PROPUESTA DE UNA GUÍA PARA EL MODELADO Y DESARROLLO DE APLICACIONES WEB MULTIMODALES BASADA EN WEB SEMÁNTICA

CARLOS JAVIER FERRIOL DORTICÓS
MARTÍN EMILIO MONROY RÍOS

Tesis

Requisito para la obtención del título de Maestría en Ciencias Computacionales

Director
Eduardo Carrillo Zambrano
Doctor en Tecnología de la Información, Computación y Comunicaciones

UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR
MAESTRÍA EN CIENCIAS COMPUTACIONALES
CARTAGENA
2006

AGRADECIMIENTOS

El presente trabajo fue realizado gracias a la colaboración de varias personas que nos brindaron su apoyo en forma directa e indirecta, entre ellos podemos nombrar a los profesores del programa de maestría en Ciencias Computacionales del Instituto Tecnológico de Estudios Superiores de Monterrey (ITESM), a los integrantes del Grupo de Investigación de la Universidad de San Buenaventura Cartagena (GIDES), al personal administrativo de la Universidad Tecnológica de Bolivar encargado de coordinar el programa de la maestría; a todos ellos manifestamos nuestro más sincero agradecimiento.

De manera especial agradecemos a:

El Dr. Eduardo Carrillo Zambrano, quien desde el comienzo nos brindó su apoyo incondicional, y durante el desarrollo de la tesis nos guió en calidad de director, permitiéndonos llegar a los resultados que presentamos en este documento, sin que las limitaciones de tiempo y espacio se convirtieran en obstáculo.

Al Coordinador Académico Maestría en Ciencias Computacionales, el Dr. Daniel Arenas Seley, por que siempre estuvo dispuesto a colaborarnos.

A Edwin Ramos, estudiante del programa de Sistemas del la Universidad San Buenaventura, por su colaboración en el manejo de los aspectos de diseño gráfico del portal.

RESUMEN

En este trabajo se propone una guía para el desarrollo de aplicaciones web que permitan interacción multimodal, y presenten un modelo de datos basado en web semántica. Para lograrlo se realizó un estudio del estado del arte de los lenguajes de marcado para el desarrollo de aplicaciones multimodales, y un estudio aplicado de las capas ontológica y lógica de la web semántica, además se probó la guía a través de la construcción de un prototipo de portal de información sobre el problema de desplazamiento forzado en la ciudad de Cartagena. Dicha guía no pretende convertirse en una metodología de desarrollo de software, sino más bien complementar las metodologías actuales, específicamente en los aspectos relacionados con la interacción multimodal y el modelado semántico. Para la construcción de la guía se tuvo en cuenta el patrón de desarrollo de software denominado: Modelo Vista Controlador, teniendo en cuenta que su principal finalidad consiste en separar las capas de presentación, lógica y datos.

Descriptores del trabajo: Interacción Multimodal, Web Semántica, Aplicaciones Web, Lenguajes de Marcado, Ontología, Marco de Descripción de Recursos, Arquitectura de software.

CONTENIDO

Pág.

LISTADO DE FIGURAS

Pág.

1. INTRODUCCIÓN

1.1. ANTECEDENTES

El presente proyecto se origina, en primer lugar a partir de una propuesta hecha por el director, el Dr. Eduardo Carrillo, quien ha trabajado como parte de su tesis doctoral conceptos sobre multimodalidad, arquitectura de desarrollo web, específicamente para dispositivos móviles. Y en segunda instancia, a partir de un proceso de revisión bibliográfica realizado por parte de los investigadores, que les permitió establecer el estado del arte sobre multimodalidad y web semántica.

El interés común por complementar conceptos, ha llevado a establecer la necesidad de generar como propuesta una guía para el modelado y desarrollo de aplicaciones web multimodales basada en web semántica; con el ánimo de hacer un aporte, intercambiar ideas, y desarrollar una aplicación, que muestre un resultado esperado.

Adicionalmente, se pretende brindar un apoyo el Grupo de Investigación de la Universidad de San Buenaventura Cartagena (GIDES), al desarrollar un prototipo de portal de información sobre el problema de desplazamiento forzado en la ciudad de Cartagena que incluya modelado semántico de los datos hasta la capa de reglas de la web semántica. Teniendo en cuenta que actualmente éste grupo, reconocido por Colciencias, se encuentra desarrollando el proyecto: “Prueba Piloto de un observatorio sobre Desarrollo, Calidad de vida y Cooperación Internacional”

1.2. OBJETIVOS

General

Proponer una guía para el modelado y desarrollo de aplicaciones web multimodales que incluyan modelado de datos basado en web semántica.

Específicos

Documentar la realización de un estudio del estado del arte de los lenguajes de marcado para el desarrollo de aplicaciones multimodales.

Documentar la realización de un estudio aplicado de las capas ontológica y lógica de la web semántica.

Crear un guía para el modelado y desarrollo de aplicaciones web multimodales que incluyan modelado de datos basado en web semántica

Desarrollar y evaluar un prototipo de portal de información sobre el problema de desplazamiento forzado en la ciudad de Cartagena que incluya modelado semántico de los datos hasta la capa de reglas de la web semántica.

1.3 PLANTEAMIENTO DEL PROBLEMA Y JUSTIFICACIÓN

A partir de la década de los ochenta, con el surgimiento de los computadores personales, se abre la posibilidad de que en cada hogar del planeta exista un computador. Más adelante, a comienzos de la década de los noventa, con el nacimiento de Internet, éstos computadores pasan de ser máquinas aisladas con funcionalidad limitada, a convertirse en la red mundial, que ha transformado a la sociedad de la era de la información. Adicionalmente, con la aparición de la telefonía celular, los avances en el campo de las redes inalámbricas, y el desarrollo de la computación móvil y ubicua, se amplía a tal escala el campo de acción y de posibilidades que brindan las nuevas tecnologías, que obliga a redefinir los conceptos de tiempo y espacio, haciendo posible tener acceso a la información en cualquier momento y en cualquier lugar.

Pero esto no es todo, por cuestión de costo, es más fácil tener en casa una línea telefónica que un computador. Razón por la cual surgen lenguajes de marcado, como VoiceXML [67], que permiten crear aplicaciones de voz, ampliando considerablemente el espectro de usuarios de la web. Sin embargo, este tipo de avances tecnológicos, aunque solucionan el problema del acceso a la información, generan nuevas necesidades, tendientes a organizar y estructurar dicha información, de tal manera que el usuario final pueda realizar búsquedas enriquecidas, integración de datos, navegación y automatización de tareas, desde cualquier dispositivo, en cualquier momento y en cualquier lugar.

Como respuesta a éstas necesidades, nace el concepto de Web Semántica, cuyo objetivo es desarrollar estándares y diseñar tecnologías que permitan a las máquinas entender más la información sobre la web. Con la web semántica no sólo se obtienen resultados más exactos sino que también se puede integrar información de diferentes fuentes, conociendo la información a comparar, y brindando posibilidades para automatizar servicios en diferentes dominios, desde futuras librerías electrónicas hasta comercio electrónico y servicios de salud.

La rapidez con que se han presentado los cambios descritos anteriormente, ha generado un gran problema para los desarrolladores de aplicaciones, puesto que, si bien dichas tecnologías están documentadas, es muy poca la información que sirve de guía, cuando se quiere realizar un trabajo de construcción de software

que combine las facilidades que proporciona la interacción multimodal para el usuario final, con las ventajas y el potencial que representa el modelamiento semántico de datos. Además, lo poco que se encuentra, abarca cada uno de éstos aspectos (multimodalidad y web semántica) en forma separada. Esto dificulta las actividades de análisis, diseño, desarrollo y prueba, al momento de construir la solución, puesto que aumenta el grado de complejidad del problema, al obligar al equipo de trabajo a improvisar técnicas no validadas en la ejecución del proyecto, incrementando los factores de riesgo, y aumentando los tiempos de obtención de resultados, por el alto grado de incertidumbre que implica el realizar una tarea sin tener claras las pautas de ejecución del proceso.

Por otra parte, no se tiene en cuenta la sinergia que existe entre el modelamiento semántico y el acceso multimodal, al permitir generar interfaces multimodales inteligentes capaces de corregir y mejorar el proceso de reconocimiento de la modalidad de entrada, a partir de consultas específicas que se hacen al modelo semántico representado a través de la ontología, subutilizando las ventajas reales que representa el uso combinado de éste tipo de tecnologías.

En este orden de ideas, se hace clara la necesidad de proponer una Guía para el Modelado y Desarrollo de Aplicaciones Web Multimodales Basada en Web Semántica, que facilite la labor del desarrollador de este tipo de aplicaciones, y le permita disminuir considerablemente el tiempo de desarrollo, al proponer una serie de pautas, que definan las aspectos relevantes al momento de modelar y construir una solución multimodal basada en Web semántica.

Se pretende que el documento resultado de la investigación sirva de guía en cada una de las actividades que involucra el proceso de construcción de software, a los diferentes equipos de trabajo responsables de cada un de estas actividades. De igual manera, dicho material servirá como elemento didáctico, a quienes deseen aprender a modelar y construir aplicaciones web multimodales que involucren modelado semántico de datos.

En forma paralela, y con el ánimo de validar la propuesta, se construirá un prototipo de portal de información sobre el problema de desplazamiento forzado en la ciudad de Cartagena, que incluya modelado semántico de los datos hasta la capa de reglas de la web semántica; lo que permite solucionar una necesidad concreta del contexto, en este caso el Centro de investigación de la Universidad San Buenaventura Cartagena.

1.4 TRABAJOS RELACIONADOS

Son numerosos los trabajos que se ha realizado en el área de la multimodalidad y la web semántica, entre los cuales cabe destacar los siguientes, por su estrecha relación con el proyecto en curso.

“Framework for ubiquitous and voice enabled web applications development: A Transport Information Systems Test-Bed” elaborado por el Dr. Eduardo Carrillo [19] en calidad de tesis de disertación para la obtención del título Doctor en Tecnología de la Información, Computación y Comunicaciones, otorgado por la Universidad de Valencia en 2004. Como resultado se obtuvo un modelo para grupos de dispositivos móviles, una arquitectura que agrupa modelos y servicios en calidad de guía par el modelamiento físico de portales móviles semánticos, un marco que guía el desarrollo de aplicaciones web móviles semánticas.

Guía turística multimodal y multilingüe para París, desarrollada desde febrero de 2001 hasta diciembre de 2002 por EURESCOM [31] (European Institute for Research and Strategic Studies in Telecommunications) en el contexto del proyecto MUST, (MUltimodal, multilingual information Services for small mobile Terminals) . Su desarrollo dio como resultado la construcción en corto tiempo de un software de tiempo real que tolera las latencias de las respuestas, pero que según los expertos requiere mejoras sustanciales en su interface de usuario.

Interface multimodal para la exploración de objetos remotos con pocos censores de datos, realizada por Jacopo Aleotti, Stefano Caselli y Monica Reggiani [2], del departamento de ingeniería de la Universidad de Parma (Italia). Quienes concluyen que las principales características de la interface son la fusión geométrica de la poca proximidad y la percepción en el sitio remoto y la vibración de retroalimentación integrada al guante de realidad virtual. El trabajo fue realizado para tener una mejor compresión de la funcionalidad de las interfaces multimodales.

Economía del Hidrógeno a través de la Tecnología Web Semántica, cuyos autores son Jane Hunter, John Drennan, Suzanne Little, plantean la necesidad de identificar combustibles más eficientes, con un menor costo de producción, una vida más extensa y de componentes reciclables [45]. Para lograrlo se requiere entender la estructura microscópica de sus componentes, analizar detalladamente la forma en que interactúan, y predecir cómo los componentes pueden comportarse en determinadas condiciones. Esto es posible manipulando el material científico por medio de técnicas de administración del conocimiento, como la tecnología Web semántica, para poder asimilar la gran cantidad de microestructuras, comportamientos y manipular datos que se adquieren durante la investigación, en el proyecto que han denominado FUSION (Fuel Cell Understanding through Semantic Inferencing, Ontologies and Nanotechnology). La aplicación de la tecnología Web Semántica cobra sentido en el momento de analizar tres conjuntos complejos de datos: microestructuras, comportamiento y procesamiento, debido a que por su cantidad sería imposible ser procesado por el ser humano, lo que obliga a utilizar los más sofisticados mecanismos de minería

de datos. Los autores concluyen diciendo que describen como han extendido, refinado y aplicado la tecnología Web semántica en un problema científico del mundo real: optimización de la celda combustible.

1.5 MÉTODO DE INVESTIGACIÓN Y DESCRIPCIÓN DE RESULTADOS

Para lograr el objetivo de la tesis se realizó un estudio detallado sobre Interacción multimodal y sobre Web Semántica, tomando como referente principal la documentación que presenta el consorcio W3C en su sitio web. Dicho estudio se documenta, logrando cumplir el primer y segundo objetivo específico de la tesis. El estudio se centró en los aspectos más relevantes de cada una de estas tecnologías, permitiendo una clasificación del los lenguajes de marcado para el desarrollo de aplicaciones multimodales, y la identificación de los elementos que conforman la capa ontológica de la web semántica.

Comprendida cada una estas tecnologías, y teniendo en cuenta que la construcción de un portal web es un ejercicio de ingeniería de software, se tomó la decisión de aplicar el patrón de diseño denominado Modelo Vista Controlador MVC, ideal para aplicaciones que soportan distintos tipos de interface de usuario, y múltiples vistas sincronizadas del mismo modelo, propias del entorno multimodal. Además, por que permite separar la capa de presentación de la lógica y los datos, en este caso modelado de datos basado en web semántica, haciendo posible trabajar en forma separada, mas no aislada, los aspectos propios de cada una de estas tecnologías. De igual manera, hace posible aplicar los principios de modularidad e independencia funcional, que plantea la ingeniería de software, al realizar las tareas de modelar y desarrollar aplicaciones.

Todo esto fundamenta la decisión de dividir la guía en dos partes, un destinada al modelo de interacción multimodal y otra al modelo semántico, sin que ello implique que se trabajen en forma aislada. La elaboración de la guía se hizo a partir de la experiencia en el desarrollo del prototipo del portal de información sobre el problema de desplazamiento forzado en la ciudad de Cartagena. Para lo cual se utilizó como referente, en el modelo de interacción multimodal, la plataforma de interacción multimodal propuesta por el Consorcio W3C, y en modelo semántico, el documento titulado Desarrollo de Ontologías 101: Guía para crear tu primera ontología de Natalya F Noy [71]

Teniendo en cuenta que la construcción del prototipo obedece a un proceso de ingeniería de software, se seleccionó como Metodología de desarrollo de portal el Proceso Unificado de Desarrollo de Software, por todas las ventajas que representa el desarrollo iterativo incremental, centrado en la arquitectura y dirigido por casos de uso. Los resultados se presentan de acuerdo a cada una de las actividades que propone dicha metodología, aclarando que la guía complementa

específicamente las actividades de diseño e implementación, sin ser exclusiva para ésta metodología.

Mientras que en la primera parte de la guía se redactan una serie de pautas que no implican un orden algorítmico sino estructural, para poder diseñar y desarrollar el modelo de interacción multimodal, en la segunda parte se indican unos pasos que representan un orden algorítmico para el diseño y desarrollo del modelo semántico. La guía se ilustra por medio del prototipo de portal de información, aplicando cada una de las pautas y pasos para la construcción de los respectivos modelos.

Para construir el portal de información sobre el problema de desplazamiento forzado en la ciudad de Cartagena, se realizó un proceso de recolección de información, brindada por el Centro de Investigación de la Universidad de San Buenaventura Cartagena CEIN, a través de entrevistas con su director y varios investigadores pertenicientes a dicho grupo. Esta información se utilizó para construir el Modelo del Negocio.

El prototipo construido fue evaluado permanentemente, haciendo uso de técnicas de caja blanca, con el fin de mejorarlo y de ésta manera depurar la guía. Para lograrlo fue necesario seleccionar una serie de recursos tecnológicos que se detallan en el anexo: Recursos Necesarios

La tesis fue desarrollada según lo muestra el anexo Cronograma de Actividades, el cual es acorde a la metodología de desarrollo de software seleccionada, que propone un modelo iterativo e incremental.

En los siguientes capítulos se presentan los resultados del trabajo realizado, comenzando por el estudio sobre el estado del arte de los lenguajes de marcado para la interacción multimodal (sección 3 del segundo capítulo), donde se propone una clasificación de dichos lenguajes, teniendo en cuenta la plataforma de interacción multimodal que presenta el consorcio W3C. En la sección cuatro del mismo capítulo, se incluye el estudio sobre la capa lógica y ontológica de la web semántica, hecho a partir del modelo clásico de capas, presentado por Tim Berners-Lee en el W3C [12] y su modelo mejorado [13]. Posteriormente se presenta la guía para el modelado y desarrollo de aplicaciones web multimodales que incluyan modelado de datos basado en web semántica (capítulo 3), la cual fue estructurada en dos partes: Modelado de la interacción mulitmodal y modelado semántico, para facilitar su compresión. Finalmente, se explica la construcción del prototipo de portal de información sobre el problema de desplazamiento forzado en la ciudad de Cartagena (capítulo 4), en calidad de ejemplo para demostrar la aplicación de la guía propuesta.

2. MARCO TEÓRICO (ESTADO DEL ARTE)

El fundamento teórico que soporta el presente proyecto está conformado por dos grandes temas: La interacción Multimodal y Web Semántica. Con en fin de entender cada una de estas tecnologías, se han incluido como resultado del estudio de la revisión bibliográfica los siguientes tópicos: Acceso Multimodal y sus requerimientos, la Interface de usuario multimodal, Plataforma de interacción multimodal, Arquitectura de interacción Multimodal, Lenguajes de marcado para la interacción Multimodal, Ontología, Capas de la Web Semántica, Marco de Descripción de Recursos, Lenguajes para la Construcción de Ontologías, Lenguaje de reglas para la Web Semántica y sus herramientas.

2.1 INTERACCIÓN MULTIMODAL

Los avances de los últimos años en el área de las comunicaciones, y la disminución del tamaño de los dispositivos con capacidad de procesamiento, abren la posibilidad de disponer de los recursos informáticos en todo momento y en todo lugar, lo cual obliga a los desarrolladores de software, a construir aplicaciones que permitan múltiples formas de interacción, acordes a las características y limitaciones del medio que utilice el usuario; sin dejar de lado el patrón de desarrollo de software denominado: Modelo Vista Controlador, cuya principal finalidad consiste en separar las capas de presentación, lógica y datos.

2.1.1 Acceso Multimodal

El acceso multimodal permite al usuario interactuar con la aplicación de diferentes formas: entradas con voz, teclado, ratón, lápices; y salida con voz sintetizada, audio, texto plano, video en movimiento y / o gráficos. Cada uno de estos modos puede ser utilizado independientemente o en forma concurrente [48]. No hay que confundir el acceso multimodal con el acceso multicanal, el cual consiste en acceder a los datos o aplicaciones por diferentes medios o canales, por ejemplo al consultar el estado de cuenta por medio de un navegador de Internet, o a través de una llamada telefónica.

Las aplicaciones multimodales deben estar en capacidad de adaptarse a las características y limitaciones del dispositivo, las preferencias del usuario y las condiciones del entorno. Además deben ser fáciles de programar dinámicamente para que se ajusten a los cambios, haciendo más disponibles los diferentes modos de interacción en un momento determinado. Adicionalmente se debe tener la posibilidad de crear aplicaciones que permitan la interacción a múltiples usuarios

desde múltiples dispositivos, trabajando individual o conjuntamente, aumentando la interacción hombre máquina.

Por sistema multimodal se entiende aquel que permite a un usuario comunicarse con la aplicación usando diferentes modalidades como: voz (lenguaje natural), gestos, escritura, audio visual, etc [93]; teniendo en cuenta que el usuario puede ser considerado para operar en un contexto de entrega, o sea, bajo un conjunto de atributos que caracterizan las capacidades de los mecanismos de acceso en términos de perfil de dispositivos, perfil de usuario (identificación, preferencias, patrones de uso) y situaciones. El usuario interactúa con la aplicación en el contexto de una sesión, usando una o más modalidades, que pueden ser realizadas a través de uno o más dispositivos, Durante la sesión el usuario puede suspender o reanudar la interacción con la aplicación haciendo uso de la misma modalidad o intercambiando diferentes tipos de modalidades. Una sesión es asociada a un contexto, el cual registra las interacciones con el usuario.

En un sistema multimodal, un evento es la representación de una ocurrencia asíncrona de interés al sistema multimodal. Los eventos asociados con información a cerca de la interacción del usuario se denominan eventos de entrada, y se clasifican en secuencial (una entrada en un solo modo, puede cambiar con el paso del tiempo), simultaneo (entradas simultaneas que pueden venir de diferentes modalidades, pero que no están combinadas, cada una es interpretada una después de la otra en el orden en que se han recibido), compuesto (entradas recibidas por múltiples modalidades al mismo tiempo y que son tratadas como si fuera una sola).

Se han identificado dos usos de la multimodalidad: La multimodalidad suplementaria, cuando permite cualquier interacción (entrada o salida) a través de cada modalidad como si fuera una sola. Y la multimodalidad complementaria cuando la interacción con una modalidad es usada para complementar la interacción con otra.

Con el fin de mantener el orden temporal de los eventos, se registra el tiempo de su ocurrencia, al igual que la fuente que lo genera, asignándole un controlador. Los sistemas multimodales también producen eventos externos de salida.

La sincronización del comportamiento de una aplicación describe la forma en la que cualquier entrada en una modalidad se refleja en la salida en otra modalidad, de igual manera como las modalidades de entrada son combinadas. Las salidas generadas por un sistema multimodal pueden tener varias formas, por ejemplo audio (utilizando generación de lenguaje natural, voz digitalizada sinterizando audio TTS Text To Speech [29]), visual, sincronización de los movimientos de los labios de una cara con el audio emitido, entre otras; de ahí la importancia de la recomendación SMIL 2.0 (Synchronized Multimedia Integration Language) [95] de

W3C que permite la sincronización de los medios en usos audio-visuales interactivos.

La interacción entre el usuario y la aplicación puede ser interpretada como una serie de diálogos dirigidos por un administrador de interacción. Un diálogo es una interacción entre el usuario y la aplicación en la cual se turnan para actuar. El administrador de interacción genera y actualiza la presentación, procesando las entradas del usuario, el contexto de la sesión y otras posibles fuentes de conocimiento que determinan la intención del usuario. De igual manera define estrategias para determinar focos e intentos de ambigüedad, corrigiendo y confirmando los subdiálogos.

El administrador de interacción puede usar entradas del usuario, el contexto de la sesión, fuentes externas de conocimiento y eliminación de la ambigüedad, corrección, configuración de subdiálogos para determinar la intención y el interés del usuario; además mantiene el contexto y el estado de la aplicación administrando la composición de entradas y sincronización de modalidades, la interface y la lógica del negocio, produciendo salidas observables por el usuario. En algunas arquitecturas el administrador de interacción tiene componentes distribuidos, utilizando mecanismos basados en eventos para la coordinación.

2.1.2 Requerimientos del acceso multimodal

El grupo de interacción multimodal de la W3C ha definido las siguientes especificaciones para los desarrolladores de aplicaciones sincronizadas que permitan el acceso a través de múltiples modalidades o dispositivos con un amplio rango de capacidades. [93]

- **Requerimientos generales**: Una aplicación multimodal debe cumplir con las siguientes características: escalabilidad a través de un amplio rango de dispositivos con diferentes tipos de capacidades, uso complementario y suplementario de diferentes tipos de modalidades, sincronización de modalidades, soporte multilingüe, fácil de implementar, accesible, segura, manteniendo la privacidad de la información; permitiendo conservar el contexto de entrega, al mantener las características y las capacidades de los dispositivos, al igual que las preferencias del usuario; y describiendo el flujo de navegación a través de la aplicación.

- **Requerimientos de la modalidad de entrada:**

    - Se debe proveer un mecanismo para indicar y adherir a la información la modalidad que se utiliza, permitiendo definir cada paso del proceso que se realiza antes que la entrada sea identificada.

- Soportar la entrada secuencial multimodal, permitiendo especificar la modalidad o dispositivo, insinuando el intercambio de modalidad, en cualquier momento dependiendo de la necesidad del usuario y la naturaleza de la interacción de entrada

- Soportar la entrada simultanea multimodal, permitiendo al autor especificar la granulidad de sincronización, y como se desenvuelve la aplicación multimodal cuando dicha granulidad es modificada por factores externos, pudiendo definir un comportamiento de entrada por omisión que permita mecanismos de sobreescritura.

- Soportar la entrada multimodal compuesta, dando la posibilidad de combinar entradas, teniendo en cuenta las capacidades de coordinación en el contexto de entrada, permitiendo al autor especificar los mecanismos usados para decidir cuando y como coordinar entradas para combinarlas.

- Soportar modos de entrada a partir de: teclados o mecanismos de pulsación, dispositivos con punteros (ratón, pantallas táctiles, lápices, etc), interfaces de entrada combinadas como consolas de juego; entradas de audio, video, lenguaje de señas. Siendo lo ideal que permita combinación de reconocimiento de audio e imagen combinado y entradas táctiles entre otros.

- Soportar la representación del significado de la entrada del usuario, definiendo una semántica de entrada generada por los componentes de la interface de usuario, independientemente de la modalidad que se utilice.

- Disponer de un mecanismo de coordinación de restricciones de entrada a través de la diferentes modalidades.

- **Requerimientos de la modalidad de salida:** Se deben permitir medios de salida secuencial, dando la posibilidad al usuario de seleccionar la modalidad dependiendo de la situación en que se encuentre; o medios de salida simultanea, con la habilidad de sincronizar los diferentes medios de salida, tales como: Audio, visual, objetos SMIL, audio sintetizado, MIDI (Musical Instrument Digital Interface) y cualquier otro que pueda surgir en el futuro. De igual manera se debe soportar la especificación de cuales medios de salida deben procesar y como deben hacerlo, proveyendo mecanismos para identificar como se puede realizar o extender diferentes modalidades.

- **Requerimientos de Arquitectura, integración y puntos de sincronización**

  - La especificación multimodal debe permitir el uso e integración de lenguajes estándares existentes (visuales, de voz y multimediales), permitiendo

sincronizar las entradas y salidas a través de SMIL 2.0 como mecanismo de control.

- La especificación multimodal debe expresarse en términos de XHTML [4], asegurando la independencia completa de las capas de presentación, lógica y datos.

- Debe permitir la separación del modelo de datos, de la capa de presentación y la lógica de la aplicación.

- Deben existir eventos asociados a los cambios del contexto, y mecanismos para especificar cómo manejar estos cambios para adaptar la aplicación multimodal.

- La sincronización debe presentarse a nivel de los formularios de entrada, de los campos de entrada, de página, de eventos (de entrada, de salida, de entrada y salida), de sesión.

- Las Interfaces de entrada y salida deben ser independientes aún cuando utilice la misma modalidad.

- Diferentes modalidades o dispositivos distribuidos a través de la red, deben permitir al usuario la capacidad de interactuar con diferentes dispositivos, manteniendo la sincronía.

- Se debe soportar el procesamiento de entradas y salidas en forma distribuida.

- Se debe especificar cómo dirigir eventos de entrada, a partir de aplicaciones externas y generar eventos de salidas externas utilizados por otros procesos.
- Proveer mecanismos que permitan identificar la posición en el tiempo de eventos de entrada y salida, uno respecto del otro.

- **Ejecución y desarrollo:**

  - La aplicaciones multimodales deben ser creadas para todas las configuraciones de desarrollo relevantes, donde las funciones son divididas o combinas en un solo equipo, o distribuidas en varios dispositivos o servidores

  - Debe ser compatible con el desarrollo basado en el uso de agentes que se ejecutan sobre plataformas móviles, soportado por redes móviles,

considerando aspectos como las limitaciones del ancho de banda y retrasos de tiempo.

- Se debe soportar la generación, representación e Intercambio de eventos de entrada, y resultados o procesos de entrada o salida.

- Se debe permitir la generación de eventos síncronos y asíncronos con sus respectivos controladores, en forma local o remota

2.1.3 Interface de usuario multimodal

Los requerimientos de la modalidad de entrada se logran a partir de la construcción de la interface de usuario, para la cual se recomienda tener en cuenta los siguientes principios al momento de diseñarla [59]:

Satisfacer las restricciones propias del mundo real, que dependen de la naturaleza de la tarea que el usuario intenta realizar, de las limitaciones físicas del usuario, y de las limitaciones del entorno en el cual se encuentra.

La comunicación con el usuario debe ser clara, concisa y consistente, manteniendo directrices organizacionales que permitan evidenciar una estructura verbal y visual, usando pausas para dividir la información en fracciones naturales, evitando ambigüedades y confusión.

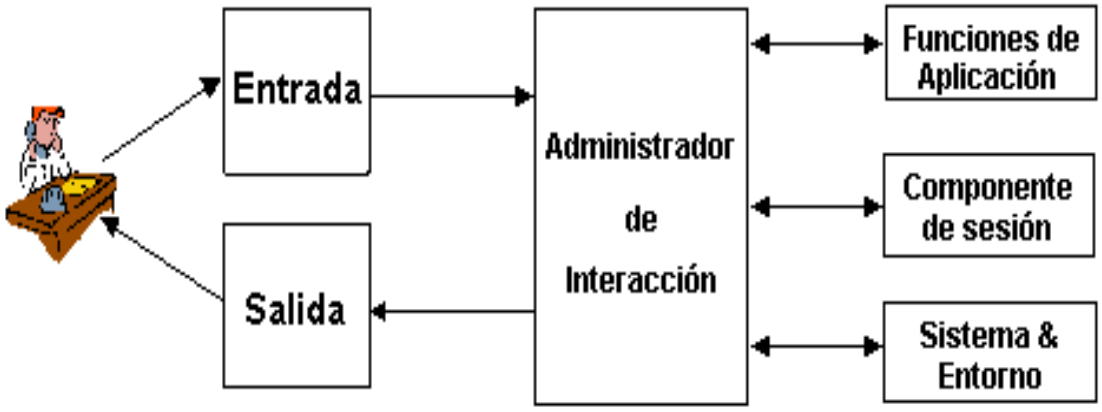
Ayudar al usuario a recuperar el error en forma rápida y eficiente, permitiendo detectar los errores, y guiando al usuario para que la recuperación sea en forma natural.

Hacer sentir cómodo al usuario, brindándole el control absoluto, evitándole cargar la memoria de información adicional, por medio de una interface agradable y previsible, fácil de adaptar a sus necesidades y habilidades, a las condiciones de conexión, al ancho de banda de la red, y demás condiciones del entorno.

2.1.4 Plataforma de interacción multimodal

La plataforma de interacción multimodal identifica y relaciona los lenguajes de marcado para los sistemas de interacción multimodal, al igual que los componentes representativos, y el conjunto de funciones que realizan. También describe los amplios modos de entrada y salida usados actualmente y que pueden ser extendidos en el futuro [60]. Su representación gráfica se presenta en la siguiente figura.

Figura Nº 1. Plataforma de interacción multimodal

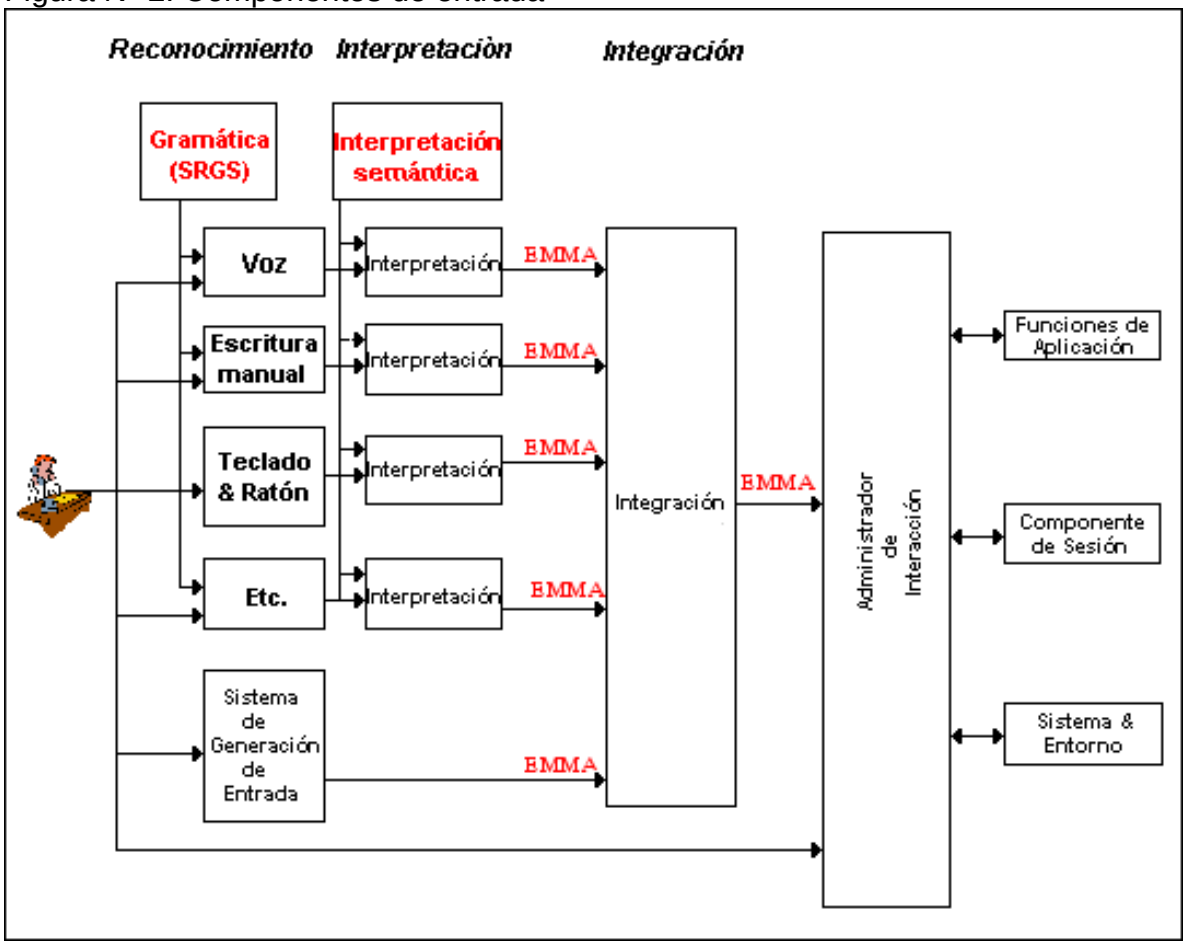


Fuente: http://www.w3.org/TR/mmi-framework/

El usuario es quien realiza las entradas al sistema y percibe su respuesta, haciendo uso de múltiples modalidades, tales como audio, habla, escritura, teclado, entre otras. El administrador de interacción es un componente lógico que coordina datos y controla el flujo de ejecución, a partir de las diferentes interfaces de los componentes de las modalidades de entrada y salida. También es el responsable de mantener el contexto de la aplicación. El componente de sesión provee una interface para soportar diferentes estados, controlar el tiempo y la persistencia de cada sesión de la aplicación multimodal. Los componentes de entorno y sistema permiten controlar la interacción, para encontrar la salida y responder a los cambios dependiendo de las capacidades de los dispositivos, preferencias del usuario y condiciones del entorno.

