digits">
      <prompt> ¿Cuál es su número de cuenta? </prompt>
    </field>
    <field name="num_telefono" type="phone" modal="true">
      <prompt> ¿Cuál es su número de teléfono? </prompt>
      <filled>
        <!-- Los valores obtenidos en ambos campos se
        proporcionan al diálogo que realiza la llamada mediante el
        elemento "return" -->
        <return namelist="num_cuenta num_telefono"/>
      </filled>
    </field>
  </form>
</vxml>
```

Este formulario es el subdiálogo

Documento
acct_info.vxml

Deshabilita otras
posibles
gramáticas que
puedan estar
activas

Devolución de variables al
documento que realiza la llamada

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`

Form Interpretation Algorithm (3)

```
<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, hasido elegido aleatoriamente para contestar a las preguntas de una encuesta. </prompt>
  </block>
  <field name="p1" type="boolean">
    <prompt> ¿ Está de acuerdo con la postura del gobierno respecto a la guerra en Irak? </prompt>
  </field>
  <field name="p2" type="boolean">
    <prompt> ¿ Cree que realmente existía una amenaza de armas de destrucción masiva en Irak? </prompt>
  </field>
  <block>
    <submit next="miServidor.miDominio.es" namelist="p1 p2"/>
  </block>
  <field name="confirmar_salida" type="boolean">
    <prompt> ¿ Seguro que desea terminar la encuesta? </prompt>
    <filled>
      <if cond="confirmar_salida">
        De acuerdo, adiós.
      <exit/>
      <else/>
        De acuerdo, continuemos por donde nos quedamos.
      </if>
      <clear namelist="confirmar_salida"/>
    </filled>
  </field>
</form>
```

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

Al hacer este clear, `confirmar_salida` puede volver a ser visitado. El FIA vuelve a buscar el siguiente ítem a visitar

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 ...

Formularios de iniciativa dirigida por el sistema

```
<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: Valencia
S: El tiempo en Valencia 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

Formularios de iniciativa mixta (2)

```

<form id="viajar_desde_a">
  <grammar src="http://www.direcciones.ejemplo/gramaticas/de_a.gram"/>
  <block>
    <prompt bargein="false">
      Bienvenido a nuestro sistema automático de información...
    </prompt>
  </block>
  <initial name="prompt_inicial">
    <prompt>¿Desde qué ciudad a qué otra ciudad desea viajar?</prompt>
    <nomatch count="1">
      Por ejemplo, diga desde Granada a Córdoba.
    </nomatch>
    <nomatch count="2">
      Lo siento, sigo sin comprender lo que dice.
      Voy a solicitarle la información por partes.
      <assign name="prompt_inicial" expr="true"/>
      <reprompt/>
    </nomatch>
  </initial>
  <field name="ciudad_origen">
    <grammar src="http://www.direcciones.ejemplo/gramaticas/ciudad.gram"/>
    <prompt>¿Desde qué ciudad desea salir?</prompt>
    ... etc. ...
  </field>
  ... etc. ...
</form>

```

Permite reconocer p. e. "desde Granada a Córdoba"

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 visitada por el FIA

Necesario para que se escuche el siguiente prompt

Permite reconocer p. e. "Granada"

Construcción del diálogo

- El diálogo se construye en base a cuatro elementos principales de VoiceXML:
 - Formularios
 - Menús
 - Elementos “Filled”
 - Enlaces

VOICE XML: Menús

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

Menús

- Presentan opciones para realizar transiciones

```

<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.real-madrid.ejemplo/voice/start.vxml">
    Noticias real madrid </choice>
</menu>

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

```

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

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

Énfasis

David Griol – UC3M

133

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)

David Griol – UC3M

134

Construcción del diálogo

- El diálogo se construye en base a cuatro elementos principales de VoiceXML:
 - Formularios
 - Menús
 - Elementos “Filled”
 - Enlaces

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 (2)

```
<form id="obtener_ciudades_origen_destino">
  <field name="ciudad_origen">
    <grammar src="http://www.gramaticas.ejemplo/voicexml/ciudad.gram">
      <prompt>¿Desde qué ciudad desea salir?</prompt>
    </field>
    <field name="ciudad_destino">
      <grammar src="http://www.gramaticas.ejemplo/voicexml/ciudad.gram">
        <prompt>¿A qué ciudad quiere 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

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>
    </field>
    <filled>
      <if cond="ciudad == 'Loja'">
        <prompt>El servicio a Loja ha sido interrumpido.</prompt>
      </if>
    </filled>
  </field>
</form>
```

<filled> hijo de field

Construcción del diálogo

- El diálogo se construye en base a cuatro elementos principales de VoiceXML:
 - Formularios
 - Menús
 - Elementos "Filled"
 - Enlaces

<LINK>

- Tienen una o más gramáticas asociadas
- Se activan cuando la entrada del usuario es aceptada por alguna gramática
- Ámbitos de un enlace
 - Hijo de <vxml> → gramáticas activas en todo el documento
 - Hijo de <form> → gramáticas activas en el formulario
 - Si están en documento raíz a nivel de documento → activas en cualquier documento de la aplicación
- Permiten
 - 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 Voice XML
</grammar>
<dtmf> 2 </dtmf>
</link>
```

Este enlace se activa al pronunciar las palabra o pulsar el botón del "2" en el teléfono

- generar un evento (como <throw>)

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

Al pronunciar estas palabras se genera el evento "help"

<GOTO>

- Realiza transiciones a:
 - Otro elemento del mismo formulario

```
<goto nextitem="confirmacion_ssn"/>
<goto expritem="(type==12)? 'confirmacion_ssn' : 'reject' "/>
```

- 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"/>
```