La siguiente figura representa los componentes de entrada.

Figura Nº 2. Componentes de entrada



Fuente: http://www.w3.org/TR/mmi-framework/

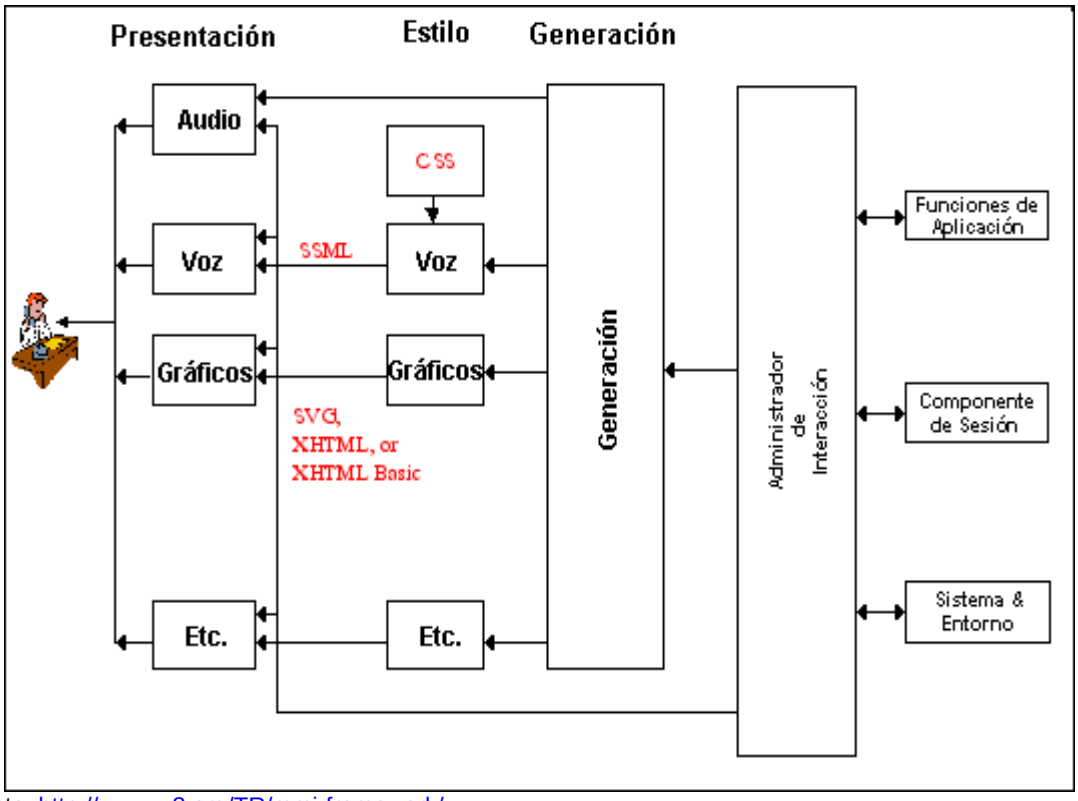
Como se puede observar, los componentes de entrada están divididos en tres categorías: De reconocimiento, encargada de la captura de la entrada natural, transformando su formato para el posterior procesamiento, pueden utilizar descriptores de gramática por medio de un lenguaje de marcado de gramática. De interpretación, responsable de identificar el significado o semántica asignado por el usuario a la entrada, su salida puede ser representada con EMMA (Extensible MultiModal Annotation markup language) [23], como lenguaje para representar la semántica o significado de los datos. De integración, encargada de combinar la entrada desde los diferentes componentes de intepretación.

Los componentes de salida están conformados por:

Componente de generación: determina cual o cuales son los modos (complementarios o suplementarios) a usar, para representar la información generada por el administrador de interacción para el usuario. El lenguaje de representación interna usado para describir la salida de este componente,

actualmente es tema de discusión para el grupo W3C. Como se observa en el gráfico, el paso de la información de salida por este componente no es obligatorio.

Figura Nº 3. Componentes de salida



Fuente: http://www.w3.org/TR/mmi-framework/

Componente de estilo agrega información a cerca de cómo se presentará la salida, definiendo aspectos como pausas en la voz, inflexiones entre otros. Se construye a partir de etiquetas del Lenguaje de Marcado SSML (Speech Synthesis Markup Language) [18]. Los componentes de estilo gráfico se crean con XHTML o SVG (Scalable Vector Graphics) [62].

Componente de presentación: convierte la información del componente de estilo en un formato fácil de entender por el usuario, como lo es la conversión de texto a voz, representaciones gráficas, etc. Para cada modo de salida le corresponde un componente de presentación. Para coordinar la sincronía de las salidas multimedia se utiliza SMIL (Synchronized Multimedia Integration Language)

La especificación de los componentes de entrada y salida se logra a partir de la encapsulación de interfaces basada en DOM (Document Object Model) [102] y su formalización, lo cual permite definir una plataforma neutral y una interface neutral de lenguaje de programación para los documentos, su estructura y su contenido.

Cada objeto de la interface de usuario debe especificar un conjunto de interfaces en términos de propiedades, eventos y métodos, usando un lenguaje de definición de interfaces formal.

El ***administrador de interacción*** es un componente lógico. Está contenido en el ambiente anfitrión de la interface del objeto. Las interfaces del objeto influyen una sobre otra al interactuar con el ambiente anfitrión, el cual provee administración de datos y control de flujo hacia la interface anfitrión de objetos. Algunos lenguajes candidatos para definir el ambiente anfitrión son: SVG, XHTML (XHTML + Xforms), y SMIL

El ***componente de sesión*** provee un mecanismo para simplificar la forma como los recursos son identificados en términos de fuente y destino de cada mensaje. Es de vital importancia para aplicaciones distribuidas que requieren de uno o más dispositivos y/o usuarios. Permite ocultar los detalles de los esquemas de nombres del recurso, los protocolos usados y provee una interface de alto nivel para requerir y liberar los recursos que intervienen en una sesión. Su función radica en simplificar las aplicaciones, permitir la temporalidad o persistencia de las sesiones, la réplica de estados a través de dispositivos, y la sincronización de las diferentes modalidades.

Los casos de uso que sirvieron de base para definir los componentes de sesión son: Dispositivos móviles con capacidad secuencial, llenado de formularios, especialmente cuando es necesario hacerlo a través del teclado y la voz; aplicaciones para múltiples dispositivos, que permitan conservar el estado para continuar su ejecución en otro dispositivo.

El desarrollador de aplicaciones multimodales debe estar en capacidad de crear software dinámicamente adaptable a los cambios de las capacidades de los dispositivos, las preferencias del usuario y las condiciones del entorno, por lo tanto el administrador de interacción debe determinar que información está disponible, y como depende esto del sistema, así se encuentre en un contexto centralizado o distribuido incluyendo múltiples dispositivos y usuarios.

Por eso, para determinar los componentes del sistema y el entorno, se han tenido en cuenta los siguientes escenarios: Los dispositivos móviles tienen capacidades y recursos limitados, y dependen de características como la ubicación, la fuerza de la señal, el nivel de ruido, la capacidad de la batería. Las preferencias del usuario, teniendo en cuenta que pueden ser dinámicas o estáticas.

2.1.5 Arquitectura de interacción Multimodal

A diferencia de la plataforma descrita anteriormente, la arquitectura indica cómo los componentes se asignan a los dispositivos hardware, y cómo los sistemas de

comunicación posibilitan que los dispositivos hardware se comuniquen unos con otros.

Existen muchas arquitecturas posibles que son consistentes con la plataforma de interacción multimodal, cada una de ellas cumple con las siguientes propiedades:

- Contienen un subconjunto de componentes de la plataforma. Una arquitectura multimedia contiene dos o más modos de salida, mientras que una arquitectura multimodal contiene dos o más modos de entrada.

- Los componentes pueden ser divididos y combinados: Una función de un componente puede ser dividida en varios módulos de la arquitectura, y la función de dos o más componentes puede ser combinada en un solo módulo de la arquitectura.

- Los componentes se asignan a los dispositivos hardware: Si todos los componentes son asignados al mismo dispositivo hardware, se dice que la arquitectura es centralizada. Una arquitectura cliente servidor consiste en dos o más dispositivos, varios dispositivos clientes contienen componentes de entrada y salida, y el servidor contiene el resto de componentes. Una arquitectura distribuida consiste de múltiples tipos de dispositivos conectados por un sistema de comunicación.

- Se especifican los sistemas de comunicación. Se diseñan protocolos específicos para el intercambio de mensajes, teniendo en cuenta los dispositivos hardware.

- Se especifica el modelo de diálogo. Los diseñadores especifican cómo los módulos se invocan y terminan, y cómo se interpreta una entrada para producir una salida.

2.1.6 Herramientas

Para implementar una arquitectura específica se necesitan lenguajes con características especiales, que permitan definir, manipular y comunicar entre sí, cada uno de los componentes de la plataforma multimodal descrita anteriormente. A continuación se presentan los lenguajes de marcado más representativos para la interacción multimodal.

XForms

Ha surgido como la mejor herramienta para la construcción de interfaces multimodales, debido a que garantiza los siguientes servicios [78]: Permite generar interfaces de usuario apropiadas para conectar diferentes tipos de dispositivos; provee mecanismos de retroalimentación para el usuario que realizan validación automática en el lado del cliente, al igual que en el servidor, además organiza las entradas en el servidor en una estructura conveniente para la aplicación final.

Esto se logra gracias a su arquitectura, que satisface las características del Modelo Vista Controlador, al presentar los siguientes componentes:

Model : Todos los aspectos no relacionados con la presentación son encapsulados por el modelo de datos de XForms. Dicho modelo incorpora una instancia de XML (Extensible Markup Language) [101] que lleva la entrada del usuario, las restricciones usadas para validarla, y los metadatos necesarios a cerca de cómo el usuario debe comunicarse con el servicio web.

User Interface: XForms define un vocabulario de la interface de usuario que consiste de controles abstractos y elementos de agregación usados para crear potentes interfaces de usuario. Dicho vocabulario ha sido diseñado para capturar la esencia de la interacción del usuario más que la presentación final, sobre cualquier dispositivo o cualquier modalidad específica, lo que lo convierte en la herramienta por excelencia para desarrollar aplicaciones web para diferentes dispositivos y modalidades.

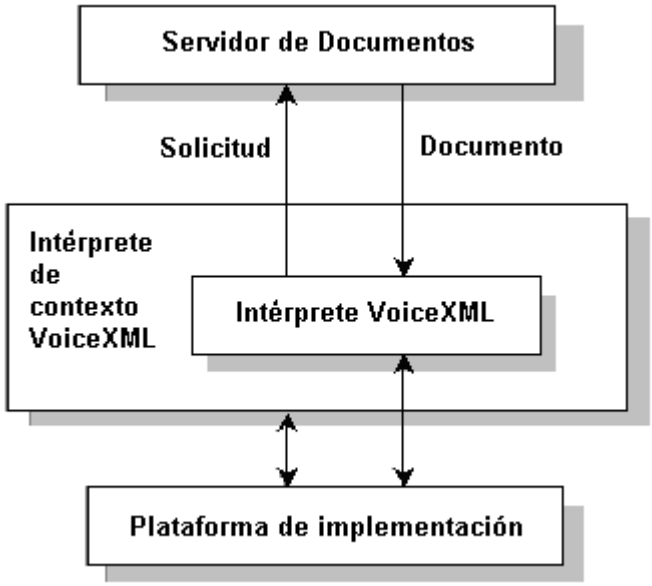
Submit: Permite a los creadores de las aplicaciones especificar donde, cómo y qué piezas de datos enviar al servidor web. De igual manera permite especificar que acciones tomar sobre las respuestas recibidas del servidor.

VoiceXML

Ha sido diseñado para crear diálogos de audio que se caracterizan por contar con voz sinterizada, audio digitalizado, reconocimiento de voz y entradas de teclado DTMF (Dual Tone Multi Frequency) [1], grabando las entradas de voz, teléfono, y combinando conversaciones. Su principal objetivo es brindar las ventajas del desarrollo basado en la web y entrega de contenidos para interactuar con aplicaciones que responden con voz, liberando al desarrollador de la administración de recursos y el bajo nivel de programación.
Esta integración de servicios de voz con datos es posible gracias al paradigma cliente servidor, presente en el modelo arquitectónico de VoiceXML, donde el servidor procesa la petición de una aplicación cliente, que viene a través del Interprete de contexto de VoiceXML. El servidor produce un documento VoiceXML como respuesta, el cual es procesado por el interprete de VoiceXML. La plataforma de implementación es controlada por el interprete de contexto de VoiceXML, encargado de tareas como controlar las llamadas que entran, cargar el

documento inicial de VoiceXML y responder la llamada, mientras que el Interprete de VoiceXML conduce el diálogo después de la respuesta. La siguiente figura representa la arquitectura VoiceXML.

Figura Nº 4. Arquitectura VoiceXML



Fuente http://www.w3.org/TR/2004/REC-voicexml20-20040316/#dml1

VoiceXML es un lenguaje de marcado que, minimiza la interacción cliente servidor especificando múltiples interacciones por documento, separa el código de interacción (VoiceXML) de la lógica del servicio (scripts), promueve la portabilidad de servicios a través de plataformas de implementación, presentando características de lenguaje que soportan diálogos complejos.

Este lenguaje permite la interacción hombre máquina por medio de un sistema de respuesta de voz que incluye: salida de voz sintetizada, salida de archivos de audio, reconocimiento de entrada de voz, reconocimiento de entradas DTMF, control del flujo de diálogo, características de teléfono como transferencia de llamadas y desconexión.

XHTML + Voice

Permite la interacción por medio del habla para el estándar de contenidos WWW, por medio de la interacción de un conjunto de tecnologías maduras, como XHTML y Eventos XML, con vocabularios XML desarrollados como parte de la plataforma de la interface de habla de W3C. Incluye módulos que soportan voz sintetizada, diálogos hablados, comandos y control, gramática del habla, y la habilidad para adjuntar controladores de voz para responder a eventos específicos DOM

(Document Object Model), reutilizando modelos de eventos familiares para los desarrolladores web. Las características de interacción de voz son integradas directamente con XHTML y Cascading Style Sheets CSS [103] , por lo tanto pueden ser consecuentemente utilizadas con contenidos XHTML.

XHTML + Voice ha sido diseñado para clientes web que soportan interacción visual y de voz. Para lograr este fin, los documentos primero se reformulan en VoiceXML 2.0 como una colección de módulos, que junto con el Lenguaje de Marcado de Voz Digitalizada (Speech Synthesis Markup Language SSML) y el formato de reconocimiento gramático de voz (Speech Recognition Grammar Format SRGF [44]), se integran con XHTML, usando módulos creados para el entorno XHTML + Voice. Finalmente el resultado se integra con Eventos XML como controladores de voz que pueden invocar a través de la interface estándar DOM2 EventListener.

Los módulos que conforman XHTML + Voice son [5]: Módulo de Salida de audio con voz y sin voz, que está definido por la especificación SSML, Módulo de diálogos de voz, usados para implementar el control del diálogo en la interacción; Módulo de gramática del habla, Módulo de eventos tipo VoiceXML, usado para crear eventos escucha que responden a los eventos de voz. Módulo controlador de eventos VoiceXML, cuya semántica es definida por la especificación VoiceXML 2.0

Speech Application Language Tags SALT

Es un lenguaje de marcado para interfaces de habla. Consiste en un pequeño conjunto de elementos XML, con atributos asociados y propiedades de objetos DOM, eventos y métodos para la interface de audio de páginas Web. SALT puede ser usado con HTML, XHTML y otros estándares para escribir interfaces de habla para aplicaciones de solo voz y aplicaciones multimodales [85].

Los dos escenarios más importantes en los que se utiliza SALT son: Aplicaciones multimodales, que soporten entradas y salidas con voz, y aplicaciones de solo voz, exclusivamente para teléfonos, en las que SALT se encarga de administrar el flujo de interacción del diálogo.

SALT ha sido diseñado de acuerdo a los siguientes principios: Integración del habla con páginas Web, separación de la interface de la lógica del negocio y los datos, potencia y flexibilidad en el modelo de programación, reutilización de los estándares existentes para gramática, salida de habla y resultados semánticos; amplio rango de dispositivos, y costos mínimos de autorización para diferentes tipos de dispositivos y modalidades [85].

Ink Markup Language InkML

Una interface basada en lápiz óptico es posible gracias a dispositivos que permiten traducir los movimientos del lápiz en “tinta digital”, que puede ser reconocida por un software que convierta la entrada del lápiz en una acción apropiada en el computador. El problema consiste en que cada productor de dispositivos y software usa su propio formato digital, haciendo difícil el desarrollo de aplicaciones independientes del dispositivo. Como solución surge InkML, que es un lenguaje de marcado que provee una plataforma simple y neutral al formato de datos que permite el intercambio de tinta digital entre aplicaciones [32].

InkML proporciona una interface de usuario robusta y flexible para sistemas multimodales que pueden ser integrados con otras modalidades de entrada como el habla. La robustez es posible gracias a la redundancia de la modalidad que puede ser usada para compensar la imperfección en el reconocimiento de cada modo individualmente. El alto grado de flexibilidad es posible por que los usuarios pueden escoger el modo más apropiado para realizar una tarea o emitir comandos.

La tarea más importante en la entrada de tinta digital es la identificación del caracter que se guarda como resultado. Todo documento InkML está contenido en por el elemento etiqueta <ink>. En su interior, el elemento de datos más importante es <trace>, que representa una secuencia de puntos Ink contiguos correspondientes a las coordenadas de la posición del lápiz. El elemento que relaciona las entradas con las entidades es <bind>

Extensible MultiModal Annotation markup language EMMA

Este lenguaje de marcas ha sido diseñado para ser usado por sistemas que proporcionan interpretación semántica a una variedad de entradas, incluyendo, pero no necesariamente limitada a: el habla, texto en lenguaje natural, y entradas de interface gráficas de usuario. Se espera que inicialmente sea utilizado como formato estándar para el intercambio de datos entre los componentes de un sistema multimodal, generado normalmente en forma automática por los componentes de interpretación, para representar la semántica de las entradas del usuario.

El lenguaje provee un conjunto de elementos y atributos, que se centran en la anotación cuidadosa de la interpretación de las entradas, para lo cual maneja tres tipos de datos [23]:

Datos de instancia: Son los correspondientes a la información de entrada de una aplicación específica. Se debe tener en cuenta que las alocuciones pueden ser ambiguas, con respecto a los valores de entrada, y que un documento puede tener más de una instancia.

Modelo de datos: Son las restricciones sobre la estructura y los contenidos de una instancia. El modelo de datos es típicamente preestablecido por una aplicación, y puede ser implícito, o sea no especificado.

Metadatos: Son las anotaciones asociadas con el contenido del dato en una instancia. Los valores de dichas asociaciones se unen a los procesos de entrada en tiempo de ejecución.

EMMA usa como componentes un administrador de interacción y un integrador multimodal, para generar componentes de Reconocimiento de voz, reconocimiento de escritura, mecanismos de entendimiento del lenguaje natural, e interpretes de otros medios de entrada.

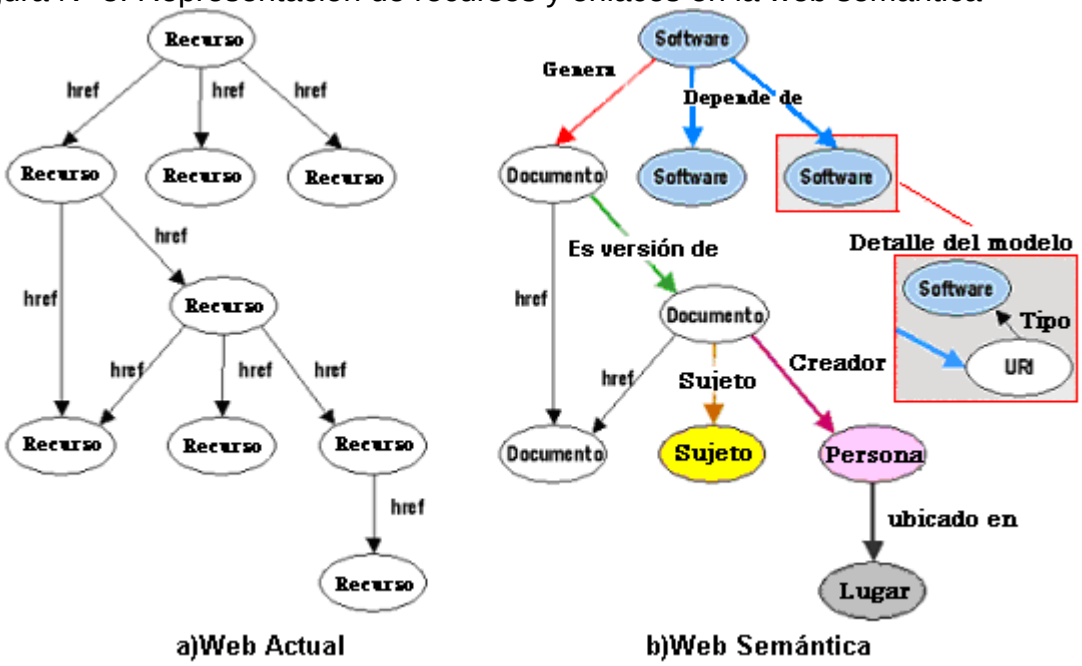
2.2 WEB SEMÁNTICA

El objetivo de la web semántica es desarrollar estándares y diseñar tecnologías que permitan a las máquinas entender más la información sobre la web, para soportar búsquedas enriquecidas, integración de datos, navegación y automatización de tareas. Con la web semántica no sólo se obtienen resultados más exactos sino que también se puede integrar información de diferentes fuentes, conociendo la información a comparar, y brindando posibilidades para automatizar servicios en diferentes dominios, desde futuras librerías electrónicas hasta comercio electrónico y servicios de salud [11].

Actualmente los recursos que se encuentran en la web, han sido registrados para el consumo humano, por lo tanto no contienen metadatos que expliquen su uso y la relación que tiene con otros, de ahí la necesidad de la web semántica, para que permita asignar tipos a los recursos y enlaces, para darles la categoría de conceptos, como se muestra en la siguiente figura.

La web semántica se implementa haciendo uso de las tecnologías: URI (Uniform Resource Identifier) [100], XML (Extentsible Markup Language), RDF (Resource Description Framework) [53], RDF Schema, ontologías, agentes inteligentes, firmas digitales y redes de confianza. Y se fundamenta en los siguientes principios [56]: Cualquier cosa puede ser referenciada por un URI, los recursos y los enlaces pueden tener tipos, se tolera la información parcial, no es necesaria la verdad absoluta, se permite la evolución, y el diseño es mínimo.

Figura Nº 5. Representación de recursos y enlaces en la web semántica



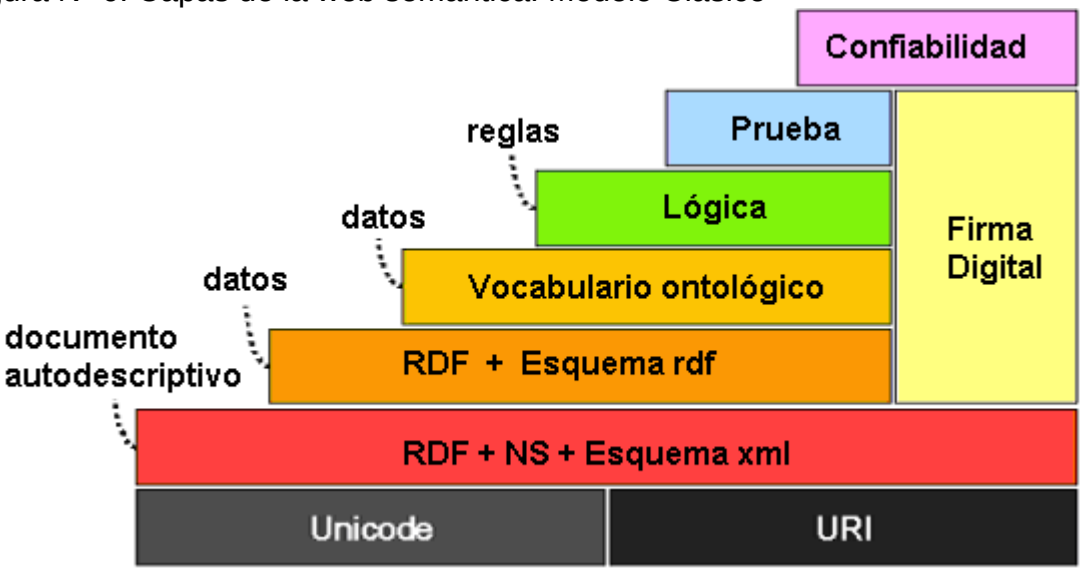
Fuente: http://www.w3.org/2001/12/semweb-fin/w3csw

2.2.1 Capas de la Web Semántica

Dichos principios se implementan en estándares y en las capas de la tecnología web, representadas en la figura Nº 6 .

La capa Unicode y URI asegura el uso de un conjunto de caracteres internacional y permite asignar un significado a los objetos identificados en la Web Semántica. La capa XML + NS + xmlschema nos permite integrar las definiciones de la Web Semántica con otros estándares XML. Con RDF y RDFShema se hace posible crear sentencias con URI acerca de objetos, y definir vocabularios que puedan ser referenciados por medio de URI. Esta es la capa donde se pueden asignar tipos a los recursos y enlaces. La capa de ontología soporta la evolución de vocabularios, ya que permite definir relaciones entre diferentes conceptos. La capa de firma digital se utiliza para identificar alteraciones en el documento. La capa lógica posibilita la escritura de reglas que permiten realizar consultas e inferir conocimiento, mientras que la capas de prueba las ejecuta y evalúa su confiabilidad, junto con el mecanismo de pruebas para aplicaciones de la capa de confiabilidad.

Figura Nº 6. Capas de la web semántica. Modelo Clásico



Fuente: http://www.w3.org/2001/12/semweb-fin/w3csw

2.2.2 Marco de Descripción de Recursos

Debido a la necesidad: de automatizar el procesamiento de la información en Internet a través de agentes, de interactuar entre aplicaciones con el fin de crear nueva información, de hacer accesible la información para ser procesada en cualquier plataforma, de proveer información acerca de los recursos en Internet y de los sistemas que los usan; surgió la idea de crear un marco para representar la información con un mínimo de restricciones, en forma flexible, que pueda ser usado en aplicaciones independientes, donde los formatos de diseño individual deben ser directos y fáciles de entender.

Dicho marco se denomina RDF (Resource Description Framework) y tiene como fin proporcionar un modelo de datos simple, una semántica formal con posibilidad de inferencia, haciendo uso de un vocabulario basado en URI, con sintaxis XML, y permitiendo construir sentencias sobre los recursos.

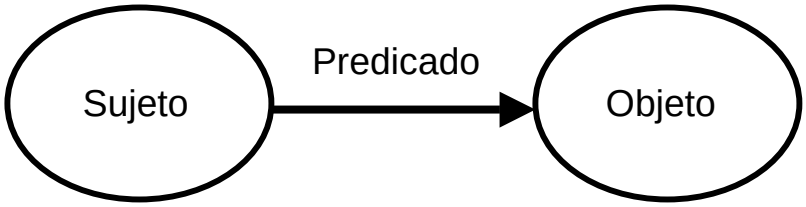
RDF es un lenguaje afirmativo usado para expresar afirmaciones, por medio de un vocabulario formal, especificado a través de RDFS, para tener acceso a los recursos de la web, proporcionando un fundamento básico para lenguajes afirmativos de propósito similar [37].

Es conveniente tener claro los siguientes conceptos, para entender RDF [53]:

Grafo de modelo de datos: Es un conjunto conformado por un sujeto, un predicado y un objeto, que se representa por medio de dos nodos (sujeto y objeto)

y un arco (predicado) dirigido, como aparece en la figura Nº 7. Un ***grafo RDF*** es un conjunto de tripletas RDF, cuyo conjunto de nodos representa el conjunto de sujetos y objetos.

Figura Nº 7. Grafo del modelo de datos



Una afirmación RDF indica que hay una relación entre el sujeto y el objeto, que se describe a través del predicado. El sujeto representa la entidad sobre la cual se está hablando en la afirmación, el predicado identifica la propiedad o característica que califica al sujeto, y la parte que representa el valor de dicha propiedad es el objeto. Cada uno de estos elementos de la tripleta se representan en función de Referencias URI, nodos en blanco y literales, de la siguiente manera:

El sujeto puede ser una referencia URI o un nodo en blanco
El predicado es una referencia URI
El objeto puede ser una referencia URI, un literal o un nodo en blanco

Teniendo en cuenta que, *una **referencia URI** e*s una cadena escrita en UNICODE que no contiene caracteres de control, y que produce una secuencia de caracteres válidos URI representado un URI absoluto. Un ***nodo en blanco*** es un nodo único que puede ser usado en una o más sentencias RDF, pero que no tiene un nombre intrínseco, se utiliza para establecer relaciones n-arias, y para construir sentencias a cerca de recursos que no tienen una URI, pero que están descritas en términos de relaciones con otros recursos que si la tienen. Y un ***literal*** es una cadena utilizada para identificar valores como nombres y fechas por el significado de su representación léxica.

Los literales pueden ser planos o tipificados. Siendo los primeros una combinación de cadenas y etiquetas de lenguajes, que pueden ser usados como texto plano en un lenguaje natural. Y los segundos son combinaciones de cadenas con tipos de datos URI, que denotan el miembro del espacio de valor del tipo de dato identificado al aplicar la conversión léxico – valor de una cadena literal.

Un ***tipo de dato*** se identifica por uno más referencias URI, consiste en un *espacio léxico* (Conjunto de cadenas Unicode), un espacio de valor y una representación

de la conversión del léxico al valor; como se indica en el siguiente gráfico, por medio de esquema XML el tipo de dato xsd:bolean.

Value Space	{T, F}
Lexical Space	{"0", "1", "true", "false"}
Lexical-to-Value Mapping	{<"true", T>, <"1", T>, <"0", F>, <"false", F>}

En la representación de conversión cada miembro del espacio léxico se relaciona con un único miembro del espacio de valores, y cada miembro del espacio de valores puede estar relacionado con cero, uno o varios miembros del espacio léxico. Los tipos de datos son utilizados para representar valores como enteros, números de punto flotante y fechas. RDF no provee un mecanismo para definir nuevos tipos de datos, en cambio utiliza esquemas XML que provee mecanismos para éste fin.

Un ***hecho simple*** indica una relación simple entre dos cosas. Cada hecho puede ser representado como una tripleta RDF, en la cual el predicado denomina la relación, y el sujeto y el objeto denotan las dos cosas. Una representación familiar de un hecho puede ser una fila de la tabla de una base de datos relacional. Un hecho complejo puede ser expresado en RDF usando la conjunción de relaciones binarias simples. Se debe tener en cuenta que RDF no posee mecanismos para expresar negaciones ni disyunciones. Por medio del uso del vocabulario extensible basado en URI, RDF permite expresar hechos acerca de sujetos arbitrarios. Un URI puede estar construido por muchas cosas que pueden ser nombradas, así como hechos acerca de cada una de esas cosas.

El *Modelo Formal* de RDF [61] se presenta en términos de tres conjuntos: Recursos, Literales y Sentencias, y un subconjunto de Recursos denominado Propiedades. Para representar una grupo de valores, RDF define tres clases de contenedores: listas ordenadas llamadas Secuencias, listas desordenadas llamadas Bolsas (Bags), y listas que representan una alternativa para un solo valor de una propiedad, denominadas Alternativas.

El *Modelo Sintáctico* [8] esta dado por medio de sentencias RDF que se pueden representar a través de un grafo (Grafo de modelo de datos), de una tripleta o haciendo uso de sintaxis XML para RDF. Esta última se especifica sintácticamente como una gramática sobre un alfabeto de símbolos, denominados eventos. Existen nueve tipos de eventos, cada uno de los cuales tiene sus respectivos valores: Raíz, Elemento, Elemento final, atributo, texto, Referencia URI, Identificador Nodo Blanco, Literal plano, Literal tipificado. Una secuencia de eventos se deriva de un documento XML ordenado, donde cada elemento se transforma para generar un árbol de eventos con valores. El contenido del evento

es una expresión que combina los diferentes tipos de eventos, aplicando anotaciones de gramática general, de gramática de combinación de eventos, y de gramática de acciones. Las reglas de producción de la gramática RDF/XML se explican en detalle en el documento “RDF/XML Syntax Specification” [8]

La intuición básica del *modelo teórico de la semántica* [66] radica en el hecho de que se hace una afirmación sobre el mundo: Esta es otra forma de decir que el mundo está, de hecho, arreglado para ser una interpretación que hace la sentencia verdadera. En otras palabras, una afirmación para decir una restricción sobre las posibles formas en que el mundo puede ser. Teniendo en cuenta que no puede presumir que cualquier afirmación contenga suficiente información para especificar una única interpretación. Por tanto, entre más grande sea el grafo RDF, más dice acerca del mundo y menor es el conjunto de interpretaciones que una afirmación del grafo permite que sean verdaderas. Todas las interpretaciones pueden ser relativas a un conjunto de nombres, denominado vocabulario de la interpretación. Algunas pueden asignar un significado especial a lo símbolos de un vocabulario en particular.

La principal característica semántica de los literales consiste en que su significado es determinado ampliamente por la forma en que la cadena los contiene. Los literales planos sin una referencia URI, se interpretan siempre como referencias a ellos mismos. En el caso de los literales tipificados la especificación del significado depende del comienzo disponible para tener acceso a la información que es externa al mismo RDF [65].

Actualmente existen *Lenguajes de consulta* como RDQL, Triple, SeRQL, Versa, N3, RQL, RDFQL, RxPath, Algea, Squish, entre otros, que permiten realizar consultas sobre documentos RDF, pero sólo hasta el 12 de octubre de 2004 el Consorcio W3C presentó SPARQL [76] como estándar para los lenguajes de consulta para datos RDF, con múltiples implementaciones, ofreciendo a desarrolladores y usuarios un camino fácil para escribir y manipular consultas y sus resultados, a través de un amplio rango de información.

SPARQL es un Lenguaje de Consulta para acceder grafos RDF, que provee facilidades para:

- Extraer información en forma de URIs, bNodes, texto plano y literales tipificados
- Extraer subgrafos RDF
- Construir nuevos grafos RDF a partir de la información de otros grafos consultados.

Peter Haase en su documento “A Comparison of RDF Query Languages” [36], presenta el resultado de una interesante evaluación que hace sobre algunos

lenguajes de consulta, que permite analizar las características relevantes de dichos lenguajes. En su informe concluye que RQL, RDQL y SeRQL son muy similares a SQL, mientras que Triple y N3 comparten características de lenguajes de reglas. El estilo de Versa es un poco diferente, ya que las consultas presentan directamente operadores del árbol. Sin embargo en éste documento no se incluye SPARQL.

2.2.3 Ontología

La web se ha convertido en una enciclopedia universal del conocimiento humano, representado en una gran cantidad de documentos estáticos, y páginas generadas dinámicamente a partir de contenidos de bases de datos. El inconveniente surge en el momento de manipular dicha información, debido a que se hace necesario alcanzar un entendimiento entre las partes que intervienen en la construcción y explotación de la web: usuarios, desarrolladores y programas de diverso perfil. Como solución se plantea el uso de ontologías.

El sentido filosófico el término ontología hace referencia a la esencia misma del ser, a su existencia, sin embargo, en el contexto de la Inteligencia Artificial, “Gruber define ontología como “a formal explicit specification of a shared conceptualization” [Gruber 1993]. Una ontología es una jerarquía de conceptos con atributos y relaciones, que define una terminología consensuada para definir redes semánticas de unidades de información interrelacionadas. Una ontología proporciona un vocabulario de clases y relaciones para describir un dominio, poniendo el acento en la compartición del conocimiento y el consenso en la representación de éste. Por ejemplo, una ontología sobre arte podría incluir clases como Pintor, Cuadro, Estilo o Museo, y relaciones como autor de un cuadro, pintores pertenecientes a un estilo artístico u obras localizadas en un museo” [21].

Lo que permite concluir que en el contexto del conocimiento compartido, el término ontología significa especificación de una conceptualización [34]: abstracción o vista simplificada del mundo que se representa para algún propósito. Por lo tanto, una ontología es una descripción de conceptos y relaciones que pueden existir para un agente o una comunidad de agentes.

Componentes

Con el fin de entender mejor el concepto de ontología, se hace necesario identifican y comprender cada uno de los componentes que la conforman, los cuales se relacionan a continuación [34]:

Conceptos: son las ideas básicas que se intentan formalizar. Los conceptos pueden ser clases de objetos, métodos, planes, estrategias, procesos de razonamiento, etc.

Relaciones: representan la interacción y enlace entre los conceptos del dominio. Suelen formar la taxonomía del dominio. Por ejemplo: subclase-de, parte-de...

***Funciones*:** son un tipo concreto de relación donde se identifica un elemento mediante el cálculo de una función que considera varios elementos de la ontología. Por ejemplo, pueden aparecer funciones como categorizar-clase, asignar-fecha,

***Instancias*:** se utilizan para representar objetos determinados de un concepto.

***Axiomas*:** son teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología. Por ejemplo: "Si A y B son de la clase C, entonces A no es subclase de B". Permiten junto con la herencia de conceptos inferir conocimiento que no esté indicado explícitamente en la taxonomía de conceptos.

Características

Algunas de las características más representativas de las ontologías son:

***Pueden existir ontologías múltiples*:** El propósito de una ontología es hacer explícito algún punto de vista, por lo que, a veces, será preciso combinar dos o más ontologías. Cada ontología va a introducir conceptualizaciones específicas.

***Se pueden identificar niveles de abstracción de las ontologías*:** Estos niveles de generalización o abstracción dan una topología de ontologías. La idea es caracterizar una red de ontologías usando multiplicidad y abstracción, y puesto que no se puede aspirar a tener una descripción completa del mundo, podemos pensar en una estrategia de construcción gradual de abajo hacia arriba.

***Multiplicidad de la representación*:** Un concepto puede ser representado de muchas formas, por lo que pueden coexistir múltiples representaciones de un mismo concepto.

***Mapeo de ontologías*:** Establecer relaciones entre los elementos de una o más ontologías, para definir conexiones, especializaciones, generalizaciones, etc.

Clasificación

En su artículo "*Using Explicit Ontologies in KBS Development*", publicado en la revista "*International Journal of Human and Computer Studies*", los autores Van

Heijst, G., Schereiber A.T. y Wielinga B.J. 1996, presentan la siguiente clasificación:

***Ontologías terminológicas*:** Especifican los términos que son usados para representar el conocimiento en el universo del discurso. Suelen ser usadas para unificar vocabulario en un campo determinado.

***Ontologías de información*:** Especifican la estructura de almacenamiento de bases de datos. Ofrecen un marco para el almacenamiento estandarizado de información.

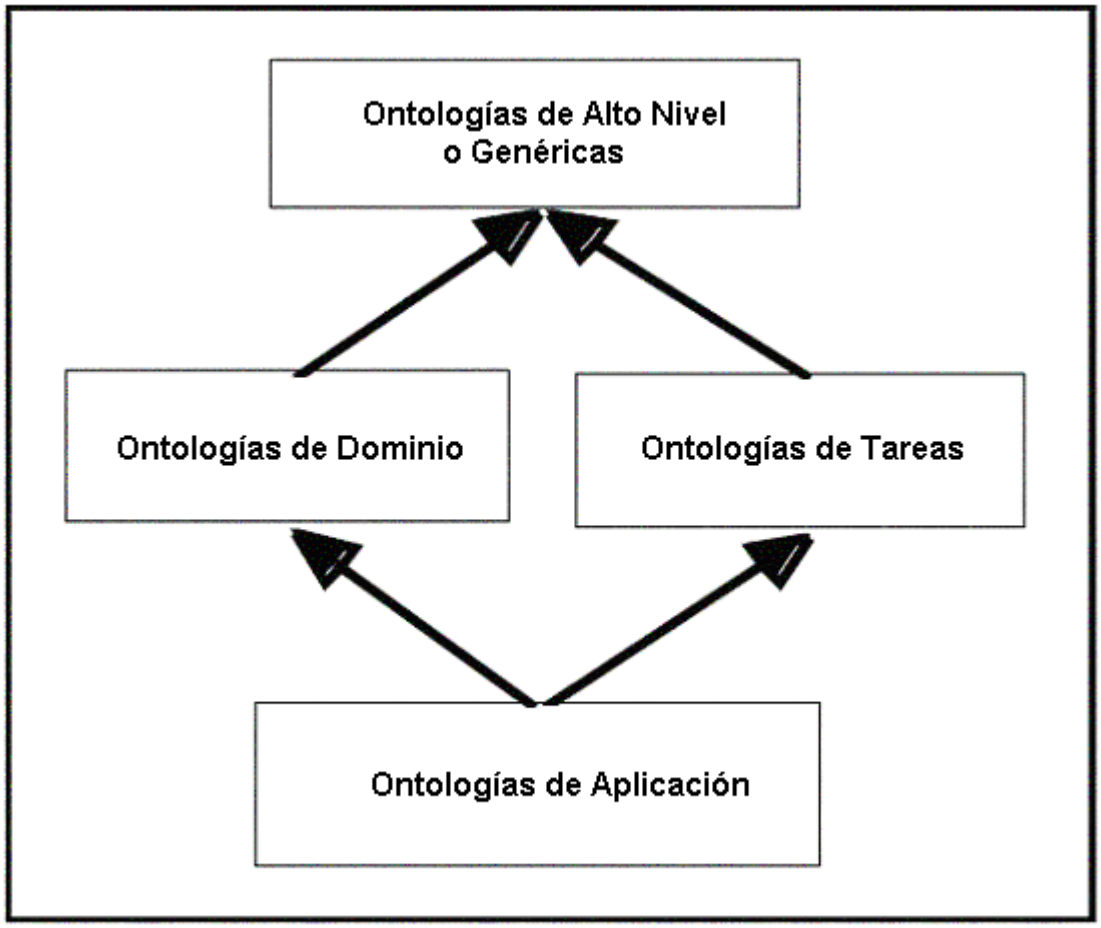
***Ontologías de modelado de conocimiento*:** Especifican conceptualizaciones del conocimiento. Contienen una rica estructura interna y suelen estar ajustadas al uso particular del conocimiento que describen.

Esta clasificación es analizada por Nicola Guarino (1997) en su documento "Understanding, building and using Ontologies" [35], en el cual indica que en dicha clasificación se distinguen dos dimensiones: La cantidad y tipo de estructuras de la conceptualización (las dos primeras) y el sujeto de la conceptualización (la última), concluyendo que la primera no es clara debido a que no se establece una diferencia entre el dominio del conocimiento y el dominio de la ontología, además afirma que la estructura lógica de una base de datos, corresponde al nivel simbólico y por lo tanto no es una ontología.

Por su parte sugiere más bien tener el cuenta el grado de detalle usado para caracterizar una conceptualización, lo que permite distinguir entre ***ontologías documentales*** **(off-line)** y ***ontologías compartidas*** **(on-line),** entendidas así:

***Ontologías documentales*:** Precisa de un lenguaje de alta expresividad y tiene un gran número de axiomas. Deberían ser usadas off-line y solamente para referencia.

Figura Nº 8: Tipos de ontologías



Fuente: http://www.loa-cnr.it/Papers/FOIS98.pdf

***Ontologías compartidas*:** Tiene un número mínimo de axiomas y su objetivo es ser compartida por usuarios que concurren sobre una determinada visión del mundo. Tienen una mayor capacidad de ser compartidas y deberían ser utilizadas on-line para dar soporte en funcionalidad de sistemas de información.

Respecto a la segunda dimensión, Nicola Guarino afirma que es mucho más clara, ya que se hace de acuerdo con su dependencia y relación con una tarea específica, y la detalla en su documento "Formal Ontology and Information Systems" (1998) de la siguiente manera [35]:

***Ontologías de Alto Nivel o Genéricas*:** Describen conceptos muy generales, como espacio, tiempo, materia, objeto, evento, etc, los cuales son independientes de un problema particular o dominio, lo que indica que es razonable, al menos en teoría, tener ontologías de alto nivel para grandes comunidades de usuarios.

***Ontologías de Dominio*:** Describen un vocabulario relacionado con un dominio genérico, para especializar los términos introducidos en la ontología de alto nivel.

***Ontologías de Tareas o de Técnicas básicas*:** Describen una tarea, actividad o artefacto, para especializar los términos introducidos en la ontología de alto nivel

***Ontologías de Aplicación*:** Describen conceptos que dependen tanto de un dominio específico como de una tarea específica y, generalmente son una especialización de ambas. Estos conceptos a menudo corresponden a los roles que desempeñan las entidades en el dominio mientras realizan una determinada actividad.

Campos de aplicación

Actualmente las ontologías son utilizadas en campos tan diversos como [20]: Ingeniería del conocimiento, Gestión del conocimiento, Procesamiento del lenguaje natural, Sistemas de información cooperativos, Integración inteligente de información, Recuperación de información y Comercio electrónico, enseñanza, entre otros; donde son utilizadas en calidad de repositorios de conocimientos e información para la organización, tanto de tipo corporativo como científico; o como herramienta para la adquisición de información, en situaciones en las que un equipo de trabajo las utiliza como soporte común para la organización del dominio; aprovechando el hecho de que permite la reutilización del conocimiento existente en nuevos sistemas, así como también, la construcción de lenguajes de representación del conocimiento, junto a la formalización del cálculo que tenga lugar entre los términos.

En Documentación se puede considerar de especial interés la aplicación, por un lado, de ontologías terminológicas que unificarían la terminología de cada concepto y las relaciones entre ellos; y por otro, las ontologías de información que unificarían las estructuras de almacenamiento de forma que pudieran ser reutilizadas por varias aplicaciones informáticas que utilicen la misma fuente de información.

El uso de Ontologías en combinación con agentes inteligentes permite mejorar la eficacia de los procesos de indización, recuperación y divulgación de la información web, permitiendo encontrar páginas especializadas pertenecientes a un dominio específico. Tal es el caso de servicios como: llamadas de trabajo para eventos científicos (call for papers), páginas como listas de publicaciones, páginas de universidades, etc. El agente ejecuta una meta-búsqueda en las bases de índices de varios mecanismos de búsqueda y, seguidamente, clasifica las páginas recuperadas, utilizando las ontologías de clases que las indexará por palabras clave. Son también utilizadas las reglas que describen las características específicas de las páginas de este dominio. Por ejemplo: detalles de estructura e información de la página.

También se utiliza la estrategia del desarrollo de plataformas, para la construcción y manipulación de ontologías, que representan dinámicamente una estructura semántica de Bases de Índices del mecanismo de búsqueda asociado. Un usuario selecciona un tema en el que se inserta su búsqueda (o contexto de búsqueda) a partir de una ontología ofrecida por la herramienta. Un ejemplo de esta plataforma es Knowledge and Information Management Plataform KIM

2.2.4 Lenguajes para la Construcción de Ontologías

La Web Semántica ha promovido el desarrollo de lenguajes estándares para la representación del conocimiento en su máxima expresividad. Estándares que permiten unificar esquemas de intercambio de información entre aplicaciones, así como contenidos semánticos por medio de ontologías. Las ontologías formalizan el conocimiento de forma consensuada y reutilizable. En este sentido se han definido lenguajes para la especificación y consulta de ontologías basados en paradigmas formales (Flogic, Knowledge Interchange Format KIF, Ontology Inference Layer OIL, Darpa Agent Markage Language DAML, DAML+ OIL [40]) y en recomendaciones desarrolladas por el grupo W3C (RDFS, Ontology Exchange Language XOL, Web Ontology Language OWL [38], Simple HTML Ontology Extension SHOE). A continuación se presentan los aspectos relevantes sobre los lenguajes DAML + OIL y OWL, por ser los que han tenido un mayor auge [54].

DAML+OIL

Es un lenguaje de marcado semántico para recursos Web. Ha sido creado bajo el estándar RDF y RDFS y extiende estos lenguajes para enriquecer las primitivas de modelado de ontologías. Permite describir la estructura de un dominio en términos de sus clases y propiedades; cuando un recurso r es una instancia de una clase C, se dice que r tiene el tipo C. Una base de conocimientos DAML + OIL es una colección de tripletas RDF, para lo cual el lenguaje prescribe un significado especifico para cada tripleta que usa en su vocabulario.

Desde el punto de vista formal, DAML + OIL puede ser visto como el equivalente a muchas descripciones lógicas (DL) expresivas, con una ontología DAML + OIL correspondiente a una terminología DL. Como en DL, las clases DAML + OIL pueden ser nombres (URIs) o expresiones, y una variedad de constructores para crear expresiones de clases. El poder expresivo del lenguaje esta determinado por las clases (y propiedades) constructoras soportadas, y por los tipos de axiomas que permite [40].

DAML a diferencia de RDF no es un modelo de datos, sino un lenguaje de esquema que puede ser usado para describir y restringir datos después del

modelo RDF. Dicho de otra forma DAML es una extensión del lenguaje RDFS, cuyo característica principal consiste en permitir describir datos RDF, agregándole valor semántico a los datos. En general DAML agrega a los esquemas RDF formas adicionales para restringir los valores permitidos de las propiedades, y que propiedades puede tener una clase. Además presenta las siguientes características que pueden ser muy útiles para el software genérico [33]:

- Permite identificar si dos propiedades RDF de diferentes esquemas son de hecho la misma propiedad.
- Permite identificar si una propiedad es inversa a otra propiedad
- Permite aplicar transitividad a las propiedades de las clases.

Web Ontology Language OWL

El lenguaje OWL ha sido creado con el fin de compartir ontologías, permitir su evolución, facilitando los posibles cambios que se puedan presentar, teniendo en cuenta los diferentes proveedores de datos. Además de permitir la integración de ontologías establecidas, la detección de inconsistencias en ontologías y sus instancias de uso; proveer un balance entre la entre la expresividad y la escalabilidad cuando se crean ontologías, eliminado la complejidad, al centrarse en la facilidad de uso, permitiendo compatibilidad con otros estándares; y soportar el desarrollo de ontologías multilingües, presentando diferentes puntos de vistas propios de cada cultura.

Este lenguaje puede ser usado en las siguientes actividades [38]: Construcción de portales web, facilitando la definición y captura de metadatos. Construcción de colecciones multimedia, permitiendo anotaciones semánticas de imágenes, audio y otros objetos no textuales. Administración de sitios web corporativos, para construir ontologías que son usadas como índices de documentos, mejorando su recuperación por contenido. Diseño de documentos, al construir ontologías que pueden ser usadas como modelos de información que permiten la exploración de espacios de información en términos de ítems, que pueden ser representados, asociados y caracterizados. Definición de servicios web para ser usados por medio de agentes, al permitir la construcción de ontologías que definen el dominio del servicio web. En la computación ubicua, ya que OWL puede ser usado para describir las características de los dispositivos, la forma en que se acceda cada uno de ellos, y las políticas que establece el propietario para su uso.

OWL facilita la interoperabilidad entre computadores, de contenidos web representados con XML, RDF, RDFS agregando vocabularios adicionales con semántica formal. Tiene tres sublenguajes [68]:

- OWL Lite: Diseñado para usuarios que requieren satisfacer necesidades primarias. Permite clasificación jerárquica y restricciones simples.

- OWL DL: Diseñado para usuarios que requieren una expresividad máxima mientras garantiza que las conclusiones son procesadas completamente, y todos los procesos terminan en un tiempo finito.
- OWL Full: Diseñado para usuarios que requieren máxima expresividad sin garantía computacional.

Cada uno de estos sublenguajes es una extensión de su predecesor, y cumplen con las siguientes relaciones:

- Cada ontología correcta en OWL Lite también es correcta en OWL DL.
- Cada ontología correcta en OWL DL también es correcta en OWL Full.
- Cada conclusión válida en OWL Lite también es válida en OWL DL.
- Cada conclusión válida en OWL DL también es válida en OWL Full.

2.2.5 Lenguaje de reglas para la Web Semántica

La formalización adecuada de reglas de inferencia ha sido identificada como un aspecto de diseño de vital importancia para la Web Semántica, por su relación con los sistemas basados en conocimiento y las aplicaciones basadas en agentes inteligentes. Esto llevó a Iniciativas que motivaron el desarrollo de lenguajes que permiten representar reglas en XML para realizar razonamiento hacia atrás, hacia adelante, reescritura de información, y otras tareas de transformación.

Rule Markup Language (RuleML)

RuleML abarca una jerarquía de reglas, que incluyen reglas de reacción (reglas de evento – condición – acción), reglas de transformación (reglas de función – ecuación), reglas de derivación (implicación – inferencia), también especialización para hechos (reglas de derivación “de menos premisas”) y consultas (reglas de derivación de “menos conclusiones”). De igual manera integra restricciones (reglas de consistencia – mantenimiento) [15].

La jerarquía RuleML de reglas generales se divide en dos categorías directas de reglas de reacción y reglas de transformación. En el siguiente nivel las reglas de transformación se especializan para las subcategorías de reglas de derivación. Entonces las reglas de derivación tienen características de subcategorías, denominadas hechos y consultas. Finalmente las consultas especializadas para restricciones de integridad.

Semantic Web Rule Language (SWRL)

Es un leguaje de reglas para la Web Semántica, basado en la combinación de OWL DL y OWL Lite con el sublenguaje Unary/Binary Datalog RuleML. Lo que le permite extender un conjunto de axiomas para incluir reglas y combinarlas con bases de conocimientos en OWL [39].

El objetivo de las reglas es el de formar una implicación entre un antecedente (cuerpo) y un consecuente (cabeza). El significado previsto se puede leer como: siempre que las condiciones especificadas en el antecedente se mantengan, entonces las condiciones especificadas en el consecuente deben también mantenerse.

Ambos, el antecedente y el consecuente consisten en cero a más átomos. Un antecedente vacío es tratado como una verdad trivial (o sea que se satisface para toda interpretación), mientras que el consecuente también debe satisfacerse para toda interpretación. Un consecuente vacío es tratado como una falsedad trivial (o sea que no se satisface para ninguna interpretación), por lo tanto el antecedente tampoco satisface ninguna interpretación. Múltiples átomos son tratados como una conjunción.

Los átomos en estas reglas pueden encontrarse de la forma C(x), P(x,y), sameAs(x,y), differentFrom(x,y). Donde C es una descripción OWL, P es una propiedad OWL, y x,y son variables, valores individuales OWL o datos OWL [41].

2.2.6 Herramientas

Actualmente para el desarrollo de aplicaciones web semánticas, se cuenta con interfaces de programación entre aplicaciones como Jena [50], JRDF[51], y plataformas como Sesame [72], SNOBASE [46] y KIM. Además se tienen editores de ontologías que facilitan su construcción y definición, tales como Swoop, Protege y OilEd, cada uno de los cuales se describirá brevemente a continuación.

Interfaces de Programación entre Aplicaciones (API)

El objetivo principal de las interfaces de programación que se presentan a continuación, es proporcionar un mecanismo que facilite el proceso de construcción y mantenimiento de aplicaciones Web semánticas [98].

Jena

Es una plataforma Java para la creación de aplicaciones web semánticas. Provee un entorno de programación para RDF, RDFS y OWL, incluyendo un mecanismo de inferencia basado en reglas.
Esta conformado por:

- Un API RDF
- Mecanismo de lectura y escritura RDF en RDF/XML, N3 y N – Tripletas
- Un API OWL
- Almacenamiento en memoria y persistente
- Lenguaje de consulta para RDF (RDQL)

Java RDF JRDF

Intenta crear un conjunto de APIs estándar e implementaciones basadas en RDF, haciendo uso de Java, centrándose en un alto grado de modularidad y siguiendo los estándares convenidos de Java. Esta conformado por:

- Un API para gráficos
- Mecanismo para crear y manipular objetos gráficos
- Manejo de transacciones
- Manejo de eventos
- Manejo de consultas
- Tipos de Datos RDF
- Mecanismo de Seguridad
- Mecanismo de inferencia

Sesame

Es una plataforma Java de código abierto para guardar, consultar y hacer razonamiento con RDF y RDFS. Pude ser usada como una base de datos para RDF y RDFS, o como una librería para aplicaciones que necesitan trabajar internamente con RDF. Provee las herramientas necesarias para interpretar, consultar y almacenar la información, embebida en la propia aplicación, separada en una base de datos, o en un servidor remoto.

Semantic Network Ontology Base SNOBASE

Es un Sistema de administración de Ontologías, similar a los sistemas de administración de bases de datos, que permite almacenar, procesar, consultar y manipular ontologías. Provee mecanismos para cargar ontologías vía Internet, y para crear y modificar ontologías locales. Internamente el sistema usa mecanismos de inferencia y de almacenamiento persistente de ontologías, que pueden estar escritas en lenguajes como RDF, DAML+OIL, DAML-S, OWL. y OWL-S

SnoBase proporciona un sistema de conexión denominado JOBC (Java Ontology Base Connectivity) similar a JDBC (Java Data Base Connectivity), que facilita el manejo de las ontologías a partir de una aplicación específica.

Knowledge and Information Management KIM

Esta plataforma provee una infraestructura para la administración de la información y el conocimiento, servicios para la Anotación semántica automática, Indización, recuperación, consulta y exploración del conocimiento formal [55].

Las aplicaciones más directas de KIM son [54]:

- Generación de matadatos para Web semántica, permitiendo hipervínculos, visualización avanzada y navegación.
- Administración del conocimiento, resaltando la eficiencia de la existencia de índices, recuperación, clasificación y aplicación de filtrado.

KIM presenta tres perfiles:

- Plataforma similar en propósito al uso de los Sistemas de Administración de Bases de Datos Relacionales
- Extensión de los productos KIM
- Servicio Web, siempre y cuando el cliente remoto cumpla con los requerimientos mínimos de instalación y mantenimiento.

Editores de Ontologías

Los Editores de ontologías son las herramientas que permiten la codificación de una determinada ontología basada en un lenguaje específico. A continuación se presentan algunos de los editores más representativos que se utilizan actualmente.

Swoop

Es un navegador simple y escalable de ontologías diseñadas con OWL. Usa URIs que son inicialmente identificadas como ontologías, clases, propiedades y entidades, que soportan navegación de hipertexto a través de ontologías. El equivalente a una página Web es una ontología OWL que contiene entidades (clases, propiedades, individualidades) análogas al contenido HTML. Una página antológica puede presentarse en varios formatos, desde una vista lógica y concisa, hasta una representación RDF/XML.

Ha sido diseñado siguiendo las recomendaciones de la especificación OWL del Consorcio W3C, por lo que contiene validaciones especiales, ofrece varias vistas

de las presentaciones de sintaxis (sintaxis abstracta, N3, etc), y permite razonamiento por medio de un mecanismo de inferencia OWL; además provee un entorno de múltiples ontologías, que permite comparar, editar y combinar las entidades de dichas ontologías. También presenta una interface amigable que facilita los cambios y respectivas actualizaciones de las ontologías [52].

Protégé

Es una herramienta Java de código libre, con arquitectura extensible para la creación y personalización de aplicaciones basadas en conocimiento, que además permite editar ontologías y bases de conocimiento. Su funcionalidad se presenta desde tres puntos de vista [92]:

- Como una herramienta que permite al usuario: construir y personalizar ontologías, definiendo clases, jerarquía de clases, restricciones y relaciones entre las clases y las propiedades de estas relaciones. Registrar datos y personalizar los formularios de ingreso de dichos datos.
- Como una plataforma que se puede extender con gráficas, tablas, diagramas, componentes de animación para tener acceso a otros sistemas basados en conocimiento.
- Como una librería que pueden utilizar otras aplicaciones para tener acceso a bases de conocimiento.

OilEd

Es un editor de ontologías que permite al usuario crear ontologías en DAML + OIL. Inicialmente el proyecto se centró en demostrar el uso y estimular el interés or el lenguaje OIL. Actualmente está disponible como un proyecto de código abierto bajo licencia GPL [7].

2.3 ESTADO DEL ARTE DE LOS LENGUAJES DE MARCADO PARA EL DESARROLLO DE APLICACIONES MULTIMODALES

En el contexto del presente documento, se entiende por aplicación multimodal, aquella que permite a un usuario comunicarse con el sistema usando diferentes modalidades como: voz (lenguaje natural), gestos, escritura, audio, imágenes estáticas o en movimiento, etc [93]; teniendo en cuenta que el usuario puede ser considerado para operar en un contexto de entrega, o sea, bajo un conjunto de atributos que caracterizan las capacidades de los mecanismos de acceso en términos de perfil de dispositivos, perfil de usuario (identificación, preferencias, patrones de uso) y situaciones. El usuario interactúa con la aplicación en el contexto de una sesión, usando una o más modalidades, que pueden ser

realizadas a través de uno o más dispositivos, Durante la sesión el usuario puede suspender o reanudar la interacción con la aplicación haciendo uso de la misma modalidad o intercambiando diferentes tipos de modalidades. Una sesión es asociada a un contexto, el cual registra las interacciones con el usuario.

Con el fin de facilitar el entendimiento de los lenguajes de marcado que se utilizan en la construcción de aplicaciones multimodales, se ha tenido en cuenta la plataforma de interacción multimodal que propone el Consorcio W3C, para identificarlos y clasificarlos de la siguiente manera:

- Según su modalidad:
    - Entrada por escritura manual: InkML
    - Entrada con Voz y DTMF: XHTML + Voice, VoiceXML, SALT, TalkML, VoxML, SpeechML
    - Salida con audio: SSML (Speech Synthesis Markup Language)
- Según la funcionalidad que representa en la interacción con el usuario final.
    - Construcción de la interface de usuario: XForms
    - Descripción de la interpretación de la entrada de Usuario EMMA (Extensible MultiModal Annotation markup language)
    - Sincronización e integración de elementos multimedia SMIL (Synchronized Multimedia Integration Language)

Adicionalmente, se tiene la especificación de la gramática para el reconocimiento de voz SRGS (Speech Recognition Grammar Specification), que define la sintaxis para representar la gramática que usa el reconocimiento de voz, que el desarrollador especifica con palabras y patrones de palabras que puede escuchar el reconocedor de voz.

De igual manera, se encuentra SVG (Scalable Vector Graphics), como el lenguaje que permite describir gráficos en dos dimensiones y aplicaciones gráficas en XML, utilizado para complementar la interacción gráfica en el momento de la salida, en combinación con señales de audio (archivos de audio y voz digitalizada).

A continuación se presenta una descripción del estado del arte de los lenguajes que forman parte de la clasificación mencionada, representada gráficamente en la figura Nº 9.

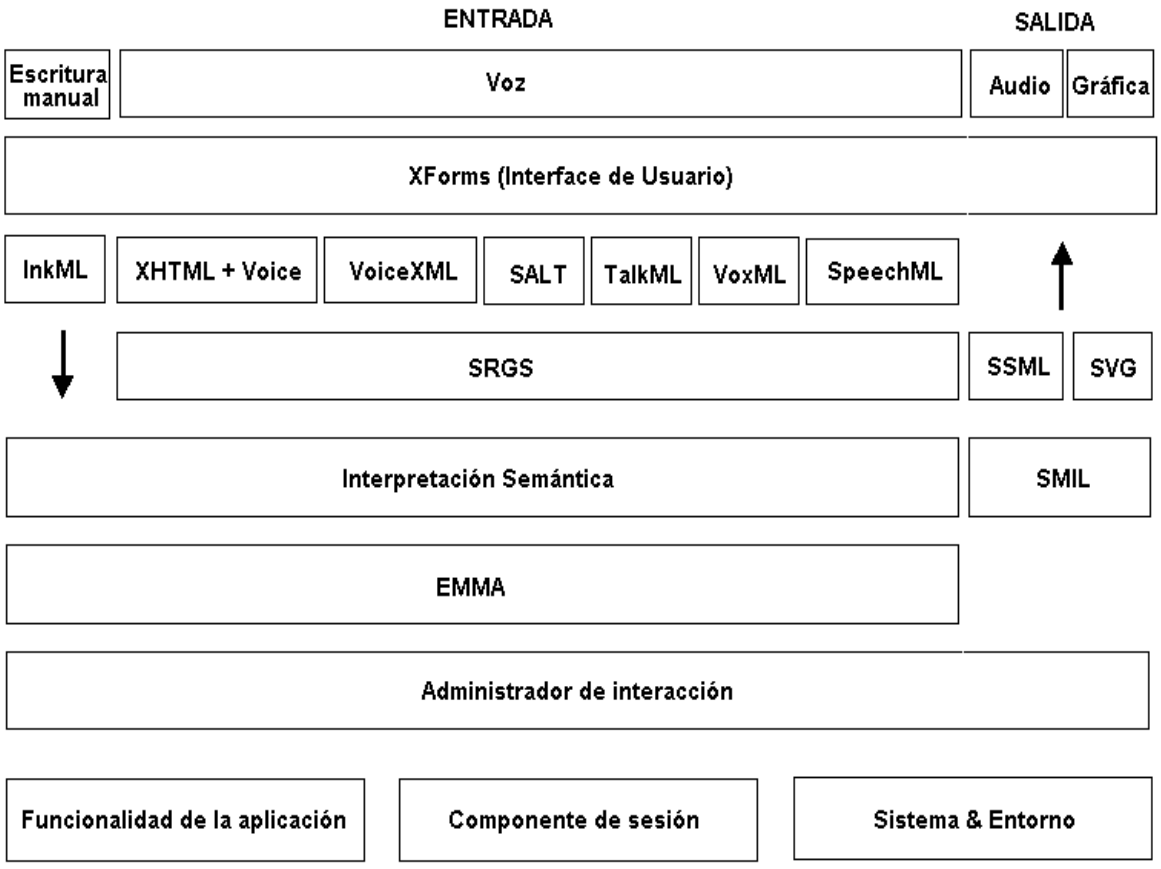
2.3.1 InkML (Ink Markup Language)

Debido a que la escritura manual es un mecanismo muy utilizado para el registro de la información, se han creado dispositivos electrónicos que permiten interactuar

al usuario a través de éste tipo de modalidad. Sin embargo, los proveedores de software y hardware han desarrollado sus propios estándares y formatos, que incluyen restricciones excluyentes que impiden la captura, transmisión y procesamiento de la tinta digital, a través de dispositivos heterogéneos. Como alternativa a dicho problema, se presenta el Lenguaje de Marcado InkML, proporcionando un formato de datos simple y neutral a la plataforma, para permitir el intercambio de tinta digital entre aplicaciones.

Este lenguaje permite una completa y cuidadosa representación de tinta digital (representación electrónica del movimiento del lápiz, presión, y otras características de entradas por escritura manual usando dispositivos digitales [10]), que además de registrar la posición del lápiz respecto al tiempo, también posibilita grabar la información acerca de las características del dispositivo, y detalles del comportamiento dinámico para soportar aplicaciones como reconocimiento y autenticación de escritura manual.

Figura Nº 9. Clasificación de los Lenguajes de Marcado para la interacción Multimodal.



Además, su aplicabilidad se extiende a todas aquellas situaciones en la que el lápiz puede ser usado como una forma natural y conveniente de entrada, por ejemplo: En dispositivos electrónicos que por su tamaño carecen de teclado, para llenar formularios, el envío y recepción de mensajes escritos a mano por medio de dispositivos móviles, a través de una red inalámbrica; y en sistemas flexibles y robustos que permitan la interacción multimodal.