<SUBMIT> y <EXIT>

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

VoiceXML - Índice

- Introducción
- Arquitectura
- Conceptos
- Construcción del diálogo
- Control de flujo 
- Contenido ejecutable
- Entrada del usuario
- Salida del sistema
- Aspectos avanzados

Control de flujo

- El control de flujo se realiza mediante **gestión de eventos**.
- Los eventos pueden ser:
 - 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>**)
- La gestión de los eventos se realiza:
 - Se aplica por *herencia de ascendientes*:
 - Si en un campo no se ha definido <catch> pero sí en el formulario que lo contiene, el <catch> de formulario se aplica al campo
 - *Etiquetas*: **<catch>**, **<error>**, **<help>**, **<noinput>**, **<no match>**

```
<throw event="nomatch"/>
<throw
event="telephone.disconnect.hangup"/>
```

Captura de eventos - <CATCH>

- Tratar excepciones o condiciones de error
- Control mechanism for dialog turn retries
 - <catch event="noinput"> ... </catch>
 - <catch event="nomatch"> ... </catch>
 - <catch event="help"> ... </catch>
- Notación abreviada:
 - <noinput> ... </noinput>, etc.
- Alcance según ocurran
 - <form>, <field>, etc.

Captura de eventos - <CATCH>

```
<form id="lanzamiento_misiles">
  <field name="id_usuario" type="digits">
    <prompt>¿Nombre de usuario?</prompt>
  </field>
  <field name="clave">
    <prompt>¿Cuál es la clave?</prompt>
    <grammar>lechuga</grammar>
    <help>Es el nombre de un vegetal.</help>
    <catch event="nomatch noinput" count="3">
      <prompt>Violación 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

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

```
<error>Se ha producido un error – por favor, llame de nuevo más tarde.</error>
```

```
<help>No hay ayuda disponible.</help>
```

```
<noinput>No he escuchado nada, por favor, inténtelo de nuevo.</noinput>
```

```
<nomatch>He oído algo, pero no se trata de una ciudad conocida.</nomatch>
```

Notación abreviada

Otros eventos

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

Captura de eventos

```

<form>
  <prompt> When were you born? </prompt>
  <field name = "month">

      <prompt> What month?</prompt>
      <grammar src="http://www.ajax.com/month.grxml"/>
    </field>
    ....
</form>

```

Captura de eventos

```

<form>
  <prompt> When were you born? </prompt>
  <field name = "month">
    <catch event="noinput">
      ....
    </catch>

    <prompt> What month?</prompt>
    <grammar src="http://www.ajax.com/month.grxml"/>
  </field>
  ....
</form>

```

Captura de eventos

```

<form>
  <prompt> When were you born? </prompt>
  <field name = "month">
    <catch event="noinput">
      .....
    </catch>
    <catch event="nomatch">
      .....
    </catch>
    <prompt> What month?</prompt>
    <grammar src="http://www.ajax.com/month.grxml"/>
  </field>
  .....
</form>

```

Captura de eventos por defecto

```

<catch event = "nomatch">
  <prompt>
    No he entendido, por favor repita
  </prompt>
</catch>

```

```

<catch event = "help">
  <prompt>
    Lo siento, no existe ayuda.
  </prompt>
</catch>

```

```

<catch event = "noinput">
  <prompt>
    No escuchado nada, por favor repita.
  </prompt>
</catch>

```

Ejercicio 3


Añadir eventos para el campo MES

```
<catch event = "nomatch">  
  <prompt>  
    _____  
  </prompt>  
</catch>
```

```
<catch event = "help">  
  <prompt>  
    _____  
  </prompt>  
</catch>
```

```
<catch event = "noinput">  
  <prompt>  
    _____  
  </prompt>  
</catch>
```

VoiceXML - Índice

- Introducción
- Arquitectura
- Conceptos
- Construcción del diálogo
- Control de flujo
- Contenido ejecutable 
- Entrada del usuario
- Salida del sistema
- Aspectos avanzados

Contenido ejecutable

- Nos referimos a un bloque de lógica procedural.
- Puede aparecer en:
 - Bloques
 - Acciones asociadas a filled
 - Capturadores de eventos como `<catch>`

Contenido ejecutable (2)

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

Bloques de código


- La etiqueta **<block>** encierra contenido ejecutable que se ejecuta si la variable asociada al bloque está indefinida y el atributo condición (en caso de existir) se evalúa a "true".
- La variable asociada se establece a "true" inmediatamente una vez se haya entrado al bloque con lo que éstos se suelen ejecutar una única vez por ejecución.

Lógica condicional

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

```
<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>
```

VoiceXML - Índice

- Introducción
- Arquitectura
- Conceptos
- Construcción del diálogo
- Control de flujo
- Contenido ejecutable
- Entrada del usuario 
- Salida del sistema
- Aspectos avanzados

David Griol – UC3M

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.

David Griol – UC3M

158

Gramáticas

- El elemento **<grammar>** se emplea para indicar una gramática que:
 - Especifique un conjunto de frases válidas que el usuario puede decir para que el sistema ejecute una acción o le de información.
 - Cuando la expresión del usuario concuerde con una de las frases válidas, devuelva la interpretación semántica correspondiente. Ésta puede ser un valor simple, un conjunto de pares atributo-valor o un objeto con distintos campos.

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>
```