InkML es un formato de datos en XML, usado para representar datos de tinta digital, para entradas con lápiz electrónico, como parte de un sistema multimodal. En el contexto de la plataforma de interacción multimodal Consorcio W3C, el lenguaje provee un formato para [32]:

- Transferir datos de tinta digital entre dispositivos y componentes de software
- Almacenar trazos de entrada manual para:
    - Reconocimiento de entrada manual (incluyendo texto, expresiones matemáticas, fórmulas químicas)
    - Validación de firmas
    - Interpretación de gestos.

La especificación InkML ha sido diseñada por un subgrupo del Grupo de Trabajo de Interacción Multimodal del Consorcio W3C, cuyo primer documento se emitió el 22 de enero de 2003, cuando se publican los requerimientos para el Lenguaje de Marcado de Tinta. Luego, el 6 de agosto del mismo año publican el primer trabajo del borrador para InkML, la segunda parte de éste trabajo se dio a conocer el 23 de febrero de 2004, la tercera y última parte (hasta el momento en que se escribe este documento) fue publicada el 28 de septiembre de 2004

El primer documento se centra en establecer los requerimientos que debe satisfacer el Lenguaje de Marcado, clasificándolos de la siguiente manera [22]: Requerimientos generales: los cuales hacen referencia al ámbito de aplicación, servicio de reconocimiento, y facilidad de uso y adopción. Requerimientos del procesamiento de entrada: enfocados hacia la captura de tinta e identificación de eventos externos. Requerimientos del procesamiento de salida: que indican la necesidad de permitir una amplia variedad de plataformas de salida. Requerimientos de la arquitectura: Definen las características estructurales que debe cumplir el lenguaje, tales como: Reutilización, modularidad, distribución de cargas e integración. Finalmente se plantean los requerimientos de movilidad, multimodalidad y sincronización, que debe satisfacer el Lenguaje de Marcado.

Los demás documentos publicados por el Grupo de Trabajo, tienen como finalidad describir la sintaxis y la semántica del Lenguaje de Marcado InkML. Definen como documento InkML aquel que cumple con las reglas de sintaxis de la especificación InkML, y además es un documento XML bien formado. También establecen el

siguiente conjunto de elementos primitivos, suficientes para todas sus aplicaciones básicas:

<ink>: Es el elemento raíz de cualquier instancia inkML.
<trace>: Es el elemento fundamental de datos. Representa una secuencia de puntos contiguos, en términos de coordenadas de la posición del lápiz.
<traceFormat>: Define el formato de datos para un trazo.
<captureDevice>: Especifica la información del dispositivo usado para capturar la tinta digital.
<brus>: Captura atributos del trazo, como color y ancho.
<traceGroup>: Sirve para agrupar trazos que comparten las mismas características.
<contex>: Permite representar y agrupar la información pertinente como formato del trazo, atributos y lienzo.
<traceRef>: Es un bloque de construcción para etiquetar trazos. Incluye un atributo genérico de contenido que puede ser usado por aplicaciones para describir el nivel básico de categorías de contenidos que los trazos representan (dibujo, texto escrito, etc)
<writerInfo>: Es utilizado para registrar información acerca del escritor.

2.3.2 VoiceXML

VoiceXML es un lenguaje de marcado que, minimiza la interacción cliente servidor especificando múltiples interacciones por documento, separa el código de interacción (VoiceXML) de la lógica del servicio (scripts), promueve la portabilidad de servicios a través de plataformas de implementación, presentando características de lenguaje que soportan diálogos complejos.

Este lenguaje permite la interacción hombre máquina por medio de un sistema de respuesta de voz que incluye: salida de voz digitalizada, salida de archivos de audio, reconocimiento de entrada de voz, reconocimiento de entradas DTMF, control del flujo de diálogo, características de teléfono como transferencia de llamadas y desconexión.

Además posibilita la integración de servicios de voz con servicios de datos, usando el paradigma familiar cliente servidor, teniendo en cuenta que un servicio de voz es visto como una secuencia de interacciones de diálogos entre un usuario y una plataforma de implementación. Los diálogos son generados a partir de un servidor de documentos, que puede ser externo a la plataforma de implementación, y es el responsable de mantener todos los servicios lógicos, el rendimiento de la base de datos, y la corrección de las operaciones del sistema al producir los diálogos. Un documento VoiceXML especifica cada diálogo de interacción para que sea gestionado por el interprete VoiceXML. El usuario realiza entradas, interpretación de diálogos, y envía solicitudes al servidor de

documentos. El servidor de documentos responde con otro documento VoiceXML para continuar la sesión de usuario con otro diálogo.

VoiceXML se crea a partir del proyecto denominado Phone Markup Language (PML), realizado por la compañía AT&T, en el año 1995, planteado con el fin de simplificar el proceso de desarrollo de aplicaciones que requieren reconocimiento de voz, utilizando como base el lenguaje de marcado XML. En 1998 el W3C organiza una conferencia sobre navegadores de voz, donde AT&T junto con Lucent presentan una variante del original PML, mientras que Motorola ha desarrollado VoxML, y la IBM ha creado su propia versión denominada SpeechML. Otras compañías habían construido lenguajes similares para el diseño de diálogos, tales como TalkML de Hewlett-Packard, y VoiceHTML de PipeBeach (actualmente parte integral de HP).

Esto dio origen al Foro VoiceXML conformado el 2 de Marzo de 1999, por AT&T, IBM, Lucent y Motorola [63], cuya misión consistió en definir un lenguaje estándar para el diseño de diálogos, que pudiera ser usado por los desarrolladores para construir aplicaciones de voz. El foro definió como base XML, puesto que es claro que ha sido la dirección en que se ha movido la tecnología, y lanzó la versión 0.9 de VoiceXML el 17 de agosto de 1999. El siguiente año, más exactamente el 7 de marzo, se publica la versión 1.0, siendo registrada en el W3C, como la base para la creación de un nuevo estándar internacional. En el año 2002 se publica la versión 2.0, la cual es registrada como recomendación del el 16 de marzo de 2004. El 28 de julio de ese mismo año se lanza el trabajo del anteproyecto para la versión 2.1 de VoiceXML [67], cuyo principal objetivo es especificar formalmente las características comunes, para asegurar su portabilidad entre plataformas y al mismo tiempo mantener compatibilidad con las versiones anteriores.

VoiceXML ha sido diseñado para crear diálogos de audio que se caracterizan por contar con voz sinterizada, audio digitalizado, reconocimiento de voz y entradas de teclado DTMF (Dual Tone Multi Frequency), grabando las entradas de voz, teléfono, y combinando conversaciones. Su principal objetivo es brindar las ventajas del desarrollo basado en la web y entrega de contenidos para interactuar con aplicaciones que responden con voz, liberando al desarrollador de la administración de recursos y el bajo nivel de programación, y permitiendo utilizar otras funcionalidades que puede proporcionar la red telefónica.

Esta integración de servicios de voz con datos es posible gracias al paradigma cliente servidor, presente en el modelo arquitectónico de VoiceXML, donde el servidor procesa la petición de una aplicación cliente, que viene a través del Interprete de contexto de VoiceXML. El servidor produce un documento VoiceXML como respuesta, el cual es procesado por el interprete de VoiceXML. La plataforma de implementación es controlada por el interprete de contexto de VoiceXML, encargado de tareas como controlar las llamadas que entran, cargar el documento inicial de VoiceXML y responder la llamada, mientras que el Interprete

de VoiceXML conduce el diálogo después de la respuesta. La figura Nº 4 representa la arquitectura VoiceXML.

Dicha arquitectura es coherente con la propuesta para los Sistemas de diálogo basados en voz (Spoken Language Dialog System[1] SLDS), la cual incluye los siguientes componentes: Reconocimiento de voz (Automatic Speech Recognition ASR): para analizar las entradas de voz, distinguiendo unidades lingüísticas como palabras o fonemas. Entendimiento del lenguaje: mecanismo utilizado para determinar el significado de la entrada. Administrador de diálogo: encargado de mantener el flujo de la conversación, historia, contexto, acceso a la base de datos y formulación de respuestas. Base de datos, Generación del Lenguaje: para representar la respuesta en palabras; y digitalizador de voz (Text To Speech TTS), para convertir la señal de salida de audio en voz. Gráficamente la arquitectura de los SLDS se representa en la figura Nº 9.

La implementación de la arquitectura propuesta para los sistemas de diálogo basados en voz se representa a través del diagrama de despliegue de la figura Nº 10, el cual refleja los componentes físicos necesarios para el funcionamiento de una aplicación de voz. Mientras que en la figura Nº 11 se presenta el caso específico de implementación para la arquitectura de VoiceXML.

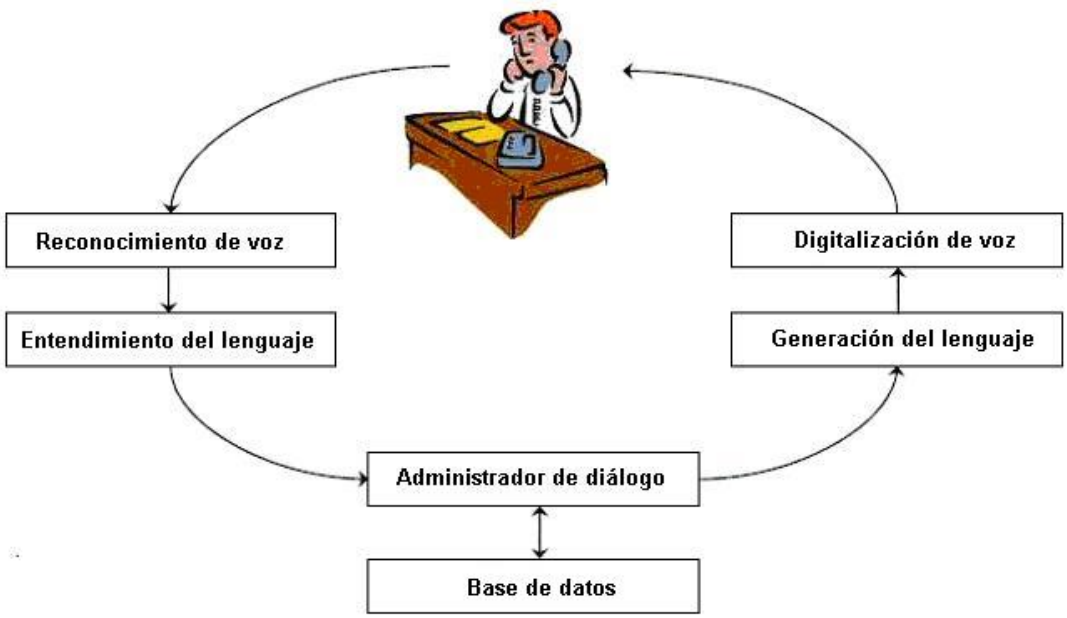
Para cumplir con los objetivos y características previamente anunciadas, el intérprete de VoiceXML requiere de una plataforma de implementación, a nivel de software y hardware, conformada por los siguientes componentes [67]:

Adquisición de documentos: debe existir un mecanismo que soporte el esquema URI “http”, de tal manera que permita cargar un documento, teniendo en cuenta que, en algunas situaciones, la solicitud del documento es generada a partir de la interpretación de un documento VoiceXML, mientras que en otras, puede ser generada por el interprete del contexto, en respuesta a los eventos que se encuentran fuera del rango del lenguaje.

Salida de audio: La plataforma de implementación debe soportar salida de audio, haciendo uso de archivos de audio y voz digitalizada TTS. También debe disponer de secuencias libres TTS y salidas de audio. Si el recurso de salida de audio no se encuentra disponible, se genera un evento de error.

[1] Sistema computacional que permite establecer un diálogo sobre un tema específico, y que típicamente se presenta de dos formas: Para presentar información específica, y para realizar transacciones como compras o reservas.

Figura Nº 10. Arquitectura SLDS



Fuente: http://www.alta.asn.au/events/altss2004/course_notes/ALTSS-Schwitter-VoiceXML.pdf

Entrada de audio: La plataforma de implementación requiere detectar y reportar caracteres y/o entradas de voz simultáneamente, además de identificar el intervalo de duración con un temporizador que es especificado por el documento VoiceXML. Si un recurso de entrada de audio no está disponible, se genera un evento de error.

Otras características con las que debe cumplir la plataforma de entrada son:
Debe reportar caracteres de entrada registrados por el usuario (como DTMF), soportando formas XML y ABNF (the Augmented Backus Naur Form) de la gramática descrita en la especificación SRGS (Speech Recognition Grammar Specification).

Debe permitir la gramática de datos para el reconocimiento de voz dinámicamente, permitiendo usar datos XML y ABNF de la gramática descrita en la especificación SRGS, y otras formas de gramática como JSGF (Java Speech Grammar Format), teniendo en cuenta que algunos elementos de VoiceXML contienen datos sobre la gramática de voz y otros sobre la URI. El reconocedor de voz debe estar dispuesto para acomodarse dinámicamente, y actualizar la entrada de voz escuchada a través cualquier método de la gramática de voz que especifica los datos.

Figura Nº 11. Diagrama de Despliegue para aplicaciones de Voz

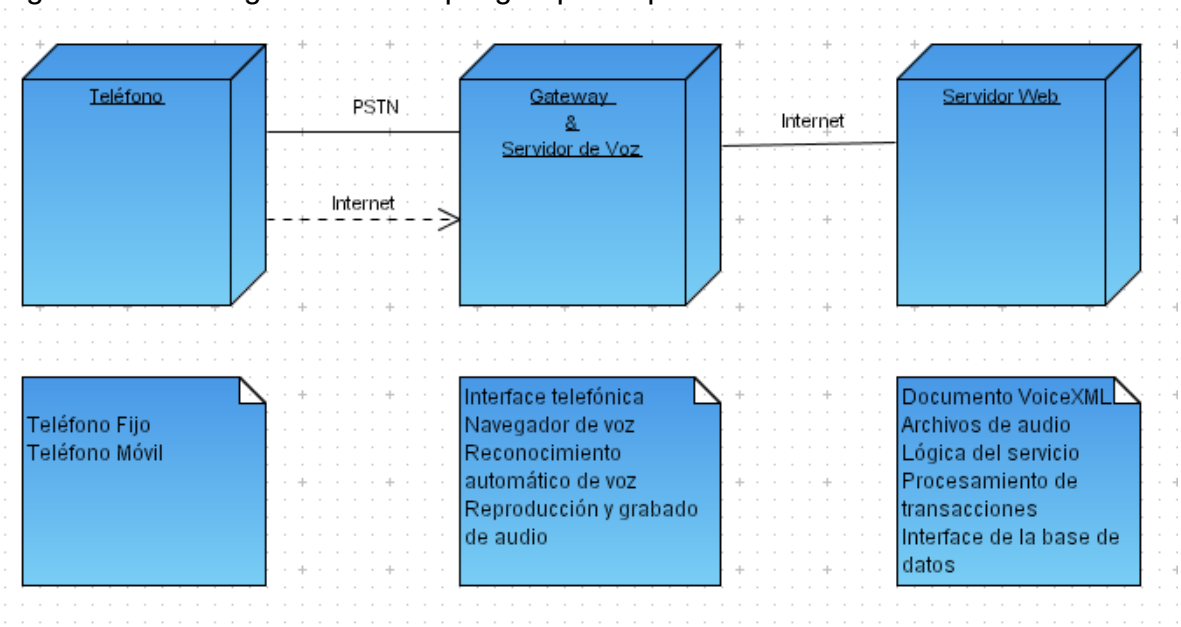
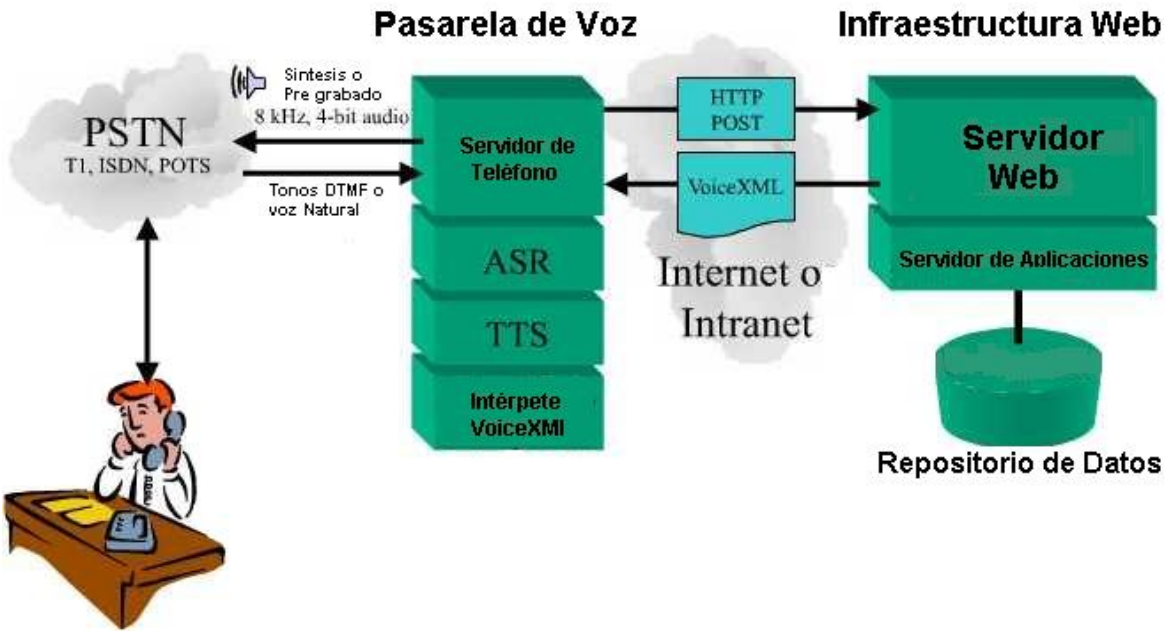


Figura Nº 12. Implementación de la arquitectura VoiceXML



Fuente: http://www.developer.com/img/VoiceXML/VoiceXMLarchitecture.jpg

De igual manera, debe permitir grabar los sonidos de audio recibidos desde el usuario. La especificación del lenguaje requiere un conjunto de formatos de archivo para el grabado de audio.

Transferencia: La plataforma debe estar en capacidad de soportar una tercera conexión a través de una red de comunicación, como lo es el teléfono.

La plataforma de implementación descrita anteriormente permite administrar un diálogo de voz, conformado típicamente por los siguientes pasos [30]:

1. El usuario llama desde un teléfono fijo o móvil y es atendido por el navegador (Browser) de VoiceXML
2. La pasarela (Gateway) de VoiceXML recupera el documento desde el servidor Web específico y presenta los diálogos con voz digitalizada.
3. El usuario dice su respuesta o presiona la tecla correspondiente
4. El dispositivo telefónico pasa el sonido grabado al mecanismo de reconocimiento de voz ASR, (si la entrada es voz), el cual utiliza la gramática predefinida y contenida en el documento VoiceXML
5. Basado en el resultado del reconocimiento, el navegador VoiceXML ejecuta los comandos, y presenta los resultados pregrabados. Si la lógica del documento VoiceXML lo requiere, presenta de nuevo los diálogos de voz digitalizada y espera la respectiva respuesta.

Un documento VoiceXML forma una máquina de estados conversacional finita. El usuario siempre está en un estado conversacional, o diálogo, en un instante. Cada diálogo determina la transición al siguiente, mediante una URI. Si dicha URI no se refiere a un documento, se toma por defecto el actual. Si no se refiere a un diálogo, se toma por defecto el primero. La ejecución termina cuando no se especifica sucesor, o si un elemento indica explícitamente el final de la conversación.
Puede contener:

- Elementos de alto nivel llamados diálogos, que pueden ser:
    - Formularios (forms). Los formularios definen una interacción para recoger valores y almacenarlos en una serie de variables.
    - Menús (menu). Un menú presenta al usuario un conjunto de opciones y la transición dependiendo de la selección.
    - Subdiálogos. Son semejantes a una llamada a funciones. Proporcionan mecanismos para invocar una nueva interacción y volver al formulario original.

- Declaraciones de variables.
- Gestores de eventos (event handler).

2.3.3 XHTML + Voice

La primer versión del lenguaje fue presentada por IBM, Motorola y Opera , y publicada por el Consorcio W3C, el 21 de diciembre de 2001, donde se indica que ha sido diseñado con el fin de brindar interacción a través del habla para contenidos web, al integrar tecnologías maduras como XHTML y eventos XML, con vocabularios desarrollados con XML, como parte de la plataforma de interface de habla del W3C. Incluye módulos de voz que soportan digitalización de voz, diálogos de voz, comandos y control, gramática del habla, y la posibilidad de añadir controles de voz para responder a eventos DOM específicos. Permitiendo reutilizar modelos de eventos familiares a los desarrolladores web. Las características de interacción por medio de la voz se integran directamente con XHTML y CSS [5].

Para lograr este fin, los documentos primero se reformulan en VoiceXML 2.0 como una colección de módulos, que junto con el Lenguaje de Marcado de Voz Digitalizada (Speech Synthesis Markup Language SSML) y el formato de reconocimiento gramático de voz (Speech Recognition Grammar Format SRGF), se integran con XHTML, usando módulos creados para el entorno XHTML + Voice. Finalmente el resultado se integra con Eventos XML como controladores de voz que pueden invocar a través de la interface estándar DOM2 EventListener.

Desde la primer versión se definieron los siguientes módulos de voz que conforman XHTML + Voice: Modularización de VoiceXML 2.0, creado para facilitar el proceso de integración entre VoiceXML y XHTML, Módulo de Salida de audio con voz y sin voz, que está definido por la especificación SSML, Módulo de diálogos de voz, usados para implementar el control del diálogo en la interacción; Módulo de gramática del habla, Módulo de eventos tipo VoiceXML, usado para crear eventos escucha que responden a los eventos de voz. Módulo controlador de eventos VoiceXML, cuya semántica es definida por la especificación VoiceXML 2.0

El 28 de enero de 2003 el Consorcio publica la versión 1.1 de XHTML + Voice. En el documento de la especificación se definen los siguientes principios de diseño [6]:

1. XHTML es el lenguaje anfitrión
2. XHTML + Voice es una extensión de XHTML Básico con un subconjunto de VoiceXML 2.0 al igual que Eventos XML y un pequeño módulo de extensión.
3. XHTML + Voice facilita la creación de tipos comunes de interacción multimodal.
4. VoiceXML es dividido en múltiples módulos para permitir la creación de perfiles que combinen diferentes entornos de desarrollo de aplicaciones.
5. Todas las partes de VoiceXML que se relacionan con el comienzo del documentos VoiceXML de una sola aplicación de voz son omitidos en XHRML + Voice

6. El proceso de dividir VoiceXML en múltiples módulos no debe alterar el modelo de ejecución de VoiceXML. Concretamente, un diálogo es ejecutado como lo ejecuta el algoritmo de interpretación de VoiceXML.
7. El proceso de dividir VoiceXML en múltiples módulos no modifica la función de los elementos VoiceXML 2.0 y los atributos que son parte del perfil.

También se presentan algunos aspectos del modelo de procesamiento de XHTML + Voice, que se deben tener en cuenta al momento de crear un documento haciendo uso de éste leguaje. Tal es el caso de la declaración de la sincronización de los modos de entrada, los eventos y sus controladores, el enlace de documentos por medio de la voz, la forma de cancelar un proceso, y la definición de hojas de estilo para la reproducción de archivos XHTML por medio de un sintetizador de voz y parlantes (Aural Style Sheets).

La versión 1.2 es la última, su desarrollo ha sido dirigido por VoiceXML Forum, y ha sido publicada el 16 de marzo de 2004. En ella se define un proceso más en el documento que permite retornar desde un formulario voiceXML. Cuando un evento es capturado con un diálogo de voz, el actor puede escoger finalizar el diálogo y retornar al contenedor XHTML, para lograrlo se incluye el elemento <return>.

También se incluye un evento de propagación en el módulo de Eventos XML. Dicho módulo permite la integración uniforme de escucha de eventos y asocia controladores de eventos con la interface de eventos DOM. De igual manera en el módulo de extensión de XHTML + Voice se incluye una gramática estándar para controles XHTML, que se utiliza para la sincronización.

2.3.4 SALT (Speech Application Language Tags)

El 15 de octubre de 2001 Cisco Systems Inc., Comverse Inc., Intel Corp., Microsoft Corp., Philips Speech Processing and SpeechWorks International Inc.. fundan el Foro SALT con el fin de crear una plataforma estándar independiente, libre de derechos de autor, que haga posible la interacción multimodal, y el acceso a la información, a las aplicaciones y servicios web, desde teléfonos, computadores personales, table PC, y asistentes personales digitalizados (PDA); brindando los siguientes servicios [82]:

- Al usuario final la posibilidad de interactuar con aplicaciones por medio de la voz, interfaces gráficas o de texto en forma independiente o combinada.
- A los desarrolladores la posibilidad de incluir voz en las aplicaciones, ampliando las tecnologías existentes HTML, XHTML, XML, usando lenguajes y herramientas de desarrollo familiares.
- A los empresarios la posibilidad de ofrecer aplicaciones web, a través de diferentes medios de presentación, reduciendo la complejidad y el costo. Adicionalmente puede utilizar la experiencia y tecnología existente,

eliminando la necesidad de crear una aplicación específica para cada tipo de salida.

- A los proveedores de servicios la posibilidad de desarrollar un amplio rango de aplicaciones que permiten incrementar e rango de servicios, ofreciendo nuevas posibilidades de negocio.

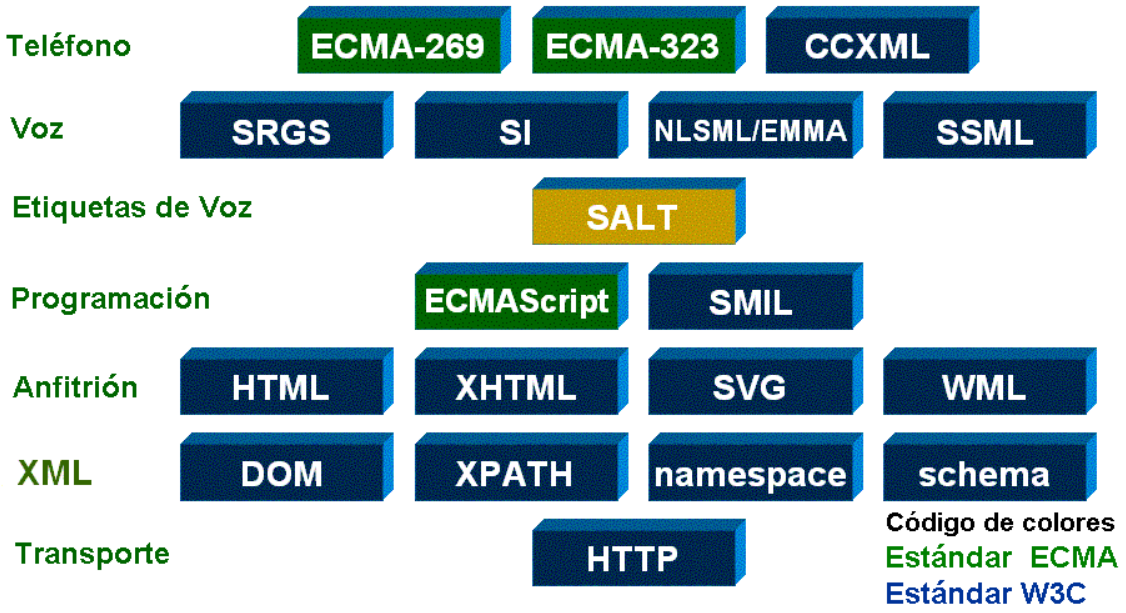
El 31 de enero de 2002 se unen 18 nuevos miembros al Foro SALT, entre los que figuran Compaq Computer Corp., Edify Corp., Genesys Telecommunications Laboratories Inc. [83] Ese mismo día se publica el documento que define los principios de diseñado de SALT [84]: Integración del habla con páginas Web, separación de la interface de la lógica del negocio y los datos, potencia y flexibilidad en el modelo de programación, reutilización de los estándares existentes para gramática, salida de habla y resultados semánticos; amplio rango de dispositivos, y costos mínimos de autorización para diferentes tipos de dispositivos y modalidades. Ese mismo año el 7 de mayo, se unen 19 miembros más y actualmente son más de setenta.

El 15 de julio de 2002 se publica la especificación de la versión 1.0 de SALT [85]. Incluye mecanismos para la entrada, con capacidad para la captura, el reconocimiento y el procesamiento de entradas de voz; mecanismos para la salida de voz, como audio y voz digitalizada; y mecanismos para el control el control de la llamada que facilitan la recepción, respuesta transferencia y desconexión de la llamada, además de posibilitar el diálogo en forma de conferencia.

En forma paralela las instituciones miembros del foro, presentaron productos basados en la especificación SAL. Tal es caso de Microsoft, que presentó la versión beta Visual Studio .NET con su respectivo SDK (software development kit) basado en la especificación SAT. Philips anunció el navegador SALT como primer prototipo de una plataforma telefónica basada en SALT. Hey Anita desarrollo un navegador FreeSpeech™ SALT, para soportar aplicaciones desarrolladas en Microsoft .NET Speech SDK y Visual Studio .NET.

El 13 de agosto de 2002 el Forum SALT anunció que ha aportado la especificación SALT al W3C [86]. También solicitó Al grupo de trabajo de interacción multimodal, y al grupo de trabajo de navegador de voz, revisar la especificación como parte de un desarrollo para promover la interacción multimodal y hacer posible el manejo de aplicaciones de voz en la web. El 14 de junio de 2003, publicó la versión compatible con el lenguaje de marcado SVG, incrementando la funcionalidad y el potencial de SALT, al posibilitar la interacción multimodal a través de voz y de interfaces gráficas [87].

Figura Nº 13. Estándares Para el trabajo de SALT

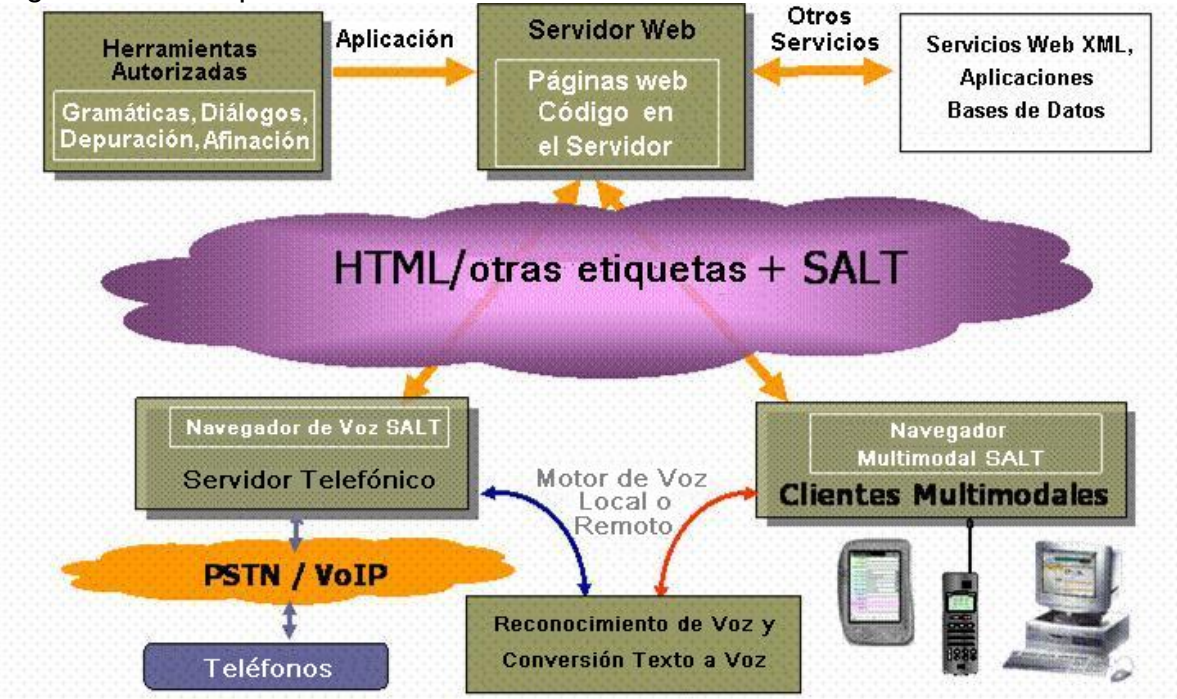


Fuente: http://www.saltforum.org/speechtek05/downloads/SALT%20at%20SW%202005%20-%20slides.zip

SALT es un Lenguaje de Marcado para interfaces de voz. Consiste de un pequeño conjunto de elementos XML, con atributos asociados y propiedades de objetos DOM (Document Object Model), eventos y métodos que aplican a las interfaces de voz para páginas Web. El formalismo y la semántica son independientes de la naturaleza del documento fuente, debido a que SALT puede ser usado igualmente en forma efectiva con HTML y todas sus formas, o con WML y otro lenguaje de marcado derivado de SGML, razón por la cual puede ser usado con diferentes estándares para escribir interfaces de sólo voz y aplicaciones multimodales [90], como se observa en la figura 12.

SALT en su arquitectura (Figura Nº 13) aplica el principio de la modularidad, lo que le permite soportar diferentes perfiles de dispositivos. Teniendo en cuenta que cada perfil define un conjunto de características obligatorias para una clase de dispositivo, que se ajustan a entornos particulares, dependiendo de aspectos como la limitada capacidad de procesamiento y memoria, entre otras. Los dos escenarios más importantes en los que se utiliza SALT son: Aplicaciones multimodales, que soporten de forma concurrente o independientemente entradas con voz, teclado, ratón, lápices, y salidas de audio, voz digitalizada, texto plano, video en movimiento y gráficos. Y aplicaciones de sólo voz, exclusivamente para teléfonos, en las que SALT se encarga de administrar el flujo de interacción del diálogo.

Figura Nº 14. Arquitectura SALT



Fuente: http://www.saltforum.org/images/SALTArch.jpg

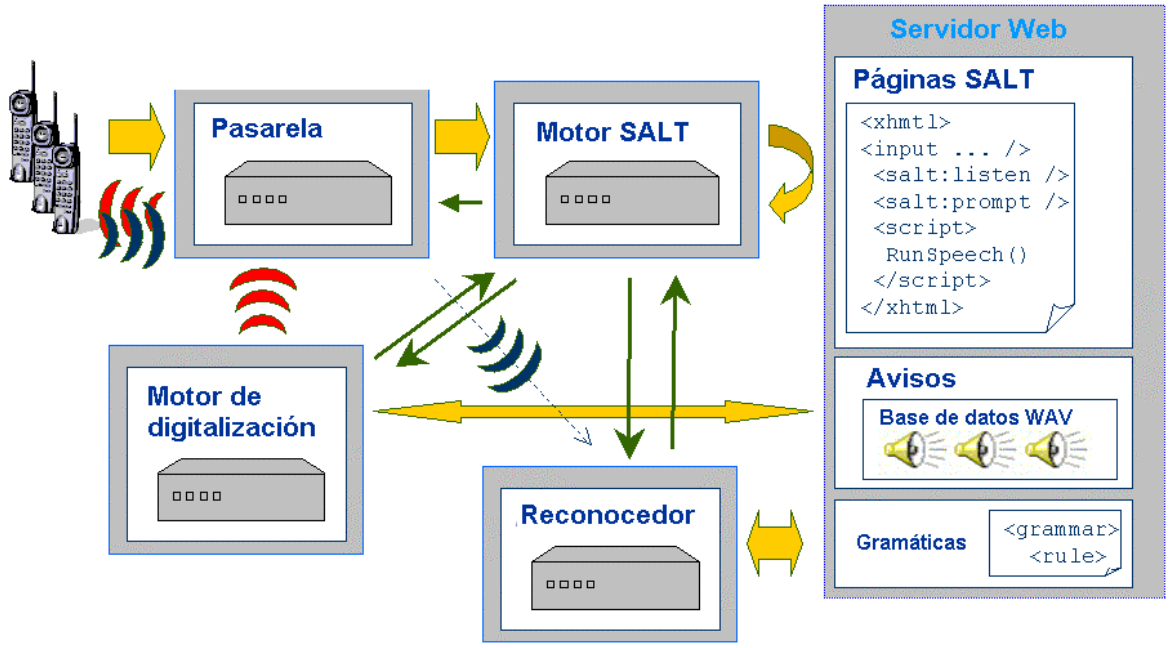
Según la figura Nº 14, cada escenario cuenta con un navegador especial, que utiliza los servicios de un mecanismo local o remoto de reconocimiento y digitalización de voz, para permitir la interacción entre el usuario final y la aplicación que se encuentra en el servidor web. La aplicación se encarga de definir los diálogos, la gramática y la lógica necesaria para tener acceso a bases de datos y/o a los servicios web. A continuación se representan en forma gráfica cada uno de los escenarios.

Los elementos de SALT están organizados jerárquicamente, según lo muestra la figura Nº 16. El elemento listen permite el reconocimiento de voz y el grabado de audio. El elemento prompt se utiliza para especificar el contenido de la salida de audio. El dtmf es útil para aplicaciones telefónicas que requieren especificar entradas DTMF y su respectivo significado. El elemento smex (Simple Messaging Extension) establece comunicación con componentes externos a la plataforma SALT[24].

Figura Nº 15. Escenario multimodal

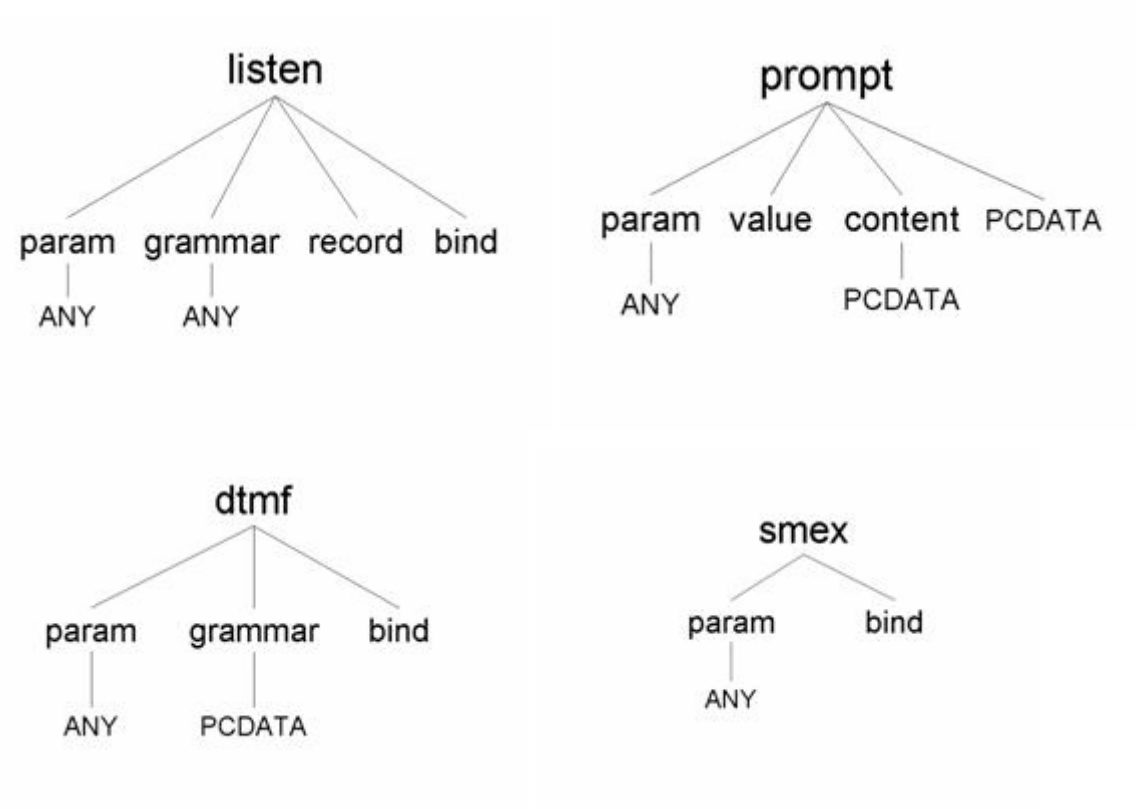
Fuente: http://www.saltforum.org/speechtek05/downloads/SALT%20at%20SW%202005%20-%20slides.zip

Figura Nº 16. Escenario sólo voz



Fuente: http://www.saltforum.org/speechtek05/downloads/SALT%20at%20SW%202005%20-%20slides.zip

Figura Nº 17. Estructura jerárquica de los elementos de SALT



Fuente: http://www.saltforum.org/devforum/tutorials/tutorials/SALTSummary2.asp

2.3.5 TalkML

Es un Lenguaje de Marcado experimental escrito en XML y desarrollado por los laboratorios de Hewlet Packard, que permite describir diálogos de voz en términos de mensajes de salida, gramáticas de voz y reglas de producción, para interactuar ante las respuestas del usuario. Inicialmente fue usado para explorar ideas sobre estructuras de diálogos orientadas a objetos, y para las siguientes generaciones de hojas de estilo[81].

Fue diseñado para los siguientes mercados [10]:

- Centros de Llamadas para ventas y soporte de servicios. Agregando reconocimiento de voz a los actuales sistemas DTMF
- Teléfonos inteligentes con pantallas
- Acceso a correo electrónico, apuntes, noticias, servicios de viajes, etc, cuando el usuario de encuentra en el camino, por ejemplo haciendo uso de los sistemas integrados al vehículo.

- Dispositivos móviles muy pequeños, con capacidades limitadas de video y teclado.

Una aplicación TalkML soporta conversaciones de diálogo más natural que los sistemas basados en DTMF, y está definida por un conjunto de bloques de diálogos, que comprenden mecanismos de escucha de solicitudes por medio de voz, mecanismos de entrada y mecanismos para identificar la siguiente acción a realizar. También se definen estrategias de espera para brindar ayuda, o para mantener el sistema en estado de espera de solicitudes. Cada aplicación define variables que representan información acerca de su estado. Además existe un conjunto de reglas gramaticales, acordes a las respuestas del usuario, que determina la forma como el diálogo es procesado, teniendo en cuenta que en cualquier momento puede ser interrumpido para cambiar de tarea o solicitar ayuda.
Un elemento ***dialog*** define un bloque de diálogo, que consiste en uno o más pasos que son ejecutados en forma secuencial, regresando al comienzo, después de haber ejecutado el último paso. Cada paso se representa por medio del elemento ***say***, el cual puede contener otros elementos, tales como ***var*** que se utiliza para insertar valores en una variable de la aplicación, como se muestra en el siguiente ejemplo

```
<say>
  Que dia  quiere viajar usted a
  <var name="destination"/>?
</say>
```

La sección de código que representa la espera de la respuesta por parte del usuario es:

```
<listen [grammar=nombre-de-la-gramática]
  [timeout=seconds] [response=nombre-de-la-variable]>
  ... reglas gramaticales ...
</listen>
```

Las reglas gramaticales se pueden incluir o referir en una definición externa con el elemento ***grammar***, a través del cual se pueden definir gramáticas separadas, y referirse a ellos como no terminales en otras reglas gramaticales, usando el elemento de reglas ***rule.*** El atributo ***response*** es usado para nombrar la variable donde se guarda la respuesta, y es conveniente cuando la gramática consiste en un alista simple de opciones, por ejemplo:

```
<listen response="airport">
  "Bristol"|"Heathrow"|"Gatwick"|"Stansted"
</listen>
```

En el caso eventual que haya transcurrido el tiempo máximo de espera, o cuando la respuesta del usuario no se entiende, se dispara un elemento de ***error***, que incrementa un contador de errores y permite presentar un mensaje acorde a la situación. Adicionalmente, se puede solicitar ayuda en cualquier momento, para lo cual se utiliza el elemento ***help***, que define el texto de la ayuda y un mecanismo para darle la posibilidad al usuario de pasar a otros diálogos, por medio del elemento next, dependiendo de los valores actuales de las variables que maneja la aplicación.

Una aplicación TalkML se caracteriza por permitir complementar la salida de voz con imágenes para pantallas pequeñas, que pueden ser sincronizadas usando SMIL. Además combina la interacción por voz con tonos DTMF, facilitando el trabajo al usuario final. También posibilita la creación de módulos y su reutilización, por medio del elemento ***task***, lo que a su vez hace posible exportar e importar un conjunto de variables.

2.3.6 VoxML

El 30 de septiembre de 1998 Motorola presenta el Lenguaje de Marcado VoxML para la creación de aplicaciones de voz [94]. Su principal objetivo fue ofrecer una plataforma común para la creación de aplicaciones que permiten el acceso a los recursos de Internet a través de la voz, posibilitando la creación de interfaces en forma de diálogos, al permitir navegar y realizar entradas del usuario por medio de la voz utilizando mecanismos de reconocimiento de voz, y presentado la salida con voz digitalizada o reproducción de sonidos previamente grabados.

Cinco meses después de liberar la especificación VoxML, exactamente el 2 de Marzo de 1999 [70], debido a las limitaciones de HTML, Motorola se une con AT&T, IBM y Lucent para conformar el Foro VoiceXML, con el fin de definir un lenguaje estándar para el diseño de diálogos, que pudiera ser usado por los desarrolladores para construir aplicaciones de voz, actualmente conocido como VoiceXML, fundamentado en las características de VoxML. El W3C hace público su agradecimiento a Motorola por el significativo aporte que representó VoxML en la construcción del Lenguaje VoiceXML, al lanzar la versión 1.0 el 7 de marzo de 2000

VoxML soporta la creación de aplicaciones interactivas de voz, debido a que se basa en el Lenguaje Extensible de Marcado XML, lo que implica que soporta sus reglas de sintaxis y semántica, con todas las ventajas que esto representa. Además permite el uso de gramáticas libres de contexto escritas en formato normal extendido (Extended Backus-Naur Form EBNF)

El gran paso que dio Motorola al presentar VoxML consisten en haber hecho posible que Internet hable y escuche [96], pero desafortunadamente los

contenidos estaban diseñados para interfaces gráficas, y cuando se necesitaba hacer uso de reconocimiento, interpretación y transformación de voz a HTML, se presentaban errores debido a que HTML no había sido diseñado para representar datos de voz, lo que limitó una amplia aceptación y difusión en su momento de ésta tecnología .

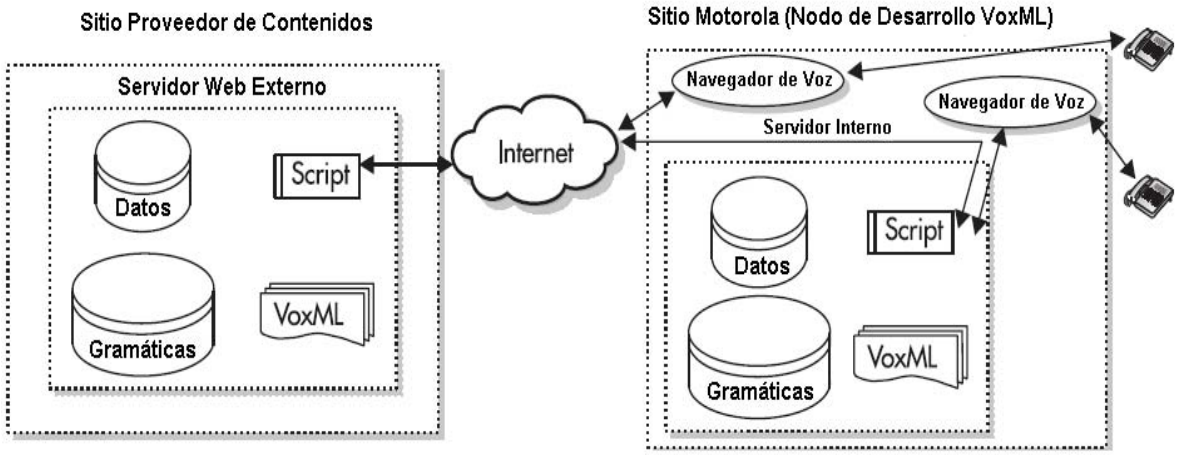
VoxML ha sido diseñado para que el usuario interactúe de la siguiente manera:

1. Identifica el sitio web que ofrece interacción a través de teléfono.
2. la compañía debe disponer de un número telefónico para llamar al sitio web.
3. el usuario realiza la llamada en forma normal
4. El usuario se conecta al navegador de voz del sitio web.
5. Al lograr la conexión se presenta un mensaje que invita al usuario a interactuar con la aplicación
6. Cuando el usuario responde, se dispara una solicitud del navegador de voz al servidor de aplicaciones, el cual debe responder con una nueva página acorde a la solicitud, que permita el control del flujo del diálogo.
7. el usuario y la aplicación interactúan hasta que la transacción quede realizada, teniendo en cuenta que el usuario puede finalizar la sesión en cualquier momento.

Para lograr dicha interacción, el desarrollador de aplicaciones VoxML debe especificar y diseñar todos los posibles diálogos que se pueden presentar, utilizando un entorno de desarrollo especial, que le permita definir formalmente los archivos que controlan el diálogo, y montarlos en el servidor para que puedan ser referenciados a través de la aplicación, haciendo uso de la línea telefónica, como se representa en la siguiente figura, que muestra los elementos que conforman la plataforma de despliegue de una aplicación VoxML.

Los documentos VoxML se encuentran en un servidor web, donde son ubicados y accedidos por el navegador, el cual reside en un lugar comercial, interactuando con el usuario vía telefónica y con el servidor a través de Internet, y haciendo uso de mecanismos de reconocimiento y digitalización de voz. Para cierto tipo de servicios, donde la información no cambia con frecuencia, VoxML permite crear documentos estáticos simples. Cuando sea necesario hacer alguna modificación, se hace manualmente. En el caso de contenidos dinámicos, la lógica de cada respuesta para las entradas del usuario es codificada en un documento VoxML o generada a partir del resultado de una consulta a la base de datos, dependiendo de las habilidades del desarrollador y la interpretación de los requerimientos.

Figura Nº 18. Plataforma de despliegue de una aplicación VoxML



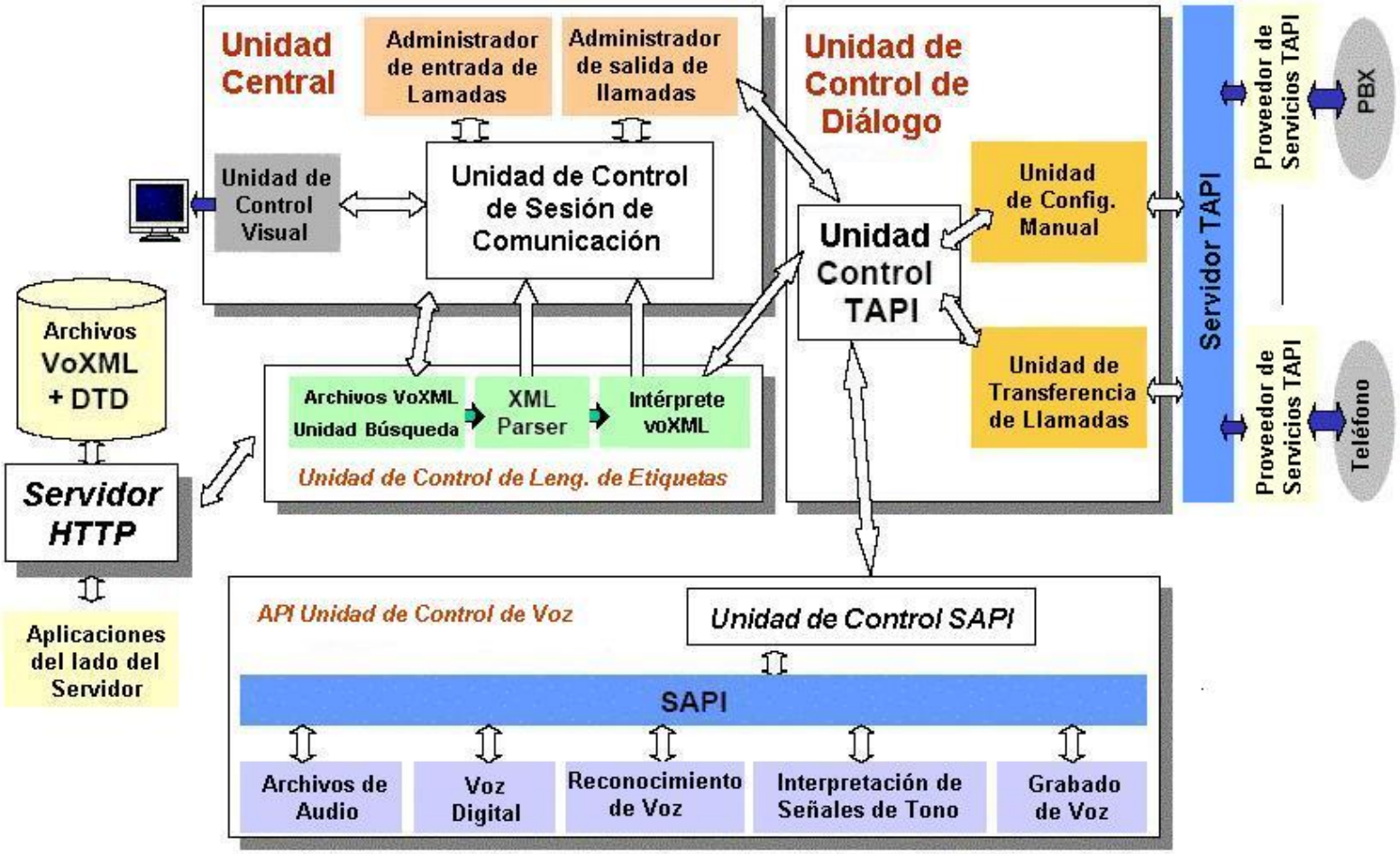
Fuente: www.osborne.com/products/0072224584/0072224584_ch01.pdf

El navegador de voz, ya sea el de Motorola o cualquier otro disponible en un sitio comercial, carga el documento VoxML desde el servidor y convierte las etiquetas DIALOG en comandos de voz. El usuario responde a los enunciados de voz que emite el navegador, el cual reconoce e interpreta la respuesta del usuario, de acuerdo al código del documento VoxML. El servidor web se encarga de realizar la lógica de la aplicación y emite un resultado que se envía al navegador para que pueda ser escuchado por el usuario final.

Además de los servicios de voz y audio, el navegador de la pasarela tiene el mismo comportamiento de cualquier otro, permitiendo almacenar cookies, páginas temporales y recuperar datos referenciados por medio de una dirección URL y el protocolo http. La pasarela VoxGateway forma parte de la plataforma de comunicación MIX (Motorola's Mobile Internet Exchange) que combina servidores, pasarelas WAP, aplicaciones y contenidos que hacen posible a usuarios móviles enviar y recibir información rápidamente, fácil y a través de una variedad amplia de dispositivos

En el documento Web Enabled Telecommunication Service Control Using VoxML, Bilel Guedhami*, Cornel Klein y Wolfgang Kellerer, presentan la Arquitectura para servicios de telecomunicación basados en VoxML (Figura Nº 19), conformada por cuatro componentes:

Figura Nº 19. Arquitectura para servicios de telecomunicación basados en VoxML



Fuente: http://www.lkn.ei.tum.de/~wolfgang/ wk/publ/kellerer_smartnet00b.pdf

Unidad de control de diálogo: Responsable de recibir, manejar y transferir las llamadas, a través de la unidad de control de la interface de programación entre aplicaciones telefónicas (Telephony Application Program Interface TAPI).

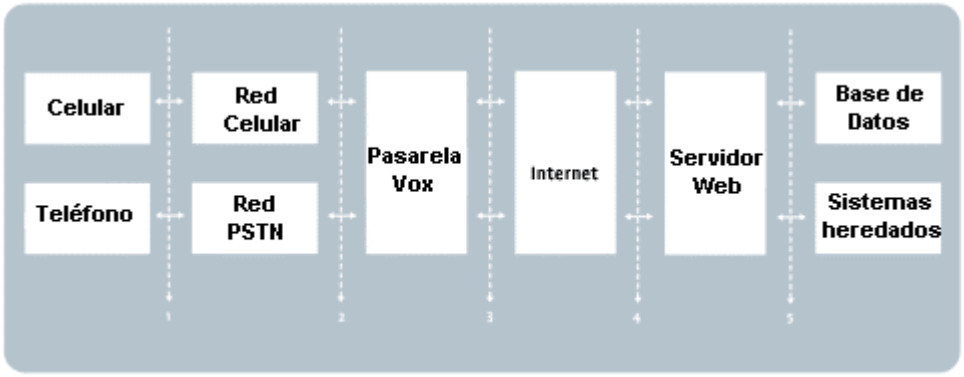
Unidad central: Encargada de controlar la sesión de comunicación, administrando las entradas y salidas de la llamada.

Unidad de control del lenguaje de Marcas: Realiza la búsqueda de los archivos VoxML, para hacer la correspondiente conversión a XML y su respectiva interpretación.

Unidad de control SAPI: La interface de programación entre aplicaciones de voz (Speech API), permite el reconocimiento de voz, la interpretación de tonos DTMF, digitalización y grabado de la voz, al igual que la reproducción de archivos de audio.

La figura Nº 20 muestra el flujo de datos entre el teléfono y la base de datos. Inicialmente el usuario marca desde un teléfono local al número donde está disponible la aplicación, haciendo uso de la red pública telefónica conmutada (PSTN Public Switched Telephone Network), o desde un teléfono celular, haciendo uso de la red móvil. La pasarela o VoxGateway utiliza sus mecanismos de reconocimiento automático y digitalización de voz, para interpretar los comandos de voz y servir de intermediario entre el teléfono y la Internet. Por una parte, la pasarela sirve como interface para la red pública telefónica, mientras que por otro lado, se comunica con Internet por medio de sus protocolos estándar.

Figura Nº 20. Flujo de datos entre el teléfono y la base de datos



Fuente: http://www.newarchitectmag.com/documents/s=4576/new1013637063/1.htm

Un documento VoxML tiene una estructura jerárquica, en la cual cada elemento (excepto el elemento DIALOG) es contenido por otro elemento. Los elementos fundamentales de VoxML son: DIALOG, que define el rango del documento, y STEP que define el estado de una aplicación. Además existen otros elementos

básicos que permiten la interacción, como PROMPT, INPUT, HELP, entre otros[70].

2.3.7 SSML (Speech Synthesis Markup Language)

SSML es parte del conjunto de especificaciones para navegadores de voz desarrolladas por el W3C. Ha sido creada para proveer un lenguaje de marcado que facilite la generación de voz digital en aplicaciones de voz. Su principal objetivo es proporcionar al autor de contenidos que se puedan convertir en voz digital, una forma estándar de controlar aspectos relacionados con la salida de voz, como son: pronunciación, volumen, tono, entre otros, a través de diferentes plataformas capaces de soportar voz digitalizada.

El primer intento para establecer un estándar de ésta naturaleza fue SABLE: A Synthesis Markup Language, lenguaje de marcado basado en XML/SGML para desarrollar aplicaciones que requieren digitalizar voz, haciendo uso de un paradigma de control común de conversión TTS [91]. La versión 1.0 se dio a conocer en 1998, y se creó teniendo en cuenta los siguientes principios [9]: Facilidad de uso, portabilidad, extensibilidad, internacionalización permitiendo el manejo de un amplio número de lenguajes, y disponibilidad de marcas para digitalizar la voz. Sin embargo, ésta solución no tuvo la acogida esperada debido a que contaba con los siguientes inconvenientes [77]: La mayoría de sus características ya habían sido abordadas por CSS, ya que permite que la misma hoja de estilo pueda ser aplicada a diferentes documentos XML y HTML. Además SABLE omite muchas características necesarias que requieren los navegadores de voz, requiere de medios específicos de marcado, además de mezclar estructura y estilos.

Aún así, SABLE se convirtió en el punto de partida para la definición de los requerimientos de marcado para voz digitalizada de los lenguajes de marcado de voz, publicados el 23 de diciembre de 1999, y clasificados en [42]: Criterios de diseño, integración y arquitectura, texto contenido y presentación específica de la voz. El primer borrador de la especificación SSML, fue publicado por el W3C, el 8 de agosto de 2000[97]. En él se presenta: el proceso de digitalización de la voz, las etiquetas que representan los elementos y atributos del lenguaje, clasificadas como: estructura del documento, procesamiento del texto y pronunciación, prosodia [91] y estilo, y otros elementos. Adicionalmente se presentan algunos ejemplos y la DTD para SSML.

El realiza siguiente publicación del borrador de la especificación, el 3 de enero de 2001, en ella incluye además de los aspectos abordados en el anterior, la correspondencia entre las etiquetas de voz digital y los fragmentos de documentos. Más tarde, el 5 de abril de 2002 se da a conocer un nuevo borrador

que incluye entre otros, temas como la estructura del documento SSML y su integración con otros lenguajes de marcado. Ese mismo año, el 2 de diciembre se publica el primer borrador de la versión 1.0 del lenguaje. Dicha versión fue presentado como candidata a recomendación el 18 de diciembre de 2003, y como propuesta de recomendación el 15 de julio de 2004, siendo finalmente aceptada como recomendación el 7 de septiembre de 2004.

La especificación SSML está basada en el formato de texto JSML (JSpeech Markup Language), propio de Sun Microsystems y establecido el 5 de junio de 2000, el cual sirve para realizar anotaciones de entradas de texto para voz digitalizada. Los elementos de JSML permiten digitalizar la voz con información sobre detalles acerca de cómo decir el texto, mejorando la calidad, la naturalidad y el entendimiento de la voz digitalizada. JSML define elementos que describen la estructura de un documento, indicando la pronunciación de palabras y frases, acento, tono, ritmo de la voz, y otros controles importantes que caracterizan la voz [43].

SSML ha sido diseñado teniendo en cuenta criterios como: consistencia, interoperabilidad, soporte de salidas en un amplio rango de aplicaciones con gran cantidad de contenidos de voz, internacionalización, al permitir la salida en un amplio rango de lenguajes, reutilización e implementación, ya que permite utilizar los mismos documentos para diferentes aplicaciones, que pueden ser implementadas con la tecnología existente.

El sistema que soporta SSML es el responsable de presentar el documento hablado de acuerdo a la intencionalidad del autor, haciendo uso de la información contenida en las etiquetas del lenguaje. Para ello debe realizar los siguientes pasos [18]:

Análisis sintáctico: Se utiliza para extraer el árbol del documento y la forma del contenido del documento de entrada. Es de vital importancia para los siguientes pasos, ya que identifica la presencia y corrección de las etiquetas.

Análisis estructural: La estructura del documento influye en la forma en que debe ser leído. Se especifica por medio de etiquetas en forma explicita, o haciendo uso de los signos de puntuación en el texto original.

Normalización del texto: es un proceso automático que convierte formas de escritura particulares del lenguaje en su representación hablada. Para ello se utiliza la etiqueta say-as.

Conversión de texto a fonema: Una vez establecido el conjunto de palabras que se van a decir, se debe definir la pronunciación de cada una de ellas. La pronunciación de la palabra se puede describir como la secuencia de fonemas, considerados como unidades de sonido de un lenguaje o dialecto, que sirven para

distinguir una palabra de otra, teniendo en cuenta que cada lenguaje tiene un conjunto de fonemas, y además existen diferencias entre la forma en que se escriben las palabras y su pronunciación. Utiliza etiquetas como phoneme, say-as, lexicon.

Análisis prosódico: Consiste en definir el conjunto de características de la voz, tales como tono, ritmo, pausas, rapidez, énfasis en algunas palabras, entre otras, que son importantes para generar un sonido de voz natural que permita asignar un significado correcto al lenguaje hablado. Hace uso de los elementos: enphasis, break y prosody.

Producción de sonido: Los fonemas y la información prosódica son usados para la producción del sonido de la voz. El lenguaje cuenta con las etiquetas voice y audio para lograr éste objetivo.

2.3.8 XForms

Desde que se introdujo en 1993 HTML se convirtió en el instrumento por excelencia para el manejo de contenidos en entornos web. Sin embargo, al ampliarse la funcionalidad de la web, con el surgimientos de nuevos servicios de comercio electrónico y ambientes virtuales de aprendizaje, entre otros, se hacen evidentes las desventajas de HTML, por ser un lenguaje que no permite separar modelo de datos de la presentación de la interface gráfica del usuario. Por otra parte, el surgimiento de dispositivos electrónicos que permiten navegar por Internet, pero que cuentan con limitaciones de ancho de banda, resolución de imagen y capacidad de memoria, como los asistentes personales digitales (PDA) y los teléfonos celulares, implican un acceso a los contenidos por medio de la interacción multimodal, para la cual HTML no fue diseñado.

Inicialmente como solución a éste problema se plantean los requerimientos para formularios extendidos de XHTML, el 6 de septiembre de 1999, clasificándolos de la siguiente manera [88]:

- Interoperabilidad y accesibilidad: separación entre la lógica y la presentación, definición de la funcionalidad del formulario en XML, navegación independiente del dispositivo y la aplicación, al igual que la sintaxis de eventos.

- Internacionalización: Soporte de varios lenguajes y conjuntos de caracteres, formatos de datos y grupos de campos para regiones específicas.

- Lógica de los formularios: Integración con XML DOM, cálculo de campos, tipos de datos, validación de entrada, dependencia de campos y datos.

- Interacción: Mecanismos enriquecidos de interacción cliente servidor, mecanismos de seguridad y autenticación, amplio rango de dispositivos de entrada, conservar el estado actual del formulario.
- Presentación: Conservar los mecanismos existentes y emergentes de presentación, realzar las posibilidades visuales para el control de formularios, personalizar el control de los formularios

El 29 de marzo de 2000, el W3C emite el documento sobre los requerimientos de XForms, concentrándose en los requerimientos básicos y de caracteres (definido en XML y utilizable en XML, fácil migración desde HTML 4, separación de la lógica y la presentación, integración con DOM, independencia de la aplicación y el dispositivo, independencia de la región e internacionalización, construcción modular), los requerimientos del modelo de datos y la lógica (tipos de datos, librerías de tipos de datos, validación de entradas, envío de XML al servidor, manejo de expresiones y cálculos, Dependencia de campos y datos, grupos de campos expansibles, características de seguridad), y los requerimientos de la interface de usuario (nuevas formas de control, soporte de múltiples páginas por formulario, y navegación independiente del dispositivo y la aplicación) [25]. Para la versión 1.0 de la especificación se emiten dos documentos más sobre los requerimientos de XForms, los días 21 de agosto de 2000 y 4 de abril de 2001. Y para la versión 1.1 se han publicado también dos documentos que definen sus requerimientos, uno el 26 de enero de 2004 y otro el 4 de abril del mismo año.

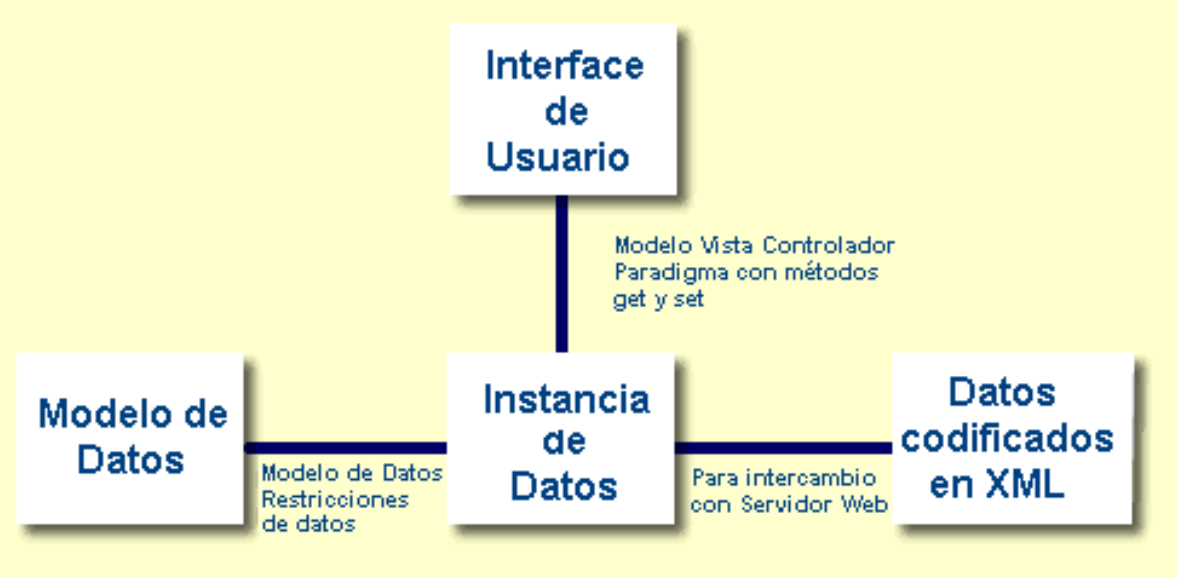
El primer borrador de la versión 1.0 de XForms se presenta el 6 de abril de 2000, convirtiéndose a partir de ese momento en un lenguaje de marcado independiente de la plataforma para la interacción del usuario, y para el comportamiento transaccional entre agentes y entidades remotas. En él se define la arquitectura de XForms y su modelo de datos, indicando los diferentes tipos de datos y las estructuras que soporta [26].