GRAMMAR
SRGS - XML

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

Tipos de gramáticas

- Dos tipos: Interna y externa

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

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

  <initial>
    <prompt> ¿En qué puedo ayudarle? </prompt>
  </initial>

  <!-- obtencion ciudad destino -->
  <field name="ciudadDestino">
    <prompt>¿A qué ciudad quiere viajar?</prompt>
    <grammar src="ciudades.jsgf"/>
  </field>
  ...
</form>
```

Gramática interna

Gramática externa a nivel
de formulario

Gramática externa a
nivel de campo

Ámbitos de Gramáticas

- **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
- **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 cuando usuario está activa en cualquier diálogo del documento) **scope="document"**
 - Si **scope="document"** y el documento es el raíz de la aplicación, está activa cuando usuario está en cualquier diálogo de cualquier documento de la aplicación
- **Gramática de menú:** por defecto, tiene como ámbito **dialog**. Sólo está activa cuando usuario está en menú

Ámbitos de Gramáticas (2)

Gramática a nivel de formulario

```
#JSGF V1.0;
grammar viajar_de_a;

public <viajar_de_a> =
    [<deseo>]
    [<viajar> <ciudad>]
    {this.ciudadDestino=$ciudad} ]
    [<procedencia> <ciudad>]
    {this.ciudadOrigen=$ciudad} ];

<deseo> = quiero | me gustaría |
[ yo ] quería | [ yo ] necesito | [ yo ] tengo que;

<viajar> = [ir | viajar ] a;

<ciudad> = jaén | córdoba | sevilla |
huelva |cádiz | Málaga | granada |
almería;

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

Si en la frase aparece "<viajar> <ciudad>" esa ciudad se asigna al campo ciudadDestino


Si en la frase aparece "<procedencia> <ciudad>" esa ciudad se asigna al campo ciudadOrigen

Gramática a nivel de campo

```
#JSGF V1.0;
grammar ciudades;

public <ciudades> = jaén | córdoba | sevilla |
huelva |cádiz | Málaga | granada |almería;
```

VoiceXML - Índice

- Introducción
- Arquitectura
- Conceptos
- Construcción del diálogo
- Control de flujo
- Contenido ejecutable
- Entrada del usuario
- Salida del sistema 
- Aspectos avanzados

David Griol – UC3M

VOICE XML: Salida del Sistema

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

David Griol – UC3M

166

Salida del sistema

- El elemento `<prompt>` controla la salida de voz sintetizada y de audio pregrabado.
- Se encolan para ser reproducidos hasta que se necesite una entrada del usuario. En ese caso, el sistema espera a la entrada y una vez recibida, se sigue con el proceso de interpretación.
- Sus atributos son:
 - `bargein` → Controla si el usuario puede interrumpir la salida del sistema.
 - `bargeintype` → Establece el bargein a "speech" o "hotword".
 - `cond` → Expresión que debe evaluarse a "true" para que se reproduzca el prompt.
 - `count` → Número que permite emitir distintos prompts si el usuario hace algo de forma repetitiva.
 - Otros: `timeout`, `xml:lang`, `xml:base`

Generación de prompts

```

<nomatch count="1">
  Para abrir la puerta diga claramente su clave.
</nomatch>
<nomatch count="2">
  <prompt> Este es su <emp> último </emp> intento.
</prompt>
</nomatch>
<nomatch count="3">
  Entrada denegada.
  <exit/>
<help>
  <prompt> Está usted llamando al número <value expr="num_telefono" class="phone"/></prompt>
  <prompt> Está usted llamando al número <sayas class="phone">312-555-1212</sayas> </prompt>
</help>
<block>
  <prompt> <audio src="bienvenida.wav"><emp> Bienvenido </emp> a este portal de voz.</audio>
</prompt>
</block>
<prompt bargein="false"><audio src="aviso_legal.wav"/></prompt>

```

Énfasis

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

<prompt> ... </prompt> necesarios si mensaje contiene etiquetas

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

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

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

Generación de prompts (2)

```

<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, dí sí. Para salir, dí no.
    </prompt>
    <filled>
      <if cond="otro">
        <goto next="#seleccionar_chiste"/>
      </if>
    </filled>
  </field>
</form>

<prompt count="1">Elija un color para su nuevo Modelo T.</prompt>
<prompt count="2" timeout="120s">
  Por favor, elija el color de su nuevo Modelo T 19 24.
  Algunos posibles colores son los siguientes: negro, negro o negro. Por favor, elija con
  tranquilidad.
</prompt>

```

Obtención número aleatorio

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

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

El usuario tiene 120 s para responder

David Griol – UC3M

169

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>¿Quiere helado de postre?</prompt>
  <prompt count="2">
    Si quiere helado, diga sí. Si no quiere, diga no.
  </prompt>
  <noinput>
    No he oído nada.
    <reprompt/>
  </noinput>

```

Logra que se genere el prompt en la siguiente interacción del FIA

S: ¿Quiere helado de postre?
 U: *(silencio)*
 S: No he oído nada.
 U: Si quiere helado, diga sí. Si no quiere, diga no.
 S: *(silencio)*
 U: No he oído nada.
 S: Si quiere helado, diga sí. Si no quiere, diga no.
 U: No

Usando reprompt

S: ¿Quiere helado de postre?
 U: *(silencio)*
 S: No he oído nada.
 S: No he oído nada.
 U: No

Sin usar reprompt

David Griol – UC3M


170

Etiquetas para la síntesis de voz

- SSML (Speech Synthesis Markup Language Specification) es el estándar diseñado para enriquecer los lenguajes XML como VoiceXML en la síntesis de voz. Versión 2.0 en W3C en 2003.
- El papel principal de este lenguaje de marcas es proveer un estándar para controlar aspectos del discurso como pronunciación, volumen, tono, prosodia, velocidad, etc.

```
<prompt> Bienvenido a nuestra tienda de acordeones.  
<audio src="http://www.acordeones_mil.com/acordeon.wav"/>  
Tenemos acordeones desde <say-as interpret-as="currency">$299.95</say-as>.  
</prompt>
```

VoiceXML - Índice

- Introducción
- Arquitectura
- Conceptos
- Construcción del diálogo
- Control de flujo
- Contenido ejecutable
- Entrada del usuario
- Salida del sistema
- Aspectos avanzados 

Búsqueda de recursos

- Acceso a recursos en una URI gobernado por tres atributos
- **caching**
 - “**safe**”: acceder a versión más reciente
 - “**fast**”: usar versión en caché del recurso
- **fetchtimeout** → intervalo de tiempo a esperar llegada del recurso antes de generar evento **error.badfetch**
- **fetchhint** → especifica cuándo el entorno del intérprete debe obtener un recurso del servidor
 - “**prefetch**”: descargar fichero cuando se carga página
 - “**safe**”: descargar fichero cuando es realmente necesario
 - “**stream**”: usado para ficheros grandes. Comenzar a procesar fichero conforme va llegando, sin esperar a su llegada completa

```
<property name="caching" value="fast"/>
<form id="test">
  <block>
    <!-- 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 elementos de este documento
usarán por defecto caching="fast"

Grabación de mensajes

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

```
<?xml version="1.0"?>
<vxml version="1.0">
  <form>
    <record name="saludo" beep="true" maxtime="10s" finalsilence="4000ms" dtmfterm="true"
      type="audio/wav">
      <prompt> Diga su mensaje tras escuchar el tono.</prompt>
      <noinput> No he oído nada, inténtelo de nuevo.</noinput>
    </record>
    <field name="confirmacion" type="boolean">
      <prompt> Su mensaje es <value expr="saludo"/>.</prompt>
      <prompt> Para mantenerlo, diga sí. Para descartarlo, diga no.</prompt>
      <filled>
        <if cond="confirmacion">
          <submit next="guardar_saludo.pl" method="post" namelist="saludo"/>
        </if>
        <clear/>
      </filled>
    </field>
  </form>
</vxml>
```

Se emite un pitido de comienzo de
grabación

Al pulsar botón del
teléfono se detiene
grabación mensaje

El mensaje se envía a un servidor

Transferencia de llamadas

- **<transfer>** se usa para transferir la llamada a otro nº de teléfono. Dos tipos de transferencia:

- **bridging**: la llamada inicial se reanuda tras transferencia (bridge="true")
- **blind transfer**: la llamada inicial NO se reanuda tras transferencia (bridge="false")

```
<form name="transferencia">
  <var name="duracion" expr="0"/>
  <block>
    <audio src="chopin12.wav">
  </block>
  <transfer name="mi_llamada" dest="phone://18005551234" connecttimeout="30s" bridge="true">
    <filled>
      <assign name="duracion" expr="mi_llamada$.duration"/>
      <if cond="mi_llamada == 'busy'">
        <prompt> Lo sentimos, nuestros operadores están atendiendo otras llamadas en estos momentos. Por favor, inténtelo más tarde.</prompt>
      <elseif cond="mi_llamada == 'noanswer'">
        <prompt> Lo sentimos, el horario de nuestros operadores es de 9 de la mañana a 7 de la tarde, de lunes a sábado.</prompt>
      </if>
    </filled>
  </transfer>
  <block>
    <submit namelist="mi_llamada duracion" next="/cgi-bin/report"/>
  </block>
</form>
```

Duración de la llamada

Resultado de la transferencia (almacenado en nombre de campo):

"busy"	- Destino rechaza llamada.
"noanswer"	- No ha habido respuesta.
"network_busy"	- Red intermedia rechaza llamada.
"near_end_disconnect"	- Llamada completada y finalizada por origen.
"far_end_disconnect"	- Llamada completada y finalizada por destino.
"network_disconnect"	- Llamada completada y finalizada por red.

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

```
<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 éste <emp>sí</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

SCRIPTS

- **<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

```
<?xml version="1.0"?>
<vxml version="1.0"?>
  <script> <![CDATA[ function factorial(n) { return (n <= 1)? 1 : n * factorial(n-1); }
]]> </script>
  <form id="form">
    <field name="numero" type="number">
      <prompt>Diga el número cuyo factorial desea conocer.</prompt>
      <filled>
        <prompt>
          ...
          El factorial de <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>
```

Ejecución en múltiples documentos

```
<?xml version="1.0"?>
<vxml version="1.0">
  <var name="despedida" expr="adiós"/>
  <link next="operador_xfer.vxml"> <grammar> operador </grammar>
</link>
</vxml>
```

Documento raíz:
app-root.vxml

```
<?xml version="1.0"?>
<vxml version="1.0" 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>
```

El documento hoja
especifica URI de
documento raíz

Documento hoja:
main.vxml

Navegación dentro del mismo documento

- Cuestiones a aprender**

- Elemento *bloque*
- Elemento de acción *goto*

- Etiquetas**

<block>

<goto>

<goto ..> tiene atributo que especifica donde continuar la ejecución

<block ..> tiene atributo que especifica el nombre del bloque

```
<?xml version="1.0" encoding="UTF-8" ?>
<callxml version="2.0">
  <block>
    <playaudio value="helloworld.wav"
      termdigits="*#"/>
    <ontermdigit value="*">
      <goto value="#one"/>
    </ontermdigit>
    <ontermdigit value="#">
      <goto value="#two"/>
    </ontermdigit>
  </block>