El 15 de agosto del mismo año se publica el segundo borrador, indicando que el principal propósito de XForms es separar la interface de usuario y los formularios de presentación, del modelo de datos y de la lógica de la aplicación, permitiendo que un mismo formulario pueda ser utilizado por una amplia variedad de dispositivos. Además se comenta que XForms está dirigido a dos grupos especiales de comunidades: creadores de páginas web dinámicas que quieran alcanzar más con un menor esfuerzo, y organizaciones que ya tienen desarrollado un amplio modelo de datos utilizando esquemas XML, y requieren formularios par ingresar datos acordes a dichos modelos [27]. A partir de dicho trabajo se publican 8 borradores más de la especificación, y sólo hasta el 14 de octubre de 2003 el publica el documento que se convierte en recomendación de XForm 1.0. La última publicación se realizó el 15 de noviembre de 2004, cuando se dio a conocer el primer borrador de la versión 1.1 de XForms.

Desde su inicio XForms planteó como principales objetivos [49]:

- Desacoplar los datos, la lógica y la presentación
- Enriquecer la interface de usuario para satisfacer las necesidades de negocio, del consumidor y de las aplicaciones de control de dispositivos.
- Permitir la internacionalización
- Soportar formularios estructurados de datos
- Formularios multiplataforma por página, y páginas por formulario
- Integración con otros conjuntos de etiquetas XML

Figura Nº 21 Arquitectura XForms

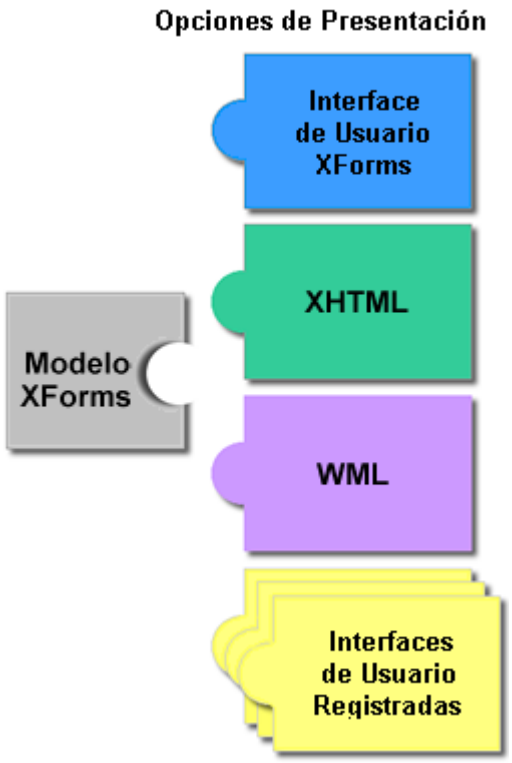


Fuente: http://www.w3.org/TR/2000/WD-xforms-datamodel-20000406/

Para lograrlos se han separado la lógica y la presentación en dos especificaciones: El modelo de datos y la interface del usuario, como lo ilustra la figura Nº 21. La primera permite abstraer la estructura del formulario para que sea definida sin necesidad de indicar explícitamente la interface de usuario. La segunda plantea un nuevo diseño de interface de usuario para enriquecer la interacción, asegurando la independencia del dispositivo, sin sacrificar la funcionalidad lo que permite enlazar otras gramáticas XML al modelo de datos.

Esto se puede observar en el siguiente gráfico, que ilustra como una sola definición de formulario XML independiente del dispositivo (Modelo XForms) tiene la capacidad de trabajar con una variedad de interfaces de usuario estándares y propietarias.

Figura Nº 22. Interoperabilidad de XForms



Fuente: http://www.w3.org/MarkUp/Forms/

La interface de usuario de XForms provee un conjunto de estándares de controles visuales que pretenden desplazar los actuales controles de formulario XHTML. Estos controles son usados directamente dentro de XHTML, y otros documentos XML. Por otra parte, el modelo XForms describe la estructura de la instancia de los datos, como una colección de datos, lo que permite al formulario representar la estructura de intercambio de datos, el cual se realiza a través del protocolo de envío de XForms , que define como enviar y recibir datos, incluyendo la posibilidad de suspender y reanudar el proceso desde el mismo formulario.

XForms ha surgido como la mejor herramienta para la construcción de interfaces multimodales, debido a que garantiza los siguientes servicios [78]: Permite generar interfaces de usuario apropiadas para conectar diferentes tipos de dispositivos; provee mecanismos de retroalimentación para el usuario que realizan validación automática en el lado del cliente, al igual que en el servidor, además organiza las entradas en el servidor en una estructura conveniente para la aplicación final.

Esto se logra gracias a su arquitectura, que satisface las características del Modelo Vista Controlador, al presentar los siguientes componentes:

Model : Todos los aspectos no relacionados con la presentación son encapsulados por el modelo de datos de XForms. Dicho modelo incorpora una instancia de XML que lleva la entrada del usuario, las restricciones usadas para validarla, y los metadatos necesarios a cerca de cómo el usuario debe comunicarse con el servicio web.

User Interface: XForms define un vocabulario de la interface de usuario que consiste de controles abstractos y elementos de agregación usados para crear potentes interfaces de usuario. Dicho vocabulario ha sido diseñado para capturar la esencia de la interacción del usuario más que la presentación final, sobre cualquier dispositivo o cualquier modalidad específica, lo que lo convierte en la herramienta por excelencia para desarrollar aplicaciones web para diferentes dispositivos y modalidades.

Submit: Permite a los creadores de las aplicaciones especificar donde, cómo y qué piezas de datos enviar al servidor web. De igual manera permite especificar que acciones tomar sobre las respuestas recibidas del servidor.

2.3.9 EMMA (Extensible MultiModal Annotation markup language)

Es un lenguaje de marcado diseñado para aplicaciones multimodales que provean interpretación semántica para una variedad de entradas que incluyan (pero no necesariamente limitadas a) voz, lenguaje natural, texto, interfaces gráficas, registro de tinta digital.

El documento que describe los requerimientos de EMMA fue publicado el 13 de enero de 2003 [64]. En él se indica el alcance y los requerimientos generales, los requerimientos del modelo de datos, los requerimientos de anotación, y la necesidad de mantener la integración de EMMA con un amplio rango de estándares adoptados por la especificación y sus características.

El primer borrador de la especificación se presentó el 11 de agosto del mismo año. Su contenido se centra en presentar los usos de EMMA, la estructura de un documento EMMA, el modelo de las anotaciones y algunos aspectos de la sintaxis formal del lenguaje. Cuatro meses después, el 18 de diciembre, se presenta el segundo borrador de la especificación, donde se incluyen las propiedades de los elementos de anotación de EMMA, y el alcance de las anotaciones EMMA en derivaciones secuenciales y compuestas.

El tercer borrador se publica el primero de septiembre de 2004. Incorpora dos elementos estructurales de EMMA: Los elementos de gramática y los la representación compacta de listas largas de posibles resultados de reconocimiento o interpretación para las entradas multimodales. De igual manera, se agregan dos propiedades adicionales: el soporte a la multimodalidad

compuesta y el costo de la anotación. Además se decide quitar la sintaxis RDF usada en los dos borradores anteriores de la especificación, y los nombres de espacio han sido sustituidos por la marca #. Respecto a la pregunta no resuelta sobre “timestamp” se resuelve con un nuevo enfoque. Las unidades basadas en milisegundos también son introducidas para tiempos relativos [23].

El cuarto borrador de la especificación publicó el 14 de diciembre de 2004. Los cambios que incorpora están representados por la inclusión de dos elementos estructurales , uno de ellos para hacer extensible la anotación que especifica la información del vendedor o de la aplicación; y el otro para conducir la interacción multimodal, permitiendo indicar el punto de entrada de la interacción multimodal.

El último borrador de la especificación se dio a conocer el 16 de septiembre de 2005. En él, los elementos estructurales son clasificados en: Elemento raíz, elemento de interpretación, elementos contenedores (uno de, grupo y secuencia), elemento látice y elemento literal semántico. Por otra parte se ha agregado el elemento emma:derivation que sirve de contenedor para etiquetas EMMA que describen escenarios de derivación. Además se especifica como se utiliza el timestamp relativo con látices.

Se espera que este lenguaje sea usado inicialmente como formato estándar para el intercambio de datos entre componentes de sistemas multimodales, en particular será generado automáticamente por los componentes de interpretación que representan la semántica de las entradas del usuario. El lenguaje se centra en anotar la información de interpretación de una entrada simple o compuesta, y no en la en la información que se debe capturar en el curso del diálogo. Para ello el lenguaje provee un conjunto de elementos y atributos que se centran en la representación detallada de la anotación sobre la interpretación de la entrada [75].

Un documento EMMA puede soportar tres tipos de datos:

Datos de instancia: Corresponden a la información de entrada de etiquetas de aplicaciones específicas, y son construidas por procesos de entrada en ejecución . Están contenidos en la interpretación EMMA. Debido a que las alocuciones pueden ser ambiguas con respecto a los valores de entrada, un documento puede soportar más de una instancia.

Datos del modelo: Representan restricciones sobre la estructura y el contenido de una instancia. Generalmente son preestablecidos por la aplicación y pueden ser implícitos. Opcionalmente son especificados como anotaciones de una instancia.

Metadatos: Son anotaciones asociadas al contenido de los datos en las instancias. Los valores de la anotación son agregados en el momento de la entrada en tiempo

de ejecución. Las anotaciones EMMA pueden ser aplicadas en cualquier nivel del documento EMMA.

Teniendo en cuenta la naturaleza de la representación de los datos explicada, se aplicaron los siguientes principios en el diseño de EMMA:

- El contenido principal de la especificación EMMA consiste en metadatos: EMMA provee un medio para especificar anotaciones de metadatos que requieren estandarización.

- Se asume que los datos de instancia y de modelo se especifican en XML

- La extensibilidad de EMMA radica en la habilidad de agregar clases de metadatos para incluir vocabularios específicos en aplicaciones y expresarlos con la misma plataforma.

El principal propósito de EMMA es representar la información automáticamente extraída por de un componente de interpretación, desde una entrada de usuario tomada en forma general, en cualquier modalidad soportada por la plataforma. Algunos componentes que generan EMMA son: Reconocedor de voz, reconocedor de escritura, mecanismos de entendimiento del lenguaje natural, interpretadores de otros mecanismos de entrada (DTMF, punteros, teclado), componentes de integración multimodal. El Administrador de interacción multimodal, y el componente de integración multimodal usan EMMA.

El modelo de datos de EMMA expresa las restricciones sobre la estructura y contenido de los datos de instancia, para fines de validación. El modelo de datos puede ser considerado como una clase particular de anotación, a diferencia de las anotaciones EMMA, no es una característica perteneciente a una entrada específica del usuario en un momento determinado.

2.3.10 SMIL (Synchronized Multimedia Integration Language)

Es un lenguaje basado en XML que permite escribir presentaciones multimedia interactivas, describiendo su comportamiento temporal, asociando hipervínculos con objetos multimedia y describiendo los diseños de presentación en una pantalla. Generalmente es usado para enriquecer presentaciones multimedia integrando salidas de audio y video con imágenes, texto y otros tipos de medios.

Desde 1995, en la cuarta conferencia de la WWW, Philipp Hoschka plantea la dificultad al integrar en la web salidas multimedia en tiempo real, coordinado la salida de datos con la presentación en la pantalla. En un primer intento por solucionar esta dificultad se crea en 1996 el primer grupo SYMM WG (Working

Group on synchronized multimedia), dedicado al diseño de un lenguaje declarativo que permita la sincronización de medios. Es así como el 6 de noviembre de 1997, el W3C publica el primer borrador de SMIL [89], brindando la posibilidad de presentar contenidos en Internet como si fuera por televisión, evitando las limitaciones producidas por el ancho de banda requerido para este tipo de transmisión.

En 1998 se presentaron la primeras implementaciones de SMIL, dentro de las que se destaca HPAS (Hypermedia Presentation and Authoring System), sistema que permite integrar y administrar documentos hipermedia, permitiendo al usuario de disfrutar contenidos multimedia en Internet. Ese mismo año también se presenta GriNS Pro, el primer editor de SMIL.

En febrero de 1999 se establece el segundo grupo SYMM WG, el cual publica en septiembre del mismo año las características de acceso de SMIL [57], y en agosto de 2001 la especificación de SMIL 2.0 que se convierte en Recomendación del W3C. Su objetivo principal ha sido extender el desarrollo de SMIL como un lenguaje declarativo, basado en XML que permite la coordinación y sincronización cubriendo las siguientes áreas de funcionalidad: animación, control de contenidos, manejo de diseños, enlaces, integración de medios, metainformación, estructuras, coordinación y sincronización, manipulación del tiempo y efectos de transición.

A partir de abril de 2004 el grupo comienza a trabajar en la versión 2.1, cuyo primer borrador fue publicado el primero de febrero del presente año, presentando como principal novedad la definición de un perfil móvil que incorpora características especiales para la industria móvil, dentro de las que se pueden destacar [95]:

- Una función para especificar los fondos de imágenes por región.
- Alineamiento simple de objetos con una región sin usar el módulo de diseño completo, lo cual es útil para imágenes centradas en una región.
- Transición de sonido similar a la forma de transición visual, para transiciones simples como la pérdida de intensidad (atenuación)
- Las transiciones pueden trabajar con múltiples objetos
- Atributos para regiones y elementos de los objetos multimedios, como familias de fuentes, tamaños de la fuente y color.
- Un nuevo atributo para manejar escalas de objetos menores al 100%. Una imagen puede ser reducida pero no ampliada más allá del tamaño de la región.

Además de esto, a nivel de diseño los principales objetivos que se establecieron para la versión 2.1 fueron [17]:

Definir un lenguaje basado en XML que permita escribir presentaciones interactivas multimodales. Usando SMIL, el autor puede describir el comportamiento temporal de una presentación multimedia, asociando hipervínculos con objetos de medios de comunicación y describiendo el diseño de la presentación en la pantalla.

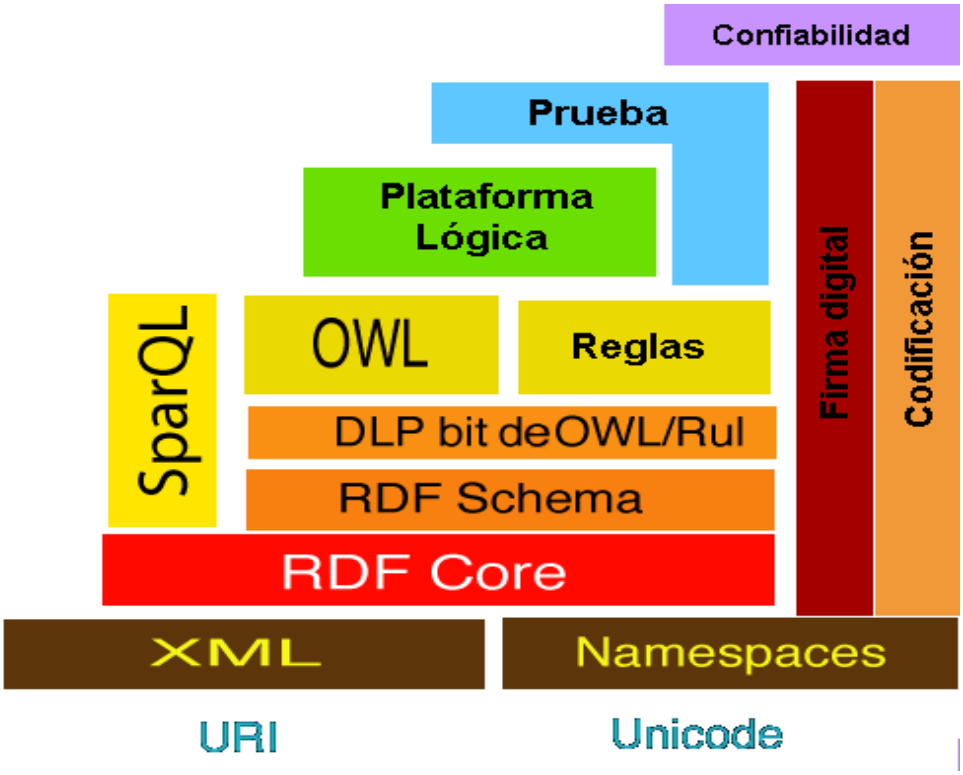
Permitir reutilizar la sintaxis de SMIL y su semántica, en otros lenguajes basados en XML, en particular todos aquellos que necesitan coordinar y sincronizar la presentación.

La especificación se encuentra estructurada en un conjunto de secciones, cada una de las cuales representa uno de los siguientes módulos: De animación, control de contenido, diseño, enlace, medios de comunicación, metainfomación, estructura, coordinación y sincronización, manipulación del tiempo, efectos de transición. También se definen cuatro perfiles que son construidos usando los módulos previamente nombrados: Perfil del lenguaje, perfil móvil, perfil móvil extendido, perfil básico y plataforma escalable.

2.4 ESTUDIO SOBRE LA CAPA LÓGICA Y ONTOLÓGICA DE LA WEB SEMÁNTICA

El estudio de las capas de la Web Semántica nos proporciona una comprensión amplia de la misma. El modelo clásico de capas, presentado por Tim Berners-Lee en el W3C [12] o su modelo mejorado [13], está basado en la jerarquía de los lenguajes, donde cada uno presenta y explota sus capacidades dejando a la capa siguiente que cubra sus dificultades.

Figura Nº 23 Último modelo de Capas de la Web Semántica.



Fuente: Berners-Lee, Tim. Web for real people.[en línea] < http://www.w3.org/2005/Talks/0511-keynote-tbl/#[17]>

Cada capa plantea el uso de sus recursos disponibles, desde los elementos básicos del entorno Web hasta la certeza de una interface con el usuario o agentes inteligentes. Dentro de esta estructura, las capas ontológicas y lógicas constituyen el núcleo de cualquier aplicación.

2.4.1 Capa RDF + RDFSchema

Hacia la base de la pirámide y soportando la capa ontológica se encuentra el RDF + RDFSchema. Ambas aportan en el modelo semántico la descripción mínima de recursos. Esto incluye el establecimiento de clases y subclases, propiedades y subpropiedades con sus correspondientes dominios y rangos, comentarios y etiquetas. Todos los elementos esenciales que garantizan la interoperabilidad entre agentes o sistemas.

Sus componentes se pueden presentar en 3 categorías:

- Recursos

- Propiedades
- Literales

Mientras el RDF constituye la proposición de hechos, por ejemplo, las *Personas* tienen una *Identificación*, *Nombre*, *Apellidos*, un lugar de nacimiento que los hace *Naturales* del mismo, entre otros.

Camilo Jaramillo Díaz es un *Persona*
Montes de María, Bolívar es un *Territorio de Guerra*
Camilo Jaramillo Díaz viene Desplazado de *Montes de María, Bolívar*
Arroz Barato es un *Asentamiento*
Camilo Jaramillo Díaz vive en *Arroz Barato.*

El RDFSchema establece las definiciones del vocabulario

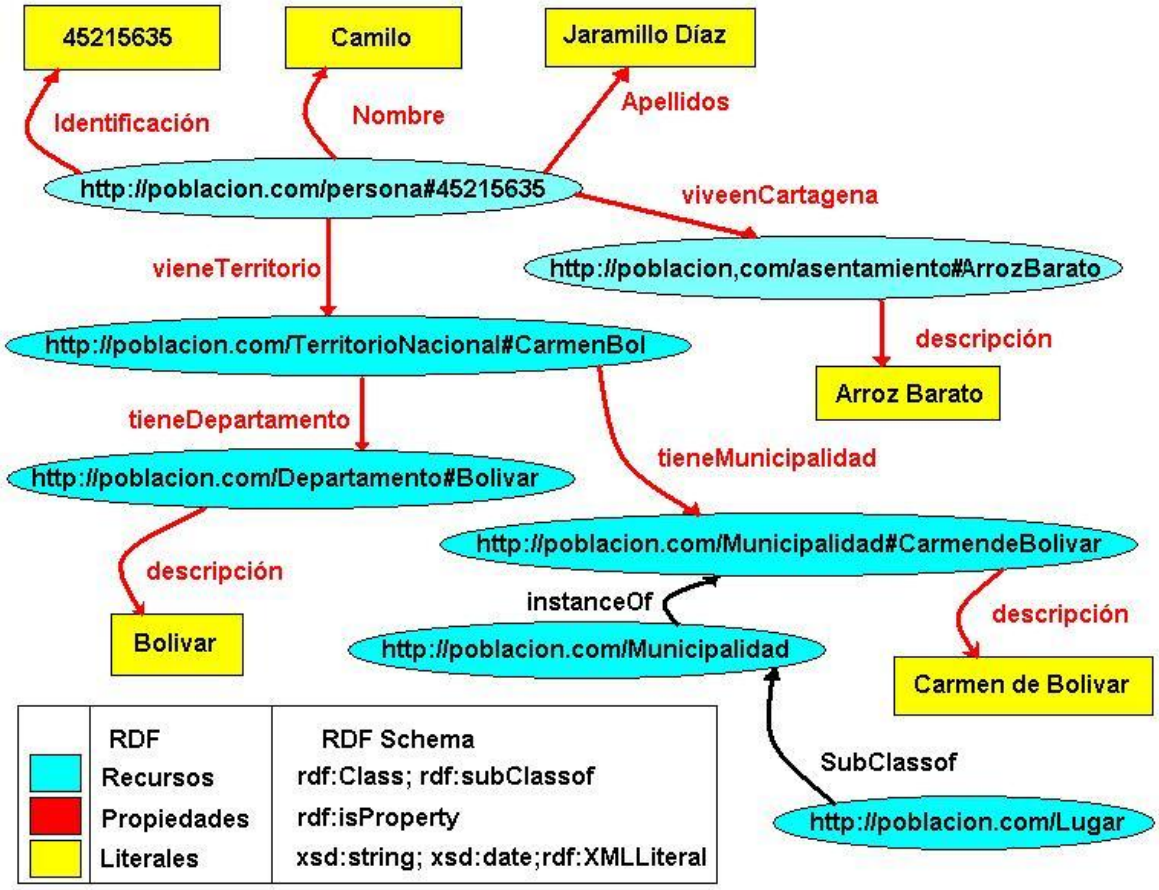
- *Persona* y *Desplazado* son clases. *Desplazado* es una subclase de *Persona*
- *Territorio de Guerra* es una clase.
- *Identificacion, Nombre, Apellidos* son atributos (Slot) de tipo cadena de caracteres (xsd:string)
- Identificación: 45478459, Nombre: Camilo, Apellidos: Jaramillo Díaz es una instancia de la clase Persona.
- Montes de María, Bolívar es una instancia de Territorio de Guerra
- vieneTerritorio es una propiedad que se aplica a instancias de la clase Persona (Dominio) sobre instancias de la clase TerritorioNacional (Rango)
- viveAsentado es una propiedad que se aplica a instancias de la clase Persona (Dominio) sobre instancias de la clase Asentamiento

Esta estructura, simple y rudimentaria, es suficiente para representar diversas y múltiples áreas del conocimiento o de hechos. Kuchling [58] define como la mayor ventaja del RDF que es descentralizado, lo cual significa:

- Cualquiera puede crear un vocabulario
- Cualquier puede utilizar y publicar información de otros recursos

Sin embargo denota como su desventaja principal que su especificación es difícil de verbalizar y extremadamente tediosa para ser desarrollada sin herramientas

Figura Nº 24. RDF + RDFSchema



2.4.2 Capa Ontológica

El RDF deja mucho vacíos para la represtación del conocimiento hacia la Web Semántica. El RDF nos permite responder las preguntas: ¿cuáles clases existen? ¿cuáles propiedades tiene una clase?, pero otras preguntas son imprescindibles para la WS, tales como:

- ¿Cuándo dos clases son idénticas? ¿En qué difiere un individuo vulnerable de uno normal? ¿Puede un individuo desplazado ser vulnerable?
- ¿Puede una propiedad tener valores múltiples? ¿Qué sucede si el valor de la propiedad *Naturalde* se desconoce?
- ¿Existe transitividad o simetría a través de las propiedades?

La capa Ontológica, estructurada por el OWL bajo el esquema del W3C, especifica una serie de relaciones importantes para la representación conocimiento:

- Para Clases
    - Equivalencias
    - Dis-junciones
- Para Propiedades
    - Equivalencias
    - Inversas
    - Transitividad
    - Simetría
    - Cardinalidad
- Para Recursos
    - Igualdad
    - Diferencias

La definición de los elementos que conforman esta capa provee en primer lugar un volumen superior de meta-información, entre las ya mencionadas y otras como unicidad y eliminación de ambigüedades, además de un espectro más amplio de funcionalidad, interoperabilidad y reutilización.

Elementos Básicos

Encabezado OWL

La especificación del OWL [73] es un componente del grupo de WS del W3C. Su esfuerzo está dirigido a hacer los recursos de la Web más legibles para los procesos automáticos adicionando información acerca de los recursos que describe o proporciona. Por su esencia distribuida, OWL debe propiciar que las ontologías sean relacionadas o implícitamente incluidas desde otras ontologías. Adicionalmente las ontologías no están enmarcadas en un único archivo, por lo que las clases pueden extenderse en otros, con la restricción de que sus elementos definitorios no pueden ser eliminados aunque si pueden contener información contradictoria. Cabe destacar que OWL es extensión del vocabulario de RDF.

Esta extensión comienza desde el mismo encabezado. El OWL permite importar ontologías. Un ejemplo clásico es el conjunto de ontologías y definiciones aportados por el Dublin Core Group.

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
```

```
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns="http://www.owl-usab.edu.co/poblacion.owl#"
   xml:base="http://www.owl-usab.edu.co /poblacion.owl">
   <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://purl.org/dc/elements/1.1/"/>
   </owl:Ontology>
   ...
  </rdf:RDF>
```

La etiqueta owl:imports proporciona un mecanismo de inclusión, parecido al de los lenguajes de programación. Esta toma el argumento identificado como recurso del RDF. Importar ontologías facilita el compartimiento de información y conocimiento. Es recomendable que los elementos importados sean identificados con un *Namespace* diferente.
Este objeto importado aporta un amplio número de propiedades de anotación de las cuales mencionaremos sus características posteriormente.

Clases

En la capa ontológica las clases tienen una reimplementación. Las clases OWL son interpretadas como conjuntos que contienen individuos directa o indirectamente. Su descripción incluye los requerimientos que se deben cumplir para ser miembros de ella. Desde el RDF, las clases se organizan en una jerarquía superclase-subclase, denominada taxonomía. Esta generalización-especialización implica que los individuos de una subclase son miembros de la superclase y pueden ser analizados por cualquier razonador que procese OWL-DL.

En la capa ontológica descrita según OWL todas las clases tienen una superclase. La clase owl:Thing está por defecto en el tope de la taxonomía. El planteamiento de una clase es muy simple:

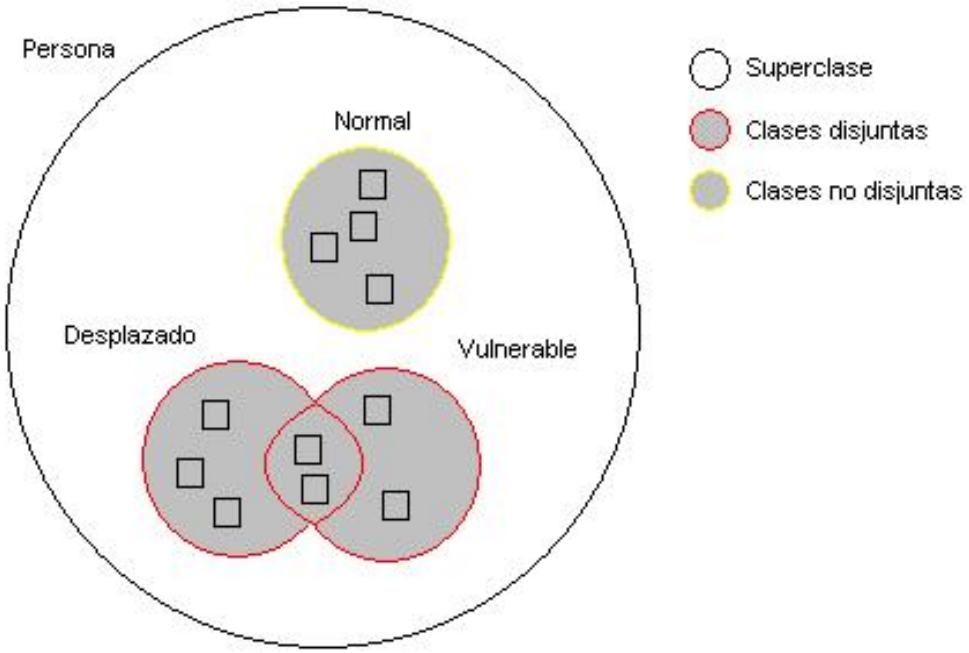
```
<owl:Class rdf:ID="Lugar"/>
<owl:Class rdf:ID="Normal">
<rdfs:subClassOf rdf:resource="#Persona" />
      ...
</owl:Class>
```

Todas las clases deben estar identificadas, la identificación es la esencia de la reutilización y de las ontologías distribuidas. Por ejemplo si una ontología quiere utilizarse en otra para enriquecerla, es suficiente con nombrarla en un *Namespace* y hacer referencia a sus elementos.

```
<rdf:RDF
   xmlns:pob="http://www.owl-usab.edu.co/poblacion.owl#"
   ... >
```

Con esto se puede hacer referencia a las clases de una ontología distribuida mediante la expresión pob:Persona o el valor de atributo &pob;Persona o por el valor del URI completo http://www.owl-usab.edu.co/poblacion.owl#Poblacion

Figura Nº 25. Diagrama de Venn para taxonomía *Persona*



En la capa ontológica se puede especificar que dos o más clases son disjuntas (owl:disjointWith). Esto significa que los individuos de una clase A disjunta de una clase B no pueden ser instancias ni incluso por inferencia.

```
<owl:Class rdf:ID="Normal">
  <rdfs:subClassOf rdf:resource="#Persona"/>
  <owl:disjointWith>
    <owl:Class rdf:ID="Desplazado"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="Vulnerable"/>
  </owl:disjointWith>
  ...
</owl:Class>
```

Individuos

Los individuos son los miembros de una clase. Ellos representan el dominio de interés. Los individuos pueden ser todos diferentes o pueden manejarse varios individuos que hacen referencia al mismo elemento del dominio.

Propiedades

Las taxonomías aunque presentan por si solas un entendimiento del conocimiento, son las propiedades las que permiten establecer los hechos del dominio de conocimiento.

Las propiedades representan relaciones entre dos individuos. En la capa ontológica se disponen de dos propiedades principales: *Object Properties* y *Datatype Properties*, y una tercera que enriquece la capa: *Annotation Properties*.
Las propiedades de objetos enlazan dos individuos. Estas propiedades tienen una redefinición en el OWL facilitando la inferencia de información. Las Object Properties tienen un dominio y un rango. El dominio representa la clase de los individuos que “tienen” la propiedad y el rango representa la clase de individuos que son los “valores” posibles de la propiedad.

```
...
<owl:ObjectProperty rdf:ID="tieneDesplazado">
  ...
  <rdfs:range rdf:resource="#Persona"/>
  <rdfs:domain rdf:resource="#TerritorioNacional"/>
</owl:ObjectProperty>
...
<TerritorioPaz rdf:ID="Barranquilla-Atlantico">
      <tieneDesplazado>
        <Persona rdf:ID="C45471431">
          ...
        </Persona>
        ...
      </tieneDesplazado>
      ...
    </TerritorioPaz>
```

Una propiedad de objeto puede ser además subpropiedad de otra, posibilitando que exista una jerarquía de propiedades.

Las propiedades de datos enlazan un individuo con un valor o literal definido en el esquema RDF. Sus tipos están predefinidos.

Tabla. XML Schema datatypes

Tipos de Cadena	Tipos Numéricos	Tipos enumerados y de Fecha
xsd:string	xsd:decimal	xsd:boolean
xsd:normalizedString	xsd:integer	xsd:token
xsd:Name	xsd:nonPositiveInteger	xsd:NMTOKEN
xsd:NCName	xsd:long	rdfs:Literal
xsd:language	xsd:unsignedLong	xsd:anyURI
	xsd:hexBinary	
	xsd:unsignedByte	xsd:date
	xsd:float	xsd:gDay
	xsd:unsignedByte	xsd:dateTime
	xsd:nonNegativeInteger	xsd:gYear
	xsd:negativeInteger	xsd:time
	xsd:int	xsd:gMonthDay
	xsd:unsignedInt	xsd:gYearMonth
	xsd:base64Binary	xsd:gMonth
	xsd:double	
	xsd:positiveInteger	
	xsd:byte	
	xsd:short	
	xsd:unsignedShort	

Las propiedades de anotación se utilizan para añadir información del tipo de metadatos, o sea, datos a cerca de datos. Un gran número de estas propiedades están estandarizadas por el Dublin Core Group.