  <block label="one">
    <text>you pressed the star key.</text>
  </block>

  <block label="two">
    <text>you pressed the pound key.</text>
  </block>
</callxml>
```

Navegación externa

- Cosas a aprender**

- enlazar documentos mediante etiqueta <goto>

<goto ..> especifica otro documento, que incluso puede estar en otro servidor

<submit ..> especifica qué variables se pasarán al otro documento (* = todas)

<method ..> especifica método de paso de las variables (get o post)

```
<?xml version="1.0" encoding="UTF-8" ?>
<callxml>
  <block>
    <playaudio value="helloworld.wav"
      termdigits="*#"/>
    <ontermdigit value="*">
      <goto value="helloworld-star.xml"
        submit="*"
        method="get"/>
    </ontermdigit>
    <ontermdigit value="#">
      <goto value="helloworld-pound.xml"
        submit="*"
        method="get"/>
    </ontermdigit>
  </block>
</callxml>
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<callxml version="2.0">

  <block label="star">
    <text>you pressed the star key.</text>
  </block>

</callxml>
```

Fichero helloworld-star.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<callxml version="2.0">

  <block label="pound">
    <text>you pressed the pound key.</text>
  </block>

</callxml>
```

Fichero helloworld-pound.xml

Contestador automático

- **Cuestiones a aprender**
 - Grabar un mensaje mediante el elemento de acción **<recordaudio>**
 - Guardar el fichero de audio como un anexo de un e-mail
- **Atributos de la etiqueta**

<format>

<value>

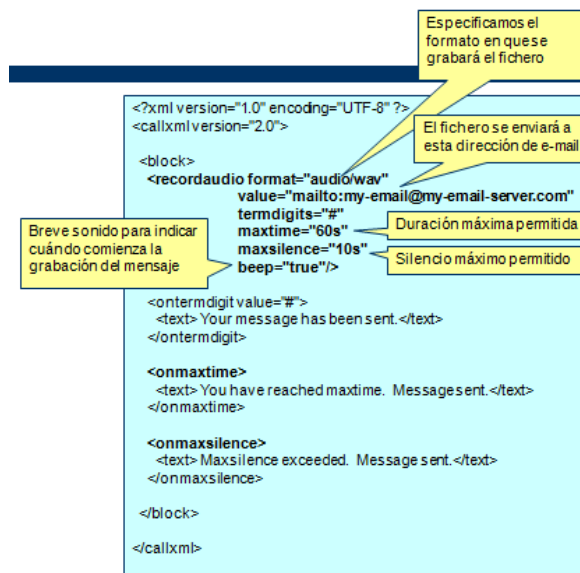
<termdigits>

<beep>

<maxtime>

<maxsilence>

Contestador automático



ID Llamante e ID Llamado

- **Cuestiones a aprender**
 - ID Llamante = nº de teléfono de quien realiza la llamada
 - ID Llamado = nº de teléfono de la aplicación llamada
 - Ambas son variables de sesión de CallXML; se pasan automáticamente en las llamadas mediante <goto> y <run>
 - Ambas son variables de sólo lectura

- **Etiqueta**

<playnumber>

<playnumber...> es muy parecida a <text> pero se usa para TTS de un número en lugar de un texto; mediante **format** permite especificar cómo se lee el dato (p. e. **format="digits"** o **format="number"**)

Pausa de 3 segundos

```
<?xml version="1.0" encoding="UTF-8" ?>
<callxml version="2.0">

  <block repeat="3">
    <text>Hello there. The caller i d number is:</text>
    <playnumber value="$session.callerid;" format="digits"/>
  </block>

  <text>The called i d of this application is:</text>
  <playnumber value="$session.calledid;" format="number"/>

  <wait value="3s"/>
</block>
</callxml>
```

p. e. 12 3 4 5

p. e. 12,345

Control de la caché

- **Cuestiones a aprender**
 - Control de la caché desde CallXML
 - Control de la caché en el lado del servidor
- **Memoria caché** (ubicada en nuestro sistema)
 - Recursos de Internet accedidos se mantienen en memoria RAM de nuestro ordenador
 - Ventaja: se evitar accesos a Internet innecesarios
 - Inconveniente: quizás los datos en caché no estén actualizados

Control de la caché (2)

- Control de la caché desde CalXML

- Cualquier elemento CalXML que cargue una URL o un fichero de audio acepta un atributo llamado **cache** que acepta dos posibles valores YES y NO, siendo el valor por defecto YES. Esto incluye a los elementos:

- <playaudio>
- <inputaudio>
- <inputdigits>
- <goto>
- <run>

helloworld.wav y greeting.wav se cargan desde el servidor cada vez que se ejecuta la aplicación (quizás cambien con frecuencia ...)

mainmenu.wav se carga desde la caché

mynextpage.xml se carga desde el servidor

```
<playaudio value="helloworld.wav" format="audio/wav" cache="NO"/>
<playaudio value="greeting.wav" format="audio/wav" cache="NO"/>
<playaudio value="mainmenu.wav" format="audio/wav"/>
<goto value="mynextpage.xml" submit="" method="get" cache="NO"/>
```

Control de la caché (3)

- Control de la caché en el lado del servidor

- Control estándar de caché de páginas web: cabecera HTTP al principio del código HTML: **"Cache-Control: no-cache"**
- ¿Cómo alterar dicha cabecera? Hay varios métodos:

- Suponiendo que el servidor use HTTP 1.1 :

```
<?PHP
header("Cache-Control: no-cache");
... body of code here...
?>
```

En PHP

```
<%
response.Expires = -1
... body of code here...
%>
```

En ASP

```
print "Cache-Control: no-cache";
... body of code here...
```

En Perl

```
<CFHEADER NAME="Cache-Control" VALUE="no-cache">
... body of code here...
```

En ColdFusion

- Suponiendo que el servidor use HTTP 1.0 :

```
<?PHP
header("Pragma: no-cache");
... body of code here...
?>
```

En PHP

Control de la caché (4)

Especifica que NO se use la caché (en contra de lo indicado en el elemento goto)

```
<?xml version="1.0" encoding="UTF-8" ?>
<callxml version="2.0">

  <block>
    <playAudio value="helloworld.wav"
      termdigits="#"
      cache="no"/>

    <ontermdigit value="#">
      <goto value="helloworld-handler.php?key=star"
        submit="#"
        method="get"
        cache="yes"/>
    </ontermdigit>

    <ontermdigit value="#">
      <goto value="helloworld-handler.php?key=pound"
        submit="#"
        method="get"
        cache="yes"/>
    </ontermdigit>
  </block>
</callxml>
```

Se indica usar helloworld-handler.php existente en la caché

```
<?PHP
header("Cache-Control: no-cache");

$$ = $HTTP_GET_VARS["key"];

echo "<?xml version='1.0' encoding='UTF-8' ?> ";
echo "<callxml> ";
echo "  <block label=\"$s\"> ";

if ($$ == "star") {
  echo "    <text>you pressed the star key.</text> ";
} else {
  echo "    <text>you pressed the pound key.</text> ";
}

echo "  </block> ";
echo "</callxml> ";
?>
```

helloworld-handler.php

David Griol – UC3M

187

Puntos a tratar

- Estándares
 - VoiceXML
 - CCXML
 - CallXML
- Herramientas de desarrollo
 - CSLU Toolkit
 - IBM WebSphere Voice Toolkit
 - Voxeo Designer
 - Otras herramientas

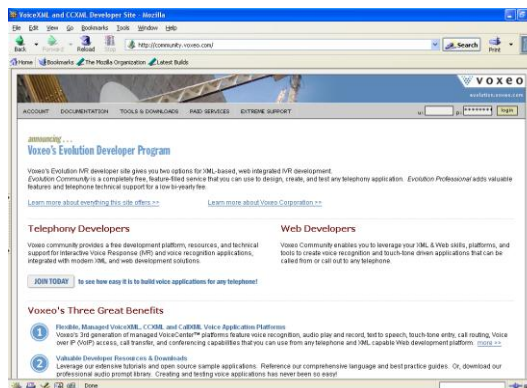
David Griol – UC3M

188

Voxeo Community

- Sitio web que proporciona recursos para construir y comprobar funcionamiento de sistemas de diálogo basados en VoiceXML, CCXML o CalXML:

- Tutoriales
- Ficheros de audio pregrabados
- Gramáticas VoiceXML y ejemplos de aplicaciones
- Herramienta gráfica de diseño
- Depurador de aplicaciones, etc.



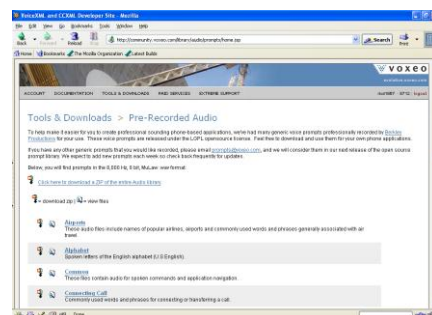
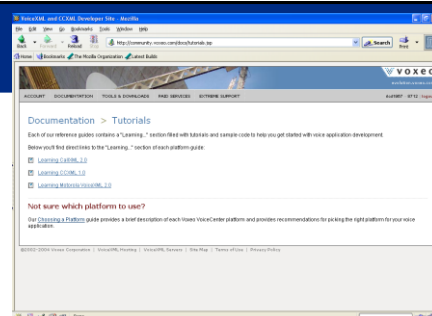
<http://community.voxeo.com>

David Griol – UC3M

189

Voxeo Community

- **Tutoriales**
 - VoiceXML 2.0
 - CCXML 1.0
 - CalXML 2.0
- **Ficheros de audio pregrabados**
 - Nombres de compañías aéreas, aeropuertos, etc.
 - Nombre de letras (en Inglés)
 - Nombres de tarjetas de crédito
 - Nombre de meses, días de la semana
 - Etc.

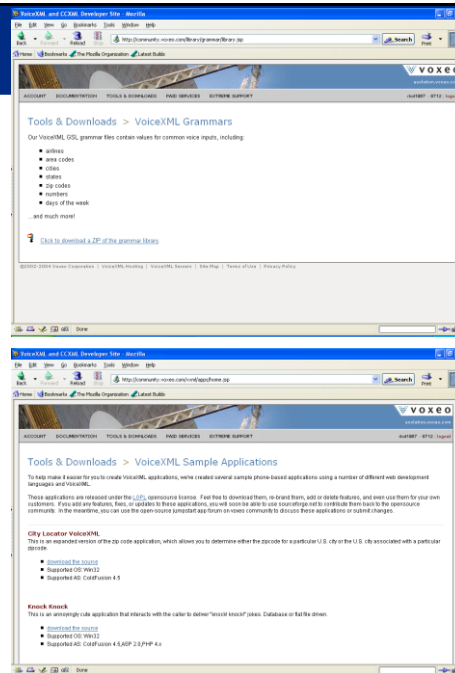


David Griol – UC3M

190

Voxeo Community

- **Gramáticas VoiceXML**
 - Compañías aéreas
 - Prefijos telefónicos
 - Ciudades
 - Códigos postales
 - Números
 - Días de la semana
 - Etc.
- **Aplicaciones VoiceXML**
 - City locator
 - VXML transfer
 - Etc.

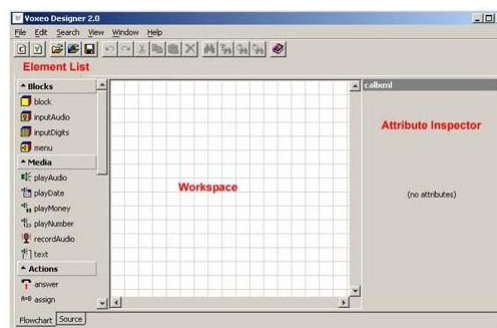


David Griol – UC3M

191

Voxeo Designer 2.0

- Herramienta gráfica para diseño de aplicaciones basadas en VoiceXML y CallXML
- Tres partes:
 - **Lista de elementos:** contiene elementos de acción, de eventos y de bloques
 - **Área de trabajo:** permite visualizar gráficamente las aplicaciones que desarrollemos
 - **Inspector de atributos:** permite ver y editar los atributos de los elementos seleccionados en el área de trabajo

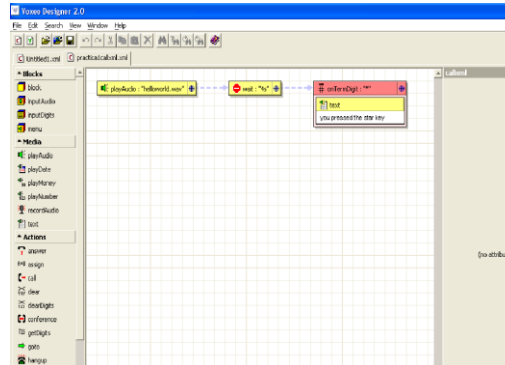


David Griol – UC3M

192

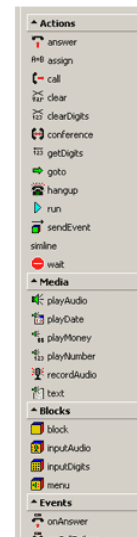
Voxeo Designer 2.0

- Para añadir elementos al espacio de trabajo
 - Hacer clic en el elemento
 - En algunos casos se insertan elementos dentro de otros elementos




Voxeo Designer 2.0

- Lista de elementos
 - **Acciones**
 - Constituyen los bloques principales de la aplicación
 - Alteran flujo de la aplicación
 - Usados para responder y generar eventos
 - **Medios de comunicación** (sólo para CallXML)
 - Usados para proporcionar y obtener información de los usuarios (restringidos a audio y texto)
 - **Bloques**
 - Contienen otros bloques y controlan flujo de aplicación
 - **Eventos** (sólo CallXML)
 - Elementos usados para responder a eventos
 - Son aplicables sólo a determinados Medios de comunicación y acciones



Puntos a tratar

- Estándares
 - VoiceXML
 - CCXML
 - CallXML
- Herramientas de desarrollo
 - CSLU Toolkit
 - IBM WebSphere Voice Toolkit
 - Voxeo Designer
 - Otras herramientas 

Otras herramientas

- **VOXEO COMMUNITY** (<http://community.voxeo.com>)
- **HeyAnita RapidApp** (<http://freespeech.heyanita.com>)
- **Tellme Debug Log** (<http://studio.tellme.com>)
- **Voice Genie Call Log Explorer** (<http://developer.voicegenie.com>)
- **BeVocal Vocal Scripter** (<http://cafe.bevocal.com>)
- **Apple** (<http://www.apple.com/macOS/speech>)
- **AT&T** (<http://www.att.com/aspg>)
- **BBN** (<http://www.bbn.com/departments/dept-slp.html>)
- **IBM** (<http://ibm.com/software/speech>)
- **Lernout & Hauspie** (<http://www.lhs.com>)
- **Microsoft** (<http://www.microsoft.com/speech>)
- **Nuance** (<http://www.nuance.com>)
- **Philips** (<http://www.speech.be.philips.com>)
- **Speechworks** (<http://www.speechworks.com>)

Aplicación práctica: Centralita Telefónica

#JSGF V1.0;

grammar centralita;

public <centralita> = "por favor" <peticion> | <peticion> ["por favor"];

<nombremasc> = pepe | juan | josé | luis | victor | 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 | jimenez | 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 | vizcaino | 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>;

GRAMÁTICA PRINCIPAL

David Griol – UC3M

197

Aplicación práctica: Centralita Telefónica

#JSGF V1.0;

grammar confirma;

public <confirma> = <si> | <no>;

<si> = si | vale | "es correcto" | bien | correcto;

<no> = no | mal | "está mal" | incorrecto;

GRAMÁTICA AFIRMACIONES Y NEGACIONES

<?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>

PROGRAMA PRINCIPAL

David Griol – UC3M

198

Aplicación práctica: Centralita Telefónica

```
<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>
```

**PROGRAMA
PRINCIPAL**

ÍNDICE

- 1.- Introduccción
- 2.- Aplicaciones
- 3.- Clasificación
- 4.- Antecedentes Históricos
- 5.- Arquitectura
- 6.- Tecnologías involucradas
- 7.- Retos actuales
- 8.- Estándar VoiceXML
- 9.- Conclusiones
- Referencias

Conclusiones

- **Aplicación muy importante de las tecnologías del habla**
 - tecnologías utilizadas: reconocimiento del habla, procesamiento del lenguaje natural, metodologías estadísticas, estándares, fuentes de información, campos de investigación...
- **Implementar un sistema de diálogo requiere**
 - reconocer palabras pronunciadas por usuarios
 - interpretarlas dentro de un contexto
 - proporcionar respuestas coherentes dentro del contexto del diálogo
- **Tarea muy difícil**
 - Estos sistemas no tienen conocimiento del mundo ni herencia cultural
 - señal de voz puede estar corrompida por varias causas
 - ruido de fondo, peculiaridades de micrófonos, etc.
 - diferentes características del tracto vocal de usuarios
 - diferentes pronunciaciones de palabras por usuarios
 - frases incorrectas gramaticalmente y/o ambiguas
 - efectos coarticulatorios

Conclusiones

- **Evaluar prototipo antes de implantación en mundo real**
 - Medidas de evaluación de dos tipos: subjetivas y objetivas
 - **subjetivas**: suelen usarse para evaluar sistema completo
 - Valoración de usuarios de test
 - naturalidad del sistema (*naturalness*)
 - satisfacción general de usuarios
 - facilidad de logro de tareas
 - calidad de mensajes generados, etc.
 - **objetivas**: suelen usarse para evaluar cada componente por separado
 - Medidas estadísticas
 - WA (*Word Accuracy*), SR (*Sentence Recognition*)
 - SU (*Sentence Understanding*), IR (*Implicit Recovery*)

ÍNDICE

- 1.- Introduccción
- 2.- Aplicaciones
- 3.- Clasificación
- 4.- Antecedentes Históricos
- 5.- Arquitectura
- 6.- Tecnologías involucradas
- 7.- Estándar VOICEXML
- 8.- Conclusiones
- 9.- Referencias

David Griol – UC3M

Referencias

- Billi, R., Castagneri, G., Danielli, M. 1997. Field trial evaluations of two different information inquiry systems. *Speech Communication*
- Callejas Z.; R. Lopez-Cozar. 2008. Relations between de-facto criteria in the evaluation of a spoken dialogue system. *Speech Communication*, 50(8–9):646–665.
- Danielli, M. 1996. On the use of expectations for detecting and repairing human-machine miscommunication. Working notes of the AAAI-96. 87-93
- Ferguson, G. M., Allen, J. F., Miller, B. W., Ringger, E. K. 1996. The Design and Implementation of the TRAINS-96 system: A Prototype-Mixed Planning Assistant. TRAINS Technical Note 96-5, Computer Science Dept., Universidad de Rochester
- Griol, D.; L.F. Hurtado, E. Segarra, and E. Sanchis. 2008. A Statistical Approach to Spoken Dialog Systems Design and Evaluation. *Speech Communication*, 50(8–9):666–682.
- López-Cózar, R., García, P., Díaz, J., Rubio, A. J. 1997. A Voice Activated Dialog System for Fast-Food Restaurant Applications. Eurospeech, pág. 1783-1786
- López-Cózar, R., Rubio, A. J., García, P., Segura, J. C. 1998. A Spoken Dialogue System Based on a Dialogue Corpus Analysis. First International Conference on Language Resources and Evaluation, pág. 55-58
- López-Cózar, R., Rubio, A. J., García, P., Díaz-Verdejo, J. E., López-Soler, J. M. 2000. Sistema Telefónico de Atención A Viajeros. Actas I Jornadas en Tecnología del Habla. CD-ROM ISBN 84-95118-58-0

David Griol – UC3M

204

Referencias

- López-Cózar, R., Rubio, A. J., García, P., Segura, J. C. 1999. Uso de Valores de Confianza y Expectativas en el Sistema de Diálogo SAPLEN. *Procesamiento de Lenguaje Natural*, nº 24, 37-41
- López Soto, M. T., Quesada, J. F., Álvarez, J. 1997. Aplicación de LEKTA al entorno ATOS. *Revista SEPLN* nº 21
- Polifroni, J., Seneff, S., Glass, J., Hazen, T. J. 1998. Evaluating Methodology for a Telephone-Based Conversational System, First International Conference on Language Resources and Evaluation, Granada, pág. 43-49
- Rubio, A. J., García, P., De la Torre, A., Segura, J. C., Díaz-Verdejo, J., Benítez, M. C., Sánchez, V., Peinado, A. M., López-Soler, J. M., Pérez-Córdoba, J. L. 1997. STACC: an automatic service for information access using continuous speech recognition through telephone line. *Eurospeech '97*, vol. 4, pág. 1779-1782
- Schatzmann, J., K. Weilhammer, M. Stuttle, and S. Young. 2006. A Survey of Statistical User Simulation Techniques for Reinforcement-Learning of Dialogue Management Strategies. In *Knowledge Engineering Review*, volume 21(2), pages 97-126.
- Scheffler, K.; and S. Young. 2001. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proc. of HLT'02*, pages 12-18, San Diego (USA).
- Seto, S., Kanazawa, H., Shinci, H., Takebayashi, Y. 1994. Spontaneous Speech dialogue system TOSBURG II and its evaluation. *Speech Communication* 15, pág. 341-353

Referencias

- Torres, F.; L.F. Hurtado, F. García, E. Sanchis, and E. Segarra. 2005. Error handling in a stochastic dialog system through confidence measures. In *SpeechCommunication*, pages (45):211-229.
- Yamada, M., Itoh, F., Sakai, K., Komori, Y., Ohora, Y., Fujita, M. 1994. A spoken dialogue system with active/non-active word control for CD-ROM information retrieval. *Speech Communication* 15, pág. 355-365
- Zue, V., Seneff, S., Polifroni, J., Phillips, M., Pao, C., Goodine, D., Goddeau, D., Glass, J. 1994. PEGASUS: A spoken dialogue Interface for on-line air travel planning. *Speech Communication* 15, pág. 331-340