```
...
<owl:AnnotationProperty rdf:ID="creadores">
  <rdf:type
     rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:AnnotationProperty>
...
<RDF:DESCRIPTION
        RDF:ABOUT="HTTP://WWW.W3.ORG/2002/07/OWL#THING">
   <creadores
      rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
     Martin Monroy
   </creadores>
   <creadores
      rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
     Carlos Ferriol
```

```
    </creadores>
  </rdf:Description>
  ...
```

Las propiedades tienen diferentes características que definen su comportamiento y su aplicación:

FUNCTIONALPROPERTY

Las propiedades funcionales establecen unicidad. Su definición lógica es: Sean *X*, *Y*, *Z* individuos | *P*(*X*,*Y*) y *P*(*X*,*Z*), entonces *Y* =*Z*. Un uso más simple es que se utilizan para aquellas propiedades de los individuos que solo pueden tomar un valor.

INVERSEOF

La definición de la característica de “inversa de”, requiere de dos propiedades. Esta característica se establece en propiedades de objetos. Su definición lógica es: Sean *X*, *Y* individuos | *P*(*X*,*Y*) <=> *Q*(*Y*,*X*)

TRANSITIVEPROPERTY

Esta característica indica el comportamiento bien conocido de la transitividad. Se establece en propiedades de objeto. Su definición lógica es: Sean *X*, *Y*, *Z* individuos | *P*(*X*,*Y*) y *P*(*X*,*Z*), entonces *P*(*X*,*Z*)

SYMMETRICPROPERTY

Esta característica se establece en propiedades de objetos. Su definición lógica es: Sean *X*, *Y* individuos | *P*(*X*,*Y*) <=> *P*(*Y*,*X*)

INVERSEFUNCTIONALPROPERTY

Su definición lógica es:

Sean *X*, *Y*, *Z* individuos | *P*(*Y*,*X*) y *P*(*Z*,*X*), entonces *Y* =*Z*

```
  ...
  <owl:ObjectProperty rdf:about="#viveTerritorio">
    <rdfs:domain rdf:resource="#Persona"/>
    <rdfs:range rdf:resource="#TerritorioNacional"/>
    <owl:inverseOf rdf:resource="#tieneHabitantes"/>
    <rdf:type
     rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  </owl:ObjectProperty>
```

```
...
<owl:FunctionalProperty rdf:about="#vieneTerritorio">
  <rdfs:range rdf:resource="#TerritorioNacional"/>
  <rdfs:domain rdf:resource="#Persona"/>
  <rdf:type
    rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <owl:inverseOf rdf:resource="#tieneDesplazado"/>
</owl:FunctionalProperty>
```

Si bien estas características son importantes, la capa ontológica no podría hacer mejor preludio a la capa lógica que con las restricciones. Las restricciones de las propiedades son la base de las inferencias, de la clasificación y del chequeo de inconsistencias que sobre la ontología hacen los razonadores. Las restricciones se aplican a una clase y los individuos que estén asociados a la clase deben cumplirlas so pena de tratarse de una ontología inconsistente.

Las restricciones son:

ALLVALUESFROM (∀), SOMEVALUESFROM (∃)

Estas restricciones establecen que los individuos de la clase que tengan valor en la propiedad deben tener todos (∀) o al menos un (∃) miembro del rango de la propiedad. Estas restricciones son las más comunes.

CARDINALITY

La restricción de cardinalidad restringe cantidad mínima o máxima de elementos del rango de la propiedad que los individuos de la clase deben tener.

HASVALUE

La restricción de tener valor representa que los individuos deben tener la propiedad con un valor establecido.

Con estas restricciones podemos incluir en la capa ontológica dos elementos importantísimo que son las condiciones necesarias y las condiciones necesarias y suficientes. En la lógica una condición necesaria y suficiente (`owl:equivalentClass`) establece requisitos de obligatorio cumplimiento, mientras una condición necesaria establece premisas (rdfs:subClassOf).

```
...
<owl:Class rdf:ID="Normal">
  <owl:disjointWith>
    <owl:Class rdf:ID="Vulnerable"/>
```

```
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="Desplazado"/>
  </owl:disjointWith>
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:someValuesFrom rdf:resource="#TerritorioPaz"/>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="viveTerritorio"/>
        </owl:onProperty>
    </owl:Restriction>
  </owl:equivalentClass>
  <rdfs:subClassOf rdf:resource="#Persona"/>
</owl:Class>
...
<owl:Class rdf:about="#TerritorioNacional">
  <owl:disjointWith rdf:resource="#Persona"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality
        rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
        1</owl:cardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="tieneDepartamento"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  ...
</owl:Class>
```

Mas elementos

Otros elementos y los ya descritos, conllevan a una capa ontológica clara, legible.

Tabla Nº 2. Otros elementos del OWL

equivalentProperty
sameAs
differentFrom
AllDifferent
Operadores de conjunto: *intersectionOf*, *unionOf*, *complementOf*
Clases enumeradas: *oneOf*

La capa ontológica determina reglas y reseña el comportamiento de los individuos del dominio del conocimiento representado, pero, ¿cómo inferir? ¿cómo hacer válidas las reglas? ¿quién determina si la ontologías es consistente?

2.4.3 Capa Lógica

La capa lógica consiste en las inferencias que se pueden lograr a partir de un conjunto de reglas. Estas inferencias deben permitir responder a preguntas u obtener resultados que no se encuentran implícitos en los datos [16]. En este punto se tiene que las afirmaciones deben ser utilizadas para generar nuevo conocimiento. La capa lógica nos provee de lenguajes interoperativos que toman colecciones de datos y lo transforman en nuevos hechos.[28]

Esta capa tiene como función la de ser capaz de cumplir con una lógica monotónica. Lo cual implica que las inferencias siguen siendo válidas aún cuando se incluyen nuevos individuos a la ontología. Algunos investigadores comienzan a contradecir este planteamiento, mientras otros lo defienden con argumentos como: "La lógica no-monotónica cambia las reglas lógicas para acomodarse al estado actual del conocimiento. La lógica monotónica establece que solo si se cambia de parecer, la situación actual puede ser inconsistente." [74]

La lógica monotónica tiene la ventaja que en las capas siguientes los agentes razonan a partir de conclusiones tomadas de otros agentes sin cuestionarlas.

Siendo la capa ontológica especificada en un lenguaje de descripciones lógicas (OWL-DL), los cuales son elementos de lógica de primer orden, s pueden construir aplicaciones que obtengan resultados a partir de interrogantes un un número finito de pasos. Esta herramienta debe ser al menos capaz de determinar relaciones de herencia e inconsistencias (una clase inconsistente no puede tener instancias).

Desarrollar una herramienta específica para cada modelo del conocimiento no es una tarea fácil ni tendría sentido. Para beneficio de la comunidad científica se han desarrollado varias de estas herramientas. Estas constituyen el elemento principal de esta capa y se denominan Razonadores (*Reasoners)*

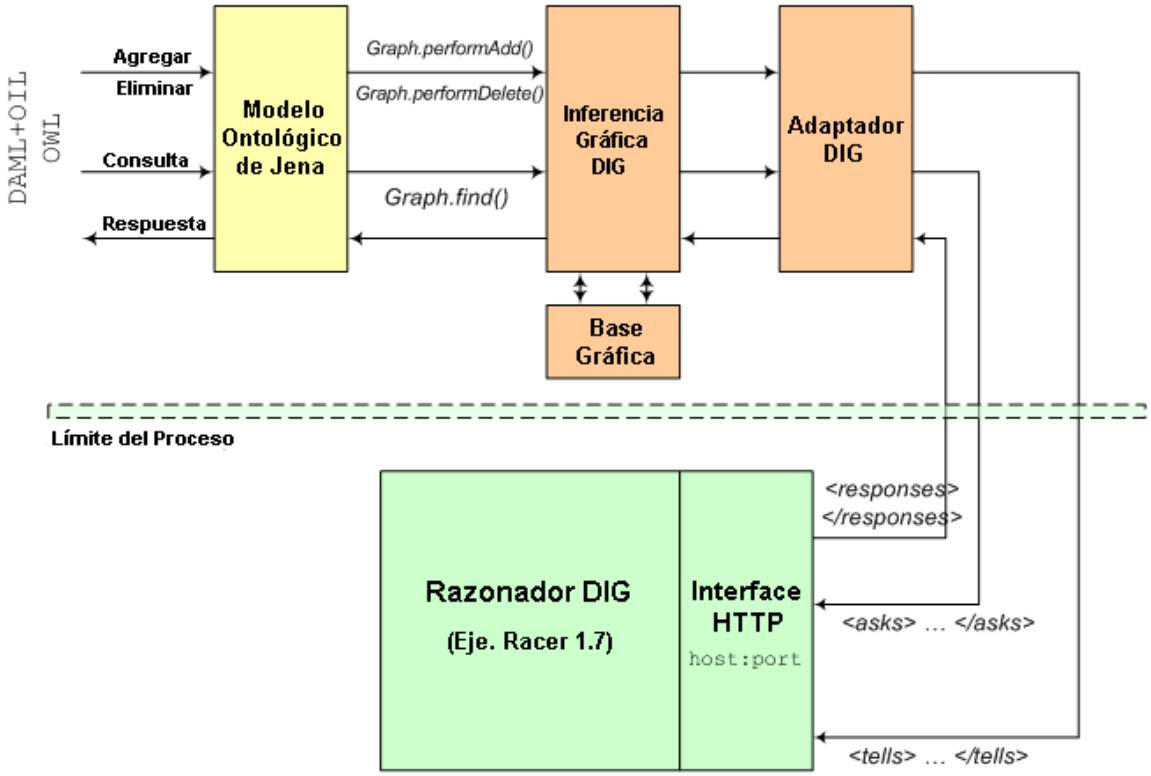
Varios razonadores tienen resultados importantes entre ellos se encuentran:

- Racer, http://www.racer-systems.com/
- FaCT, http://www.cs.man.ac.uk/~horrocks/FaCT/
- Pellet OWL, http://www.mindswap.org/2003/pellet/index.shtml

La mayoría de los razonadores suministran una interface la cual permite con un lenguaje bien definido obtener la información inferida. Racer por el contrario adopta una comunicación por *sockets* y una API neutral para realizar las

consultas. Estos razonadores se basan en las definiciones del *Description Logic Interface Group* (DIG) .

Figura Nº 26. Funcionamiento de un razonador DIG



Fuente: HOWTO use Jena with an external DIG reasoner. [en línea] <http://jena.sourceforge.net/how-to/dig-reasoner.html>

Los lenguajes de un razonador DIG son objetos XML con todas sus características. La ventajas de los DIG es que estos suministran medios de comunicación que se abstraen de los protocolos de interacción.

Una implementación muy simple es a través de una API. Estas constan de 4 pasos fundamentales:

1. La inicialización de la instancia del razonador
2. La introducción de la ontología
3. El inicio y establecimiento de la conexión con el razonador
4. La consulta

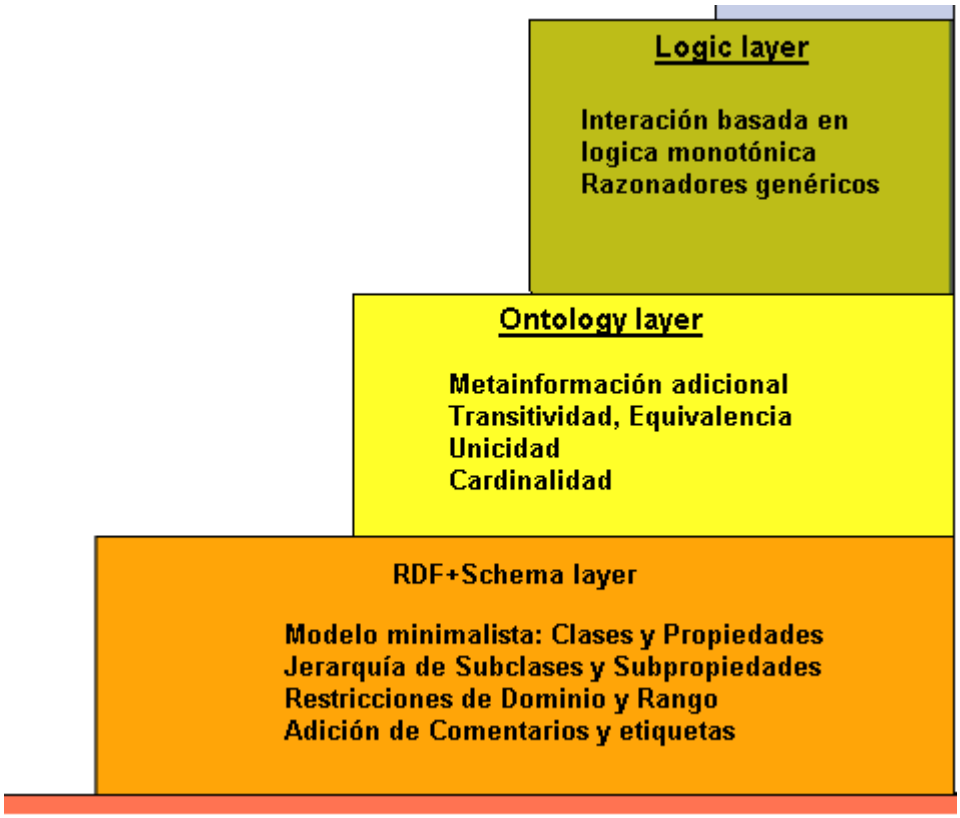
Para interactuar con el Racer a través de la *Protegé OWL Reasoning* API los pasos son:

```
// Paso 1
ReasonerManager reasonerManager = ReasonerManager.getInstance();
// Paso 2
ProtegeOWLReasoner reasoner =
        reasonerManager .getReasoner(OWLmodel);
// Paso 3. Inicio de la conexión
reasoner.setURL(REASONER_URL);
if(reasoner.isConnected()) {
  // Paso 3. Establecimiento de la conexión
  DIGReasonerIdentity reasonerIdentity = reasoner.getIdentity();
  // Paso 4. Consulta
  OWLNamedClass Normal =
        (OWLNamedClass) poblacion.getNormalClass();
  Collection inferredSubclasses =
        reasoner.getIndividualsBelongingToClass (Normal,null);
  System.out.println( "Number of inferred Persona: " +
        inferredSubclasses.size());
}
```

En este ejemplo, juega el papel fundamental en el razonamiento el método getIndividualsBelongingToClass (...). Mediante este se hace la inferencia y se obtienen las instancias que componen el resultado.

En resumen las capas ontológica y lógica de la Web Semántica son determinantes en la representación del conocimiento. No existe una sin la anterior aunque: “todo en última instancia es RDF” [12].

Figura Nº 27. Elementos de las Capas: RDF, Ontológica y Lógica



3. GUÍA PARA EL MODELADO Y DESARROLLO DE APLICACIONES WEB MULTIMODALES QUE INCLUYAN MODELADO DE DATOS BASADO EN WEB SEMÁNTICA

Esta guía tiene como fin proponer una serie de pautas ordenadas, que sirvan de apoyo a desarrolladores de aplicaciones web multimodales que incluyan modelado de datos basado en web semántica; específicamente en el momento de modelar y desarrollar la interface de usuario, y al definir el modelo de datos haciendo uso de RDF y OWL

Para lograrlo, se tienen en cuenta los siguientes aspectos:

1. El modelado y desarrollo son actividades típicas de un proceso de ingeniería de software, por lo tanto, se hace imperativa la selección de una metodología de construcción de software que defina cómo construir técnicamente la aplicación, realizando las actividades propias del ciclo de vida: análisis, diseño, construcción del programa, pruebas y mantenimiento. Aunque la guía es independiente de la metodología seleccionada, se recomienda utilizar el Proceso Unificado de Desarrollo de Software, por todas las ventajas que representa el desarrollo iterativo incremental, centrado en la arquitectura y dirigido por casos de uso.

2. Una aplicación web implementa la arquitectura cliente servidor, cuyo contexto de ejecución se caracteriza por ser concurrente, escalable y heterogéneo a nivel de hardware y software.

3. Por otra parte, las aplicaciones multimodales deben estar en capacidad de adaptarse a las características y limitaciones del dispositivo, las preferencias del usuario y las condiciones del entorno, permitiendo la interacción a muchos usuarios desde múltiples dispositivos, trabajando individual o conjuntamente, aumentando la interacción hombre máquina.

4. Adicionalmente, la web semántica propone el intercambio de datos que pueden ser compartidos y procesados igualmente en forma automática por herramientas de software o por personas.

En este orden de ideas, se hace evidente la necesidad de implementar una estrategia que permita aplicar los principios de modularidad e independencia funcional, que plantea la ingeniería de software, al realizar las tareas de modelar y desarrollar una aplicación web multimodal que incluya modelado de datos basado en web semántica.

Por lo tanto, se propone aplicar un modelo arquitectónico de múltiples capas, soportado en el patrón de diseño Modelo Vista Controlador MVC, ideal para aplicaciones que soportan distintos tipos de interface de usuario, y múltiples vistas sincronizadas del mismo modelo, propias del entorno multimodal. Además permite separar la capa de presentación de la lógica y los datos, en este caso modelado de datos basado en web semántica, haciendo posible trabajar en forma separada, mas no aislada, los aspectos propios de cada una de estas tecnologías; razón por la cual se ha estructurado la guía en dos partes partes: La primera destinada a definir el Modelo de la interacción multimodal, incluyendo la sinergía que se obtiene al fusionar el modelo de datos basado en web semántica con la interacción multimodal; y la segunda dedicada al Modelado semántico.

Este orden se plantea teniendo en cuenta, que se debe empezar con un idea clara de cómo se quiere la interface y como serán las interacciones con el usuario, para después desarrollar las especificaciones funcionales que sirvan de guía al diseño posterior.

Con el fin de ilustrar y facilitar el entendimiento de la guía, al terminarla, se presentará a nivel de ejemplo el modelado del prototipo de portal de información sobre el problema de desplazamiento forzado en la ciudad de Cartagena.

3.1 MODELADO DE LA INTERACCIÓN MULTIMODAL.

Se parte del supuesto de que se ha hecho el estudio correspondiente para identificar los requerimientos de la aplicación, con el fin de centrar la atención exclusivamente en el modelado de la interface de usuario. Como referente se utiliza la arquitectura multimodal [3] propuesta a partir de la plataforma de interacción multimodal [14] que plantea el W3C, debido a que se ajusta a los principios de modularidad, independencia funcional, encapsulación y extensibilidad, lo que permite facilitar la tarea de modelado, al separar los procesos de entrada y salida y por lo tanto los componentes que lo conforman. De esta manera se obtiene una aplicación con un alto grado de cohesión y un bajo grado de acoplamiento, como lo propone actualmente la ingeniería de Software.

Además, es importante señalar que cada una de las pautas que se mencione a continuación, ha sido definida teniendo en cuenta los siguientes principios de diseño de interfaces [59] de usuario en general:

- Satisfacer las restricciones del mundo real
- Comunicarse en forma clara, concisa y consistente con el usuario
- Ayudar a recuperar los errores en forma rápida y eficiente
- Hacer sentir cómodo al usuario

Y específicamente para la interface de usuario multimodales [79]:

- Las modalidades múltiples deben ser sincronizadas
- La interacción multimodal debe ser agradable
- Las modalidades múltiples deben de compartir un estado de interacción común
- La interface multimodal debe ser predecible
- La interface multimodal debe adaptarse al entorno del usuario.

Acorde con lo expuesto anteriormente, a continuación se presentan las pautas para modelar los componentes de la interacción multimodal, sin pretender un orden lógico temporal a nivel de metodología, sino más bien, un orden estructural coherente con la plataforma de interacción multimodal. Para ello la guía trata inicialmente los aspectos generales, luego los relacionados con las funciones de la aplicación, el componente de sesión, Sistema y entorno, el administrador de interacción, los procesos de entrada, y finalmente los aspectos relacionados con los procesos de salida.

3.1.1 Aspectos generales:

Para construir el modelo de una aplicación multimodal es conveniente tener en cuenta los siguientes aspectos:

Definir el orden de interacción: A partir de los requerimientos establecidos para la aplicación, representados a través de casos de uso, se puede establecer el orden de interacción (flujos de diálogo para el caso de la voz) entre el usuario final y la interface. Para lograrlo se recomienda:

- Identificar la información que la aplicación requiere del usuario
- Determinar los comandos o acciones que el usuario puede realizar para interrumpir o terminar el caso de uso en ejecución
- Establecer el modelo del flujo de la interacción a través de un diagrama de secuencias.
- Definir la reacción de la aplicación ante las respuestas del usuario (correctas e incorrectas) o en ausencia de éstas.
- A partir de la información requerida, identificar la gramática que determinará la corrección de las entradas.
- Identificar las salidas de la aplicación.

Definir los recursos tecnológicos para la realización de la aplicación: Teniendo en cuenta la funcionalidad que realizará la aplicación, es conveniente seleccionar los recursos que se utilizarán para su desarrollo, ya que sus características influyen representativamente en el resultado obtenido. Los recursos que se deben seleccionar son los siguientes:

- Entorno de desarrollo: Actualmente existen dos entornos de desarrollo de aplicaciones multimodales que permiten la interacción por medio de la voz, uno propuesto por IBM denominado WebSphere Voice Server Software Development Kit, el cual forma parte de WebSphere® Voice Application Access (WVAA) [47], que implementa el estándar propuesto por el W3C, específicamente utilizando voiceXML. Y otro por Microsoft denominado Microsoft Speech Application Software Development Kit SASDK [69], el cual utiliza SALT y mantiene compatibilidad con la tecnología punto NET.
- Lenguajes de programación: Teniendo en cuenta el estudio del estado del arte de los lenguajes de marcado para el desarrollo de aplicaciones multimodales, hecho en la presente tesis, se recomienda utilizar los lenguajes propuestos por el W3C, en combinación con java, puesto que cuenta con interfaces de programación entre aplicaciones como: Java Speech Grammar Format JSGF, que permite la definición de gramáticas; Jena y Java RDF, las cuales han sido utilizadas para crear plataformas como Sesame, SNOBASE y Protege que facilitan el desarrollo de aplicaciones web semánticas.
- Servidor de aplicaciones: Para ser coherentes con el lenguaje de programación seleccionado, se considera conveniente el uso de Apache Tomcat.
- Navegador multimodal: se sugiere utilizar Opera por que incorpora la tecnología de voz desarrollada por IBM, la cual, al igual que el navegador puede ser utilizada sin incurrir en ningún costo.

3.1.2 Funciones de la aplicación

Según el modelo MVC, la funcionalidad de la aplicación está representada por el modelo, que como se dijo anteriormente corresponde al modelo semántico que será analizado detalladamente en la segunda parte de la presente guía, permitiendo ver la sinergia que existe entre la interacción multimodal y la web semántica.

3.1.3 Componente de sesión

El componente de sesión es de vital importancia para aplicaciones distribuidas que involucran más de un dispositivo y / o usuario, por que define el mecanismo que permite sincronizar la captura y salida de la información a partir de las múltiples modalidades, manteniendo el control temporal y la persistencia de la sesión, razón por la cual en el momento de diseñarlo y crearlo se sugiere:

- Definir si el contexto del problema requiere de entradas y / o salidas que involucren más de una modalidad al mismo tiempo.

- Definir si el contexto del problema requiere que se permita guardar una sesión en un dispositivo, para posteriormente continuarla en otro similar o de características distintas.
- Identificar los dispositivos que podrán ser usados para que la aplicación se ejecute, dependiendo las características en los dos puntos anteriores.
- Para cada dispositivo identificar si permite múltiples modos en forma secuencial (uno solo en un determinado momento) o concurrente.
- Establecer una estrategia de sincronización para los datos que permitan capturas concurrentes a partir de múltiples modalidades.
- Establecer una estrategia de sincronización para los recursos compartidos por varios usuarios.

3.1.4 Sistema y Entorno

Los componentes de entorno y sistema permiten controlar la interacción, para encontrar la salida y responder a los cambios dependiendo de las capacidades de los dispositivos, preferencias del usuario y condiciones del entorno, características que son analizadas por el W3C a través del grupo Device independence Work Group, cuyo principal objetivo es permitir el acceso a la web en forma unificada, desde cualquier dispositivo y en cualquier contexto, lo cual complementa el trabajo del Grupo de Interacción Multimodal, como se puede apreciar en el documento Device Independence, Accessibility and Multimodal Interaction [99]. La construcción de éste componente se encuentra fuera del alcance del presente proyecto.

3.1.5 Administrador de interacción

El administrador de interacción es un componente lógico que coordina datos y controla el flujo de ejecución, a partir de las diferentes interfaces de los componentes de las modalidades de entrada y salida. También es el responsable de mantener el contexto de la aplicación. La interacción está definida por las reacciones de la aplicación, que se han identificado previamente (ver aspectos generales), ante las respuestas del usuario, ya sean correctas e incorrectas, o en ausencia de éstas. También se deben tener en cuenta los comandos o acciones que el usuario puede realizar para interrumpir o terminar el caso de uso en ejecución

Según el modelo MVC, es fácil identificar que el administrador de interacción cumple el papel de controlador, por lo cual, se propone construirlo haciendo uso de XHTML + XForms, SVG y SMIL, en combinación con servlets de java y scripts. Los servlets por que permiten crear contenidos web dinámicamente como respuesta a una petición del cliente, manteniendo el control del flujo de ejecución y

coordinando el acceso a los datos, realizando sólo las validaciones pertinentes en el lado del servidor, facilitando el manejo de las sesiones al hacer uso de la interface HttpSession, que permite guardar y recuperar objetos por su nombre, permitiendo el control del contexto de la aplicación. Mientras que los scripts permiten realizar las validaciones típicas del lado del cliente, mejorando su rendimiento.

Adicionalmente se recomienda hacer uso de las características propias del lenguaje multimodal que se esté utilizando. Por ejemplo en el caso de XHTML + Voice, se pueden identificar las ausencias de respuestas por parte del usuario, utilizando el elemento <prompt> con su atributo timeout, junto con el elemento <noinput>.

3.1.6 Componente de entrada:

Al combinar el modelado semántico con la interacción multimodal, surge la posibilidad de enriquecer el mecanismo de control de diálogo, al permitir generar interfaces multimodales inteligentes capaces de corregir y mejorar el proceso de reconocimiento de la modalidad de entrada [80], a partir de consultas específicas que se hacen al modelo semántico representado a través de la ontología.

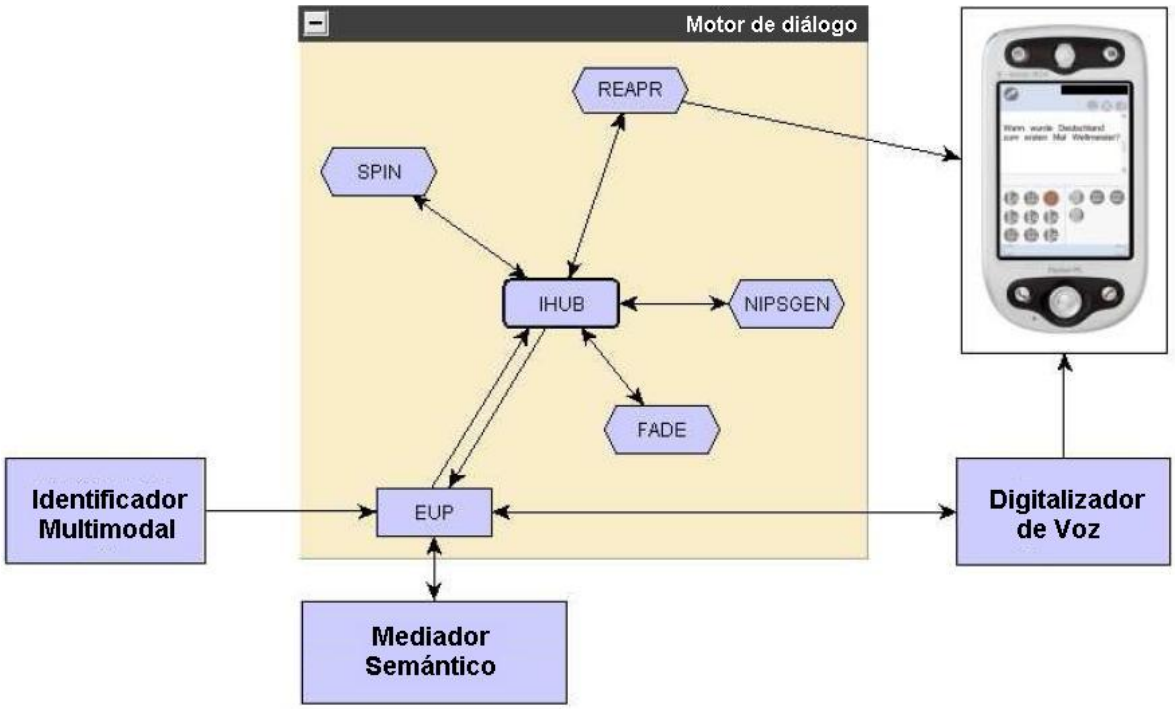
Una implementación de éste planteamiento se encuentra documentada por Norbert Reithinger y Daniel Sonntag, en su artículo “Plataforma de integración para un sistema de diálogo móvil multimodal con acceso a la Web Semántica” [80], en el que proponen un mecanismo de control de diálogo capaz de idenficar y corregir errores en las alocuciones del usuario, antes de pasarlas a la capa del modelo semántico de la aplicación.

Teniendo en cuenta dicha propuesta, un sistema normal de diálogo en lenguaje natural realiza los procesos de reconocimiento, análisis, reacción, generación y sístesis; una vez iniciado el proceso la información se propaga hasta el final, lo que significa que la interacción se reduce a mensajes por parte del usuario y resultado por parte del sistema. Esto exige al usuario realizar su entrada en forma correcta, o de lo contrario tendrá que esperar hasta obtener el resultado para hacer la corrección correspondiente, lo que va en contra de los principios de diseño de interfaces de usuario, puesto que no hace sentir comodo al usuario final.

La solución implica darle la posibilidad al usuario de corregir su entrada en cualquier momento. Para lograrlo, **Reithinger y Sonntag proponen un mecanismo de control de diálogo (Ver Figura 28), que se explica a continuación para ser tenido en cuenta al momento de modelar el componente de entrada.**

La comunicación con el medio externo es responsabilidad del componente empaquetador y desempaquetador EMMA (EUP), quien también se encarga de la comunicación con los módulos Identificador de Modalidad y Mediador Semántico, los cuales a su vez, proveen las inteface hacia la Web Semántica y la digitalización de la voz, respectivamente. El proceso comienza con el componente de interpretación (SPIN), cuya entrada está representada por las N – mejores cadenas de palabras, y su salida es una cadena de palabras analizada sintácticamente que contiene una descripción RDF/S de las partes analizadas de la cadena.

Figura Nº 28. Mecanismo de Control de diálogo



Fuente: **http://www.dfki.de/~bert/interspeech2005.pdf**

El componente de fusión de modalidad y discurso (FADE) sigue el discurso progresivo, completa los tipos diferentes de anáfora, y une la entrada de diferentes modalidades. Su entrada es una cadena de palabras analizada sintácticamente, y la descripción del gesto en un nivel ontológico con su posición geométrica. Su salida es una cadena de palabras aumentada, junto con la descripción ontológica completada de la expresión de usuario.

El componente de reacción y representación (REAPR) decide que hace con la interpretación de la entrada del usuario. El toma todas las entradas desde los componentes de interpretación (SPIN) y de fusión de modalidad. Su resultado se presenta en la pantalla para que pueda ser editado por el usuario. Si el usuario realiza cambios, el mensaje es enviado de nuevo al componente de interpretación.

En caso contrario, la solicitud del usuario es enviada al mediador semántico. Los resultados obtenidos por el mediador semántico son preparados para ser presentados y enviados a la capa que contiene la lógica de la aplicación (Modelo)

Si hay una representación semántica como respuesta, el módulo generador de texto (NIPSGEN), genera el texto que es enviado al Digitalizador de Voz. Todos los mensajes son dirigidos a través del Componente Integrador (IHUB), responsable de conocer los requerimientos programados entre los subcomponentes y las direcciones de los mensajes particulares. Dicha comunicación interna entre componentes está basada tecnicamente en el intercambio de objetos contenidos en la información ontológica, haciendo uso de la plataforma Jena.

Por otra parte, según la plataforma de interacción multimodal, el componente de entrada está conformado por tres categorías: De reconocimiento, De interpretación y de integración. Para cada una de ellas a continuación se presenta una serie de pautas que facilitan el modelado y la construcción de una aplicación multimodal.

De reconocimiento: Teniendo en cuenta que es la categoría encargada de la captura de la entrada natural, transformando su formato para el posterior procesamiento, se recomienda:

- Identificar la información que la aplicación requiere del usuario. Esto se logra a partir de los casos de uso que implemente la aplicación multimodal.
- Identificar las características del sistema y entorno de ejecución que requiere la aplicación multimodal, para establecer las ventajas que proporciona, y sus correspondientes limitantes.
- Identificar las posibles modalidades de entrada, dependiendo de las características del sistema y entorno de ejecución.
- Dependiendo las modalidades identificadas, seleccionar los correspondientes lenguajes de marcado para la interacción multimodal.
- Para cada uno de los datos requeridos definir la gramática, con sus respectivas restricciones.

En el caso de la interacción por medio de la voz, es conveniente utilizar Java Speech Grammar Format, ya que permite la combinación con scripts, lo que facilita el trabajo de reconocimiento.

De interpretación: Es la categoría responsable de identificar el significado o semántica asignado por el usuario a la entrada, su salida puede ser representada con EMMA (Extensible MultiModal Annotation markup language), como lenguaje para representar la semántica o significado de los datos, a través de anotaciones. Para modelar este componente se sugiere:

- Para cada dato de entrada identificar sus posibles formas de representación, según la modalidad que utilice. Por ejemplo, una afirmación se puede representar por medio de la voz con: si, ok, de acuerdo, bien, etc; por medio de un manuscrito se representa con un visto bueno; por medio de la expresión corporal con un movimiento vertical de la cabeza, o con la mano, señalando el índice hacia arriba.
- Crear una tabla que relacione cada una de las formas de representación identificada con su respectivo significado, teniendo en cuenta la modalidad de entrada, para todos y cada uno de los datos de entrada
- A partir de la tabla generar las correspondientes anotaciones haciendo uso de EMMA.

De integración: Es la categoría encargada de combinar la entrada desde los diferentes componentes de interpretación. Es conveniente tener en cuenta que algunas o todas las funcionalidades de éste componente se pueden implementar en el componente de reconocimiento, en el de interpretación o en el de interacción. Para facilitar el modelamiento de éste componente se recomienda:

- Identificar los datos que permiten capturas a partir de múltiples modalidades.
- Establecer una estrategia de integración para cada uno de los datos identificados en el punto anterior.
- Generar las correspondientes anotaciones que resulten de la implementación de la estrategia de integración, haciendo uso de EMMA.

3.1.7 Componente de salida:

En el momento de modelar la salida de la aplicación se deben determinar los siguientes aspectos:

- Para cada caso de uso de la aplicación establecer las posibles salidas.
- Para cada salida identificar las modalidades en que se representará
- En caso de hacer uso de múltiples modalidades al mismo tiempo, establecer una estrategia de sincronización
- Establecer las características de estilo para cada salida según su modalidad.
- Establecer las características de presentación para cada salida según su modalidad.

Cada uno de estos aspectos debe ser tenido en cuenta para la construcción de los elementos que conforman el componente de salida, como se describe a continuación.

Componente de Generación: Determina cual o cuales son los modos (complementarios o suplementarios) a usar, para representar la información generada por el administrador de interacción para el usuario. Debido a que la plataforma propuesta por la W3C lo representa como un componente no obligatorio, y además teniendo en cuenta que el lenguaje de representación interna usado para describir la salida de este componente, actualmente es tema de discusión para el grupo W3C, la presente guía no lo contempla, delegando esta responsabilidad al administrador de interacción.

Componente de Estilos: Es el encargado de agregar información a cerca de cómo se presentará la salida, definiendo aspectos como pausas en la voz, inflexiones entre otros. Para su construcción se sugiere hacer uso de etiquetas del Lenguaje de Marcado SSML (Speech Synthesis Markup Language), en el caso de que la salida sea por medio de la voz. Para los componentes de estilo gráfico se recomienda utilizar con XHTML o SVG (Scalable Vector Graphics)

Componente de Presentación: Su función consiste en convertir la información del componente de estilo en un formato fácil de entender por el usuario, como lo es la conversión de texto a voz, representaciones gráficas, etc. Es de vital importancia tener en cuenta que a cada modo de salida le corresponde un componente de presentación. En caso tal de que sea necesaria la coordinación de la salida a través de diferentes medios, se recomienda utilizar para su sincronía SMIL (Synchronized Multimedia Integration Language).

3.2 MODELADO SEMÁNTICO

El modelado semántico, al igual que cualquier forma de modelado del conocimiento es un proceso complejo cuya complejidad esta relacionada, pudiéramos decir de manera inversa, con las habilidades de abstracción y las facilidades lingüísticas del especialista. Desarrollar una ontología generalmente implica especialistas que están ubicados geográficamente en lugares distintos, endureciendo un proceso que es esencialmente complejo.
La creación de ontologías ha ido saliendo de los laboratorios de Inteligencia Artificial hacia la Web, motivado por los esfuerzo del W3C de que los recursos de la red sirvan para que agentes inteligentes ayuden a las personas a tomar decisiones. Este proceso puede tener varias motivaciones, entre las que se pueden destacar [71]:

1. Compartir con la comunidad informática y con los usuarios finales la estructura del conocimiento de un dominio determinado.
2. Posibilitar la reutilización del conocimiento de un dominio.
3. Separar el conocimiento del dominio del conocimiento operacional.

La necesidad de compartir el conocimiento es muy antigua. Las ontologías proporcionan un estándar legible para cualquier persona que novata en el tema del dominio o tan complejo como se haya desarrollado para expertos.

Estos expertos o estudiosos novatos pueden tomar el conocimiento ya desarrollado por otros y enriquecerlo mediante su reutilización. El enriquecimiento de un conocimiento dado debe ser consecuente con este, o sea, no ser contradictorio ni eliminar parte de él.

Ahora bien, con los nuevos modelos de la ingeniería de software la separación de los elementos de una aplicación juega un papel preponderante. Por tanto si el conocimiento experto se aísla y se abstrae de las operaciones, los resultados pueden ser mas genéricos y por tanto mas inteligentes.

Hay que anotar que desarrollar la ontología siempre tiene que estar orientada hacia la posibilidad de que este conocimiento sea usado en primer lugar por máquinas.

Los elementos de una ontología son 3: Clases, Propiedades y Restricciones. Nos gustaría pensar que el modelado semántico es equivalente al diseño de un diagrama entidad-relación, y aunque puede servir en la etapa inicial del proceso, características de la representación del conocimiento lo van alejando de este a medida que avanza el proceso. También sería muy simple si el modelado semántico sería equivalente a modelar clases para la programación orientada a objetos, pero la programación orientada a objetos requiere modelar como funcionan las cosas y a partir de esto enmarca una estructura.

Los especialistas no han llegado a un consenso formal a cerca de una metodología de diseño, sin embargo se ha definido claramente cuál es la ruta a seguir.

En la práctica desarrollar una ontología implica en primer lugar [71]:

- Determinar las clases
- Organizar las clases en una jerarquía (Clase-Subclase). A esto se le denomina taxonomía
- Definir las propiedades que van a relacionar individuos de las diferentes clases y propiedades que van a ayudar a describir individuos o a describir la ontología (metadatos)
- Asignar los valores a cada una de las instancias.

Otros elementos importantes son aquellos que dependiendo de la complejidad del diseño se deseen implementar (OWL-Lite, OWL-DL, OWL-Full)

- Definición de las características de las propiedades (inversa, cardinalidad, transitividad, etc.)
- Definición de características de las clases e individuos en sus relaciones (unicidad, equivalencia, separaciones, etc.)

Con esto queda claro que solo existe una línea delgada entre el modelado semántico y la base de conocimientos. Pudiéramos expresar la fórmula:

$$BasedeConocimiento = Taxonomía + Individuos$$

3.2.1 Premisas

Los investigadores y las actividades desarrolladas durante este trabajo, avalan la existencia de un grupo de premisas que siempre se deben tomar en cuenta. Estas premisas encaminan el trabajo hacia un modelo semántico del dominio del conocimiento eficaz. Dichas premisas aunque elementales deben ser observadas constantemente:

1. No existe un único modelo correcto del dominio, siempre existirán varias alternativas

   El modelo ”más correcto” es aquel que satisface los requerimientos de la aplicación de la ontología. Si el modelo va a representar conocimiento es evidente que todos podemos pensar diferente y cada cual tiene su modo particular de aprender. Crear una ontología es como explicarle a un niño como nacen los bebés, cada papá se busca el símil que considere mas sencillo y menos comprometedor. Este símil esta sin duda ligado a las habilidades lingüísticas del padre.

2. La creación de una ontología es un proceso iterativo.

   El desarrollo de una ontología sigue el modelo de aprendizaje *ensayo y error*. El concomimiento es dinámico (a diferencia del razonamiento que siempre sigue el mismo modelo para cada individuo normal a no ser que tenga un pensamiento divergente)

3. Los conceptos en la ontología deben limitarse a objetos (físicos o lógicos) y sus relaciones en el dominio de interés.

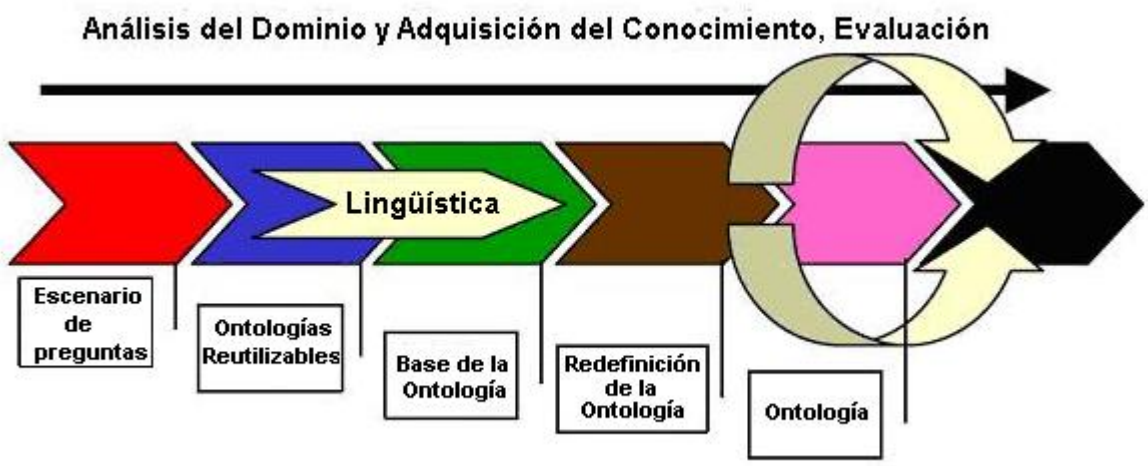
   Las soluciones deben ser simples: sustantivos equivalen a clases, verbos equivalen a propiedades.

3.2.2 Pasos

Para la creación de una ontología se deben seguir los siguientes pasos [71]:

Paso 1. Determinar el dominio y el alcance de la ontología.
Paso 2. Considerar la reutilización de ontologías existentes
Paso 3. Enumerar términos importantes en la ontología
Paso 4. Definir las clases y su jerarquía
Paso 5. Definir las propiedades de las clases
Paso 6. Definir las restricciones de las propiedades
Paso 7. Crear las instancias

Figura Nº 29. Guía de diseño de una ontología



Fuente: The use of conceptual maps for two ontology developments: nutrigenomics, and a management system for genealogies. Alexander Garcia y otros.

Expliquemos en detalle cada paso.

Paso 1. Determinar el dominio y el alcance de la ontología.

Tener claridad de que es lo que vamos a representar, cuál es el dominio del conocimiento que se desea expresar y hasta que área va a cubrir es el primer elemento a considerar. Sería el punto equivalente a definir los objetivos de una investigación conociendo y habiendo formulado el problema. Este proceso puede ser conducido por preguntas concretas sobre lo que se desea representar o lo que se debe alcanzar con la aplicación en mente.

Estas preguntas pueden ser generales y no estar ligadas al problema en si, tales como:

- ¿Cuál es el dominio que la ontología cubrirá?
- ¿Para qué usaremos la ontología?
- ¿Para qué tipos de preguntas la información en la ontología deberá proveer respuestas?
- ¿Quién usará y mantendrá la ontología?

Una vez determinado el dominio, se debe pensar cuál es la finalidad de diseñar esta ontología, o deforma mas clara, que preguntas deben poder responderse basado en ella. Basado en nuestro objeto de estudio pudiéramos tener preguntas como:

- ¿Cuan relevante es la composición de genero en el fenómeno del desplazamiento?
- ¿Un desplazado por el conflicto armado puede estar amenazado nuevamente?
- ¿Qué áreas del territorio nacional son mas proclives a generar desplazamiento?
- ¿Cuan grande en términos de población es el problema de desplazamiento hacia Cartagena de Indias?

Este paso nos debe permitir sacar una lista inicial de clases (muy similar al proceso de creación de un diagrama entidad relación)

Paso 2. Considerar la reutilización de ontologías existentes

El re-uso es muy importante en el modelado de ontologías y un aspecto muy bien logrado en el OWL. Re-usar no es simplemente un ahorro de tiempo e investigación sino también un aporte al trabajo ya realizado.
Varias bibliotecas de ontologías (OWL y DAML) se pueden encontrar en la Web algunas tan grandes y bien definidas que son muestra de una larga investigación.

Paso 3. Enumerar términos importantes en la ontología

Determinar los términos nos debe conducir a propiedades e incluso a la aparición de nuevas clases. Los términos deben estar relacionados sobre el dominio. Un buen recurso sería proponer una conversación sobre el dominio entre conocedores con sus planteamientos concretos y novatos elucubrando sobre las posibles respuestas.

En algunos casos los términos son triviales, por ejemplo: las personas tienen nombre, apellidos, edad. Sería muy fácil pensar que las personas tienen un número de identidad que los identifica ante el estado, pero un conocedor del tema del desplazamiento puede hacer desplomar esta opinión.

Paso 4. Definir las clases y su jerarquía

Este es el punto que marca el verdadero arranque en la investigación (parecido a una carrera de la F1 donde en el momento de la salida ya se ha hecho la *Pole*).

El desarrollo de la jerarquía puede ser *top-down*, *botton-up* o combinado. Es importante destacar que no existe una taxonomía correcta para un dominio dado. La taxonomía esta supeditada a los posibles usos de la ontología, el nivel de detalle que es necesario para la aplicación, preferencias personales y algunas veces requerimientos de compatibilidad con otros modelos

Para desarrollar este paso es recomendable lo siguiente:

- Asegurarse que la taxonomía es correcta
- Analizar las clases hermanas entre sí
- Herencia múltiple
- ¿Cuándo introducir (o no) una nueva clase?
- ¿Una nueva clase o un valor de propiedad?
- ¿Una instancia o una clase?
- Limitación del alcance
- Subclases disjuntas

Asegurarse que la taxonomía es correcta

Una jerarquía de clases es una relación *is-a* (es un): una clase A es una subclase de B si cada instancia de B es también una instancia de A. Esta relación no debe indicar que un objeto simple es una subclase, por ejemplo una Persona no es una subclase de Personas. Es recomendable utilizar siempre la misma forma singular o plural para nombrar las clases.

Una jerarquía establece una relación transitiva por tanto: Si B es una subclase de A y C es una subclase de B, entonces C es una subclase de A. Los elementos de un dominio de conocimiento pueden cambiar con el tiempo por lo que la definición de la jerarquía puede ser un gran reto. Adicionalmente si la ontología es re-utilizada algún investigador puede tener información contradictoria .

Otros elementos relevantes son la verificación de que la ontología no tenga ciclos en las clases ni que tengas clases que representen el mismo objeto pero con nombre diferentes.

Analizar las clases hermanas entre sí

Las clases hermanas en una jerarquía son clases que son subclases directas de la misma clase. Las clases hermanas deben ser generalizadas por los mismos atributos de su antecesor.

La relación superclase-subclase debe ser tan legible como se pueda, pero tener pocas clases o muchas puede significar errores de diseño. Si una clase tiene solamente una subclase directa, puede existir un problema de modelamiento o sino la ontología no está completa. Si hay más de una docena de subclases para una clase dada, entonces categorías intermedias adicionales pueden ser necesarias [71]

Herencia múltiple

Muy criticada por muchos diseñadotes de software y hasta considerada un error por algunos, el pensamiento humano es por naturaleza basado en este esquema. Es necesario revisar que una clase que herede de varias tiene los elementos requeridos sin que le sobren características que en realidad no posee.

¿Cuándo introducir (o no) una nueva clase?

Las subclase tienen por definición características que la superclase no tiene y que la diferencia de alguna manera de sus hermanas. Una subclases no tiene que ser necesariamente disjunta de sus hermanas pero debe tener diferencias.

¿Una nueva clase o un valor de propiedad?

Tanto en el modelado Entidad-Relación como en el modelado semántica, discernir entre si un elemento es clase o propiedad es de los elementos que requieren mayor habilidad del diseñador.

Si los conceptos con diferentes valores de propiedades se vuelven restricciones para diferentes propiedades en otras clases, entonces debemos crear una nueva clase para esta distinción. Caso contrario, representamos la distinción en un valor de propiedad.

Si la distinción es importante en el dominio y pensamos en los objetos con diferentes valores para la distinción como diferentes tipos de objetos, entonces deberíamos crear una nueva clase para la distinción.

Considerar potenciales instancias individuales de una clase puede ser también útil al decidir si se introduce una nueva clase o no. *Una clase a la cual una instancia individual pertenece no debería cambiar a menudo.*

Las propiedades que almacenan números, colores, fechas casi nunca son clases, aunque si una de esta es muy significativa para el dominio entonces debe tomarse

en cuenta como clase. POr ejemplo, es valioso conocer la cantidad de hijos de una persona pero si se debe saber el sexo del hijo primogénito entonces hay que considerar crear una clase.

¿Una instancia o una clase?

Las instancias son los elementos de una base de conocimiento. Estos son el núcleo del razonamiento y la fuente de resultados.

Un individuo puede tener características especiales al tener una relación con otro individuo en particular. Por ejemplo, un individuo desplazado del Carmen de Bolívar hacia Turbaco es parte del problema social en estudio mas no el objetivo fundamental. Son fundamentales los individuos que su desplazamiento los dirige a Cartagena, en ese caso se necesita conocer donde están asentados. Estos elementos son suficientes para definir a Cartagena como una clase.

Limitación del alcance

Una ontología tan general que abarque todo el ámbito del conocimiento, a parte de inoperante, nunca es necesaria. El problema a resolver tiene un grupo de elementos fáciles de distinguir. Por “encima” y pro debajo de este grupo debe haber como máximo un nivel.

Subclases disjuntas

Muchos sistemas nos permiten especificar explícitamente que varias clases sean disjuntas. La clases son disjuntas si no pueden tener ninguna instancia en común.

Paso 5. Definir las propiedades de las clases

La taxonomía no significa mucho si no se incluyen en este modelo las propiedades. Las propiedades pueden clasificarse en:

- propiedades “intrínsecas”. Describen directamente a la clase. Están asociadas con lo que la clase representa (sexo, edad, identificación)
- propiedades “extrínsecas”. Describen a la clase pero están ligadas a lo que se desea conocer (número de hijos, estado de salud)
- partes. Describen objetos estructurados
- relaciones con otros individuos. Estas son las relaciones entre miembros individuales de una clase e individuo de otra clase

Al definir propiedades es relevante indicar propiedades inversas y valores por defectos. Esto mejora el trabajo del razonador y del manejador de la ontología.

Paso 6. Definir las restricciones de las propiedades y las restricciones de las clases

Las propiedades puedes tener diferentes restricciones que describen el tipo de valor, valores admitidos, el número de los valores (cardinalidad), y otras características de los valores que las propiedades pueden tomar. Para definir estas restricciones se debe tomar en cuenta:

- Funcionalidad. Esta puede determinar si la propiedad admite solo uno o varios valores.
- Tipo de datos. Los tipos de datos son muy sencillos para cualquier conocedor de las herramientas de computación. Una persona familiarizada con el Excel conoce que tipos de datos se pueden representar. Técnicamente estos tipo de datos están preestablecidos en el RDF.
- Dominio y rango. Estos conceptos, explicados en la capa ontológica, son los que van a establecer las relaciones entre las clases a través de las propiedades. Esto es la esencia del RDF.

Las restricciones de las clases son las encargadas de determinar si un individuo es o no miembro de la clase durante el proceso de razonamiento.

Paso 7. Crear las instancias

Crear instancias es “poblar” la ontología. Este proceso debe tener en cuenta las restricciones ya que podemos estar generando una ontología inconsistente.

4. MODELADO DEL PROTOTIPO DE PORTAL DE INFORMACIÓN SOBRE EL PROBLEMA DE DESPLAZAMIENTO FORZADO EN LA CIUDAD DE CARTAGENA

Para la elaboración del prototipo se aplica la metodología de construcción de software denominada: Proceso Unificado de Desarrollo de software, por todas las ventajas que representa el desarrollo iterativo incremental, centrado en la arquitectura y dirigido por casos de uso. Los resultados se presentan teniendo en cuenta cada una de las actividades propias de dicha metodología, comenzando por la definición del *modelo del Negocio*, cuya finalidad consiste en entender el problema, luego se definen los *requerimientos* que establecen la funcionalidad de la aplicación, para posteriormente crear el *modelo de diseño*, a partir del cual finalmente se elabora la *implementación*. Estas dos últimas actividades muestran la forma en que se aplica la guía propuesta en la tesis.

4.1 MODELO DEL NEGOCIO

Colombia ha vivido un conflicto interno desde su nacimiento, reflejado de diferentes formas: El mismo proceso de conquista obligó a los indígenas a abandonar sus territorios, los esclavos tenían que huir para lograr su libertad, las guerras internas entre estados y departamentos a lo largo del siglo XIX; la guerra de los mil días, la violencia de los años 50, el nacimiento y la proliferación de las guerrillas a principios de los años 60, la creación de las Autodefensas Unidas de Colombia y los grupos paramilitares a finales del siglo XX. Todo esto ha generado como resultado lo que hoy se vive en el país: un desmedido, constante y al parecer irreversible desplazamiento de pobladores, casi siempre de zonas ruarles del país, hacia los centros urbanos en busca de seguridad, oportunidades y protección para sus vidas, aspectos que se tornan imposibles en su lugar de origen.

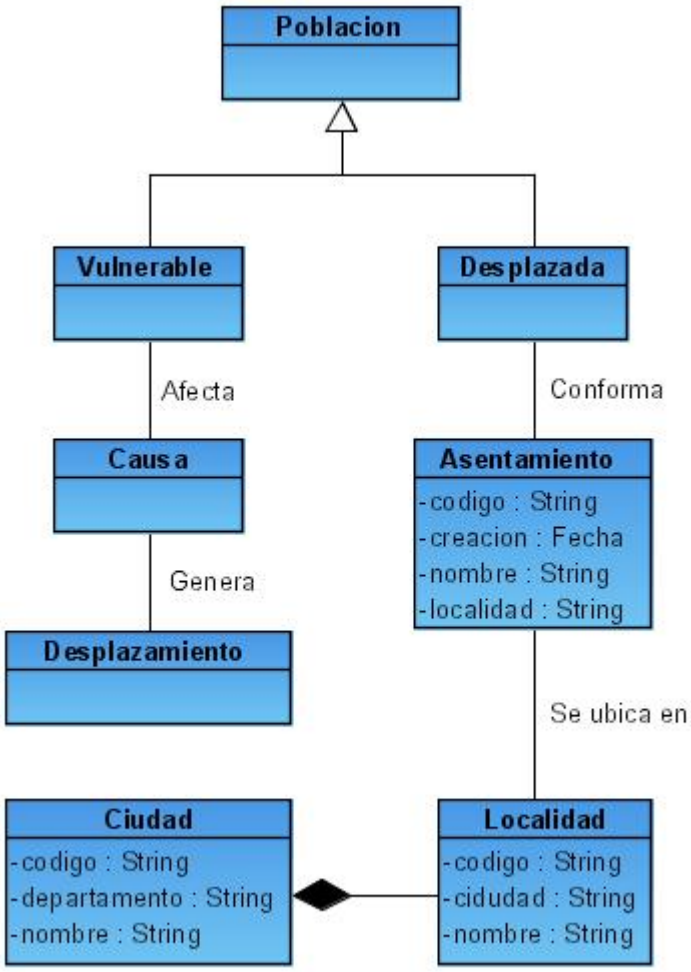
El desplazamiento interno forzado es la consecuencia más drástica que el conflicto interno trae consigo. El desarraigo que las personas expulsadas de sus tierras o que han tenido que huir por amenazas, muertes, masacres o muchas otras formas de violencia, la pérdida de sus bienes tangibles e intangibles, de la fe, la tranquilidad, la seguridad y la confianza, hacen que este fenómeno tenga proporciones mayores no sólo por el número de desplazados sino por las consecuencias que implica este tipo de problemáticas a nivel social, para los colectivos y para el desarrollo del país.

Este problema es objeto de estudio para el Centro de Investigaciones - CEIN, de la Universidad de San Buenaventura de Cartagena, el cual actualmente se

encuentra desarrollando un proyecto tendiente a construir un observatorio sobre el desplazamiento forzado específicamente en la ciudad de Cartagena, que permita hacer análisis sobre ésta problemática social que afecta a la población vulnerable, convirtiéndola en desplazada al huir de sus lugares de origen, hacia la ciudad en donde conforman asentamientos.

Para lograrlo es indispensable construir un sistema de información que permita registrar, procesar y analizar la información sobre la población desplazada, teniendo en cuenta que en la mayoría de los casos la recolección de información se hace directamente en los asentamientos, lo que obliga ha hacer uso de dispositivos móviles.

Figura Nº 30. Modelo del Dominio



Existen diferentes causas que originan el desplazamiento forzado interno, dentro de las cuales las más comunes son el conflicto armado y los fenómenos naturales, esto afecta a la población vulnerable, obligándola a abandonar su lugar de origen. Al llegar a la ciudad conforman asentamientos que se instalan en diferentes

localidades. Dicha situación se representa a través del siguiente diagrama de clases, que representa los objetos que intervienen y sus relaciones.

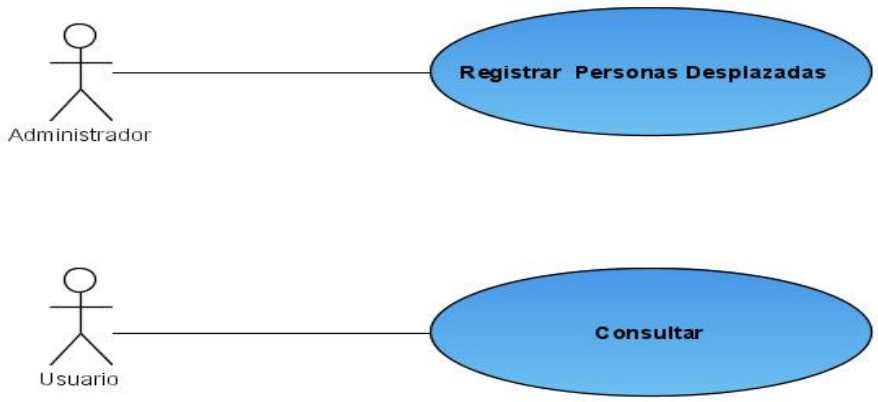
4.2 IDENTIFICACIÓN DE REQUERIMIENTOS

El CIEN necesita registrar la información pertinente sobre la población desplazada, teniendo en cuenta que dicho proceso se hace directamente en los asentamientos lo que obliga ha hacer uso de dispositivos móviles. Además se requiere una plataforma que permita realizar consultas sobre la información previamente registrada.

Teniendo en cuenta que el objetivo principal es presentar un prototipo que sirva de ejemplo que ilustre la guía propuesta, la funcionalidad de la aplicación se centrará en el proceso de captura de información del desplazado, y en consultas sobre la población desplazada.

4.3 MODELO DE DISEÑO

Figura Nº 31. Diagrama de Casos de Uso

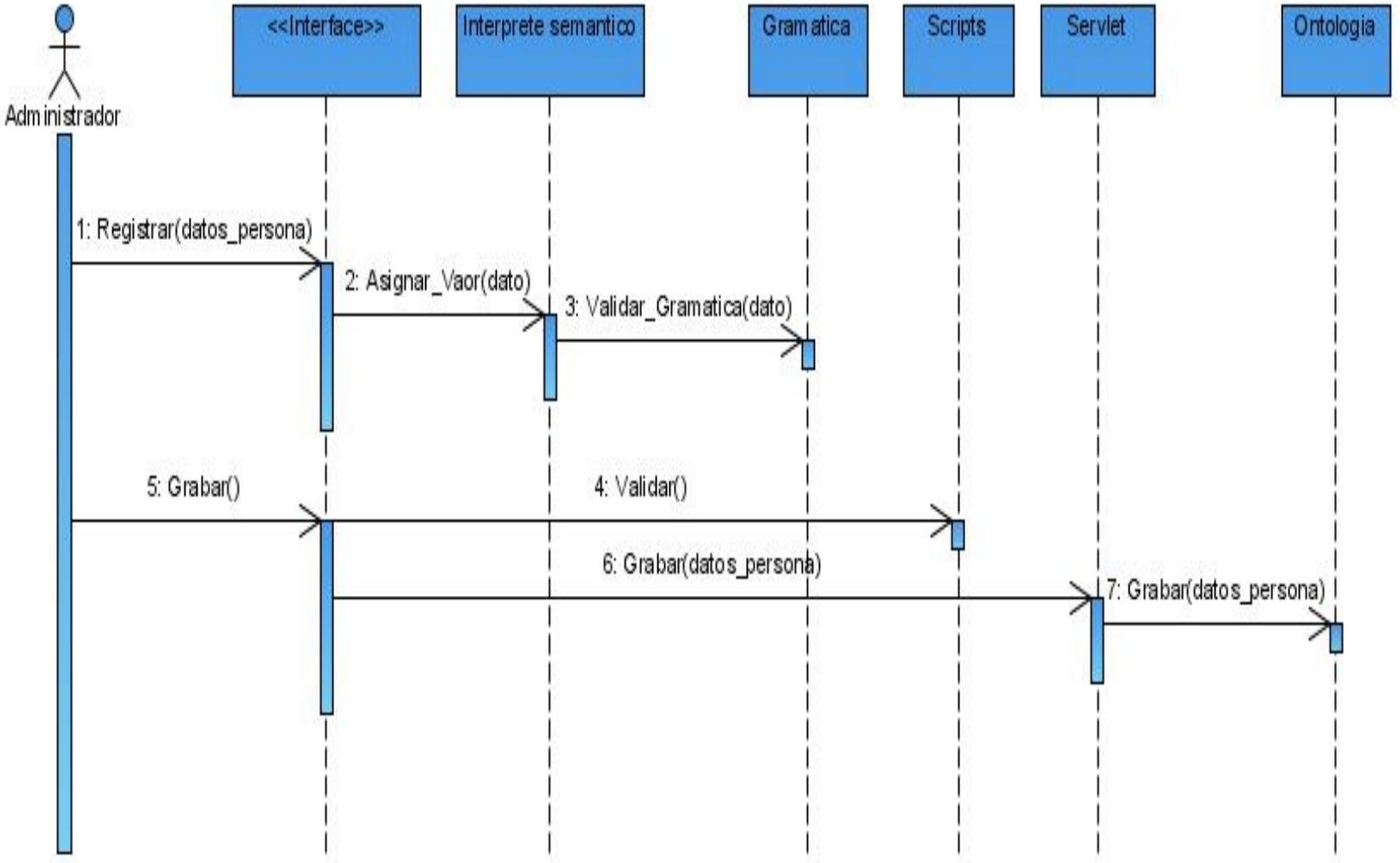


Los requerimientos planteados definen dos funcionalidades de la aplicación, realizada cada una de ellas por diferentes actores. La primera hace referencia al proceso de registro de la información, el cual debe ser realizado sólo por aquellas personas debidamente autorizadas, y que para este caso de estudio denominaremos administrador. La segunda se refiere al proceso de consulta, que puede ser realizado por cualquier tipo de usuario, debido a que no representa ningún tipo de riesgo para la integridad de la información del sistema. Por lo tanto se han identificado dos casos de uso, como se representa a continuación, en el gráfico y en la respectiva descripción de cada uno de ellos, junto con sus correspondientes diagramas de secuencia.

Descripción de los casos de Uso

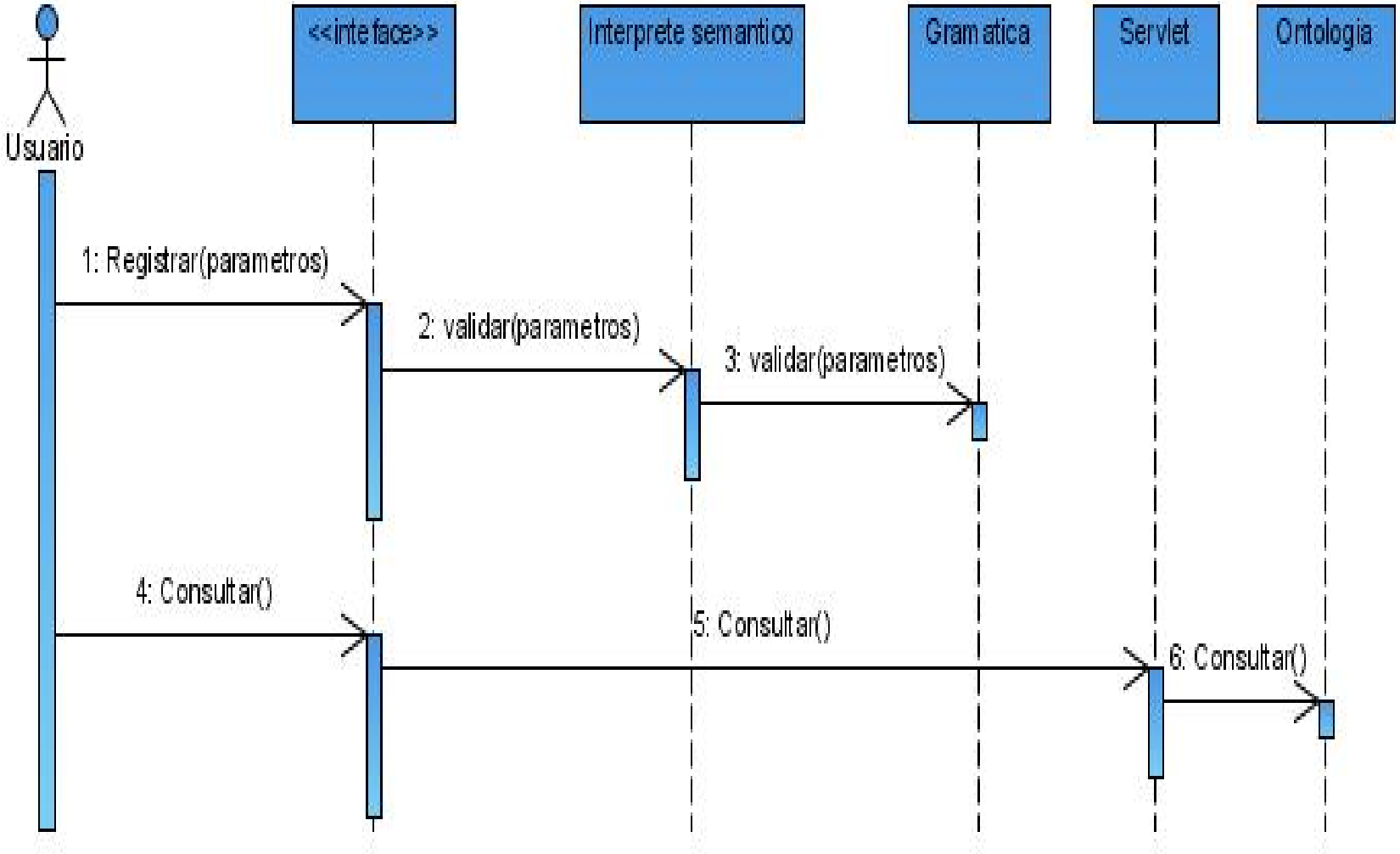
Nombre: Registrar personas desplazadas
Actor: Administrador
Personal involucrado e intereses: • Director del centro de investigación: Desea que la información registrada correcta y completa. • Administrador: Desea que el proceso de registro sea sencillo y ágil • Desplazado: Suministra la información relacionada con su estado en calidad de desplazado.
Precondición: El administrador se identifica y autentica
Poscondición: Los datos de la persona quedan registrados correctamente en el sistema.
Flujo Principal de Éxito:
1. El actor registra los datos de la persona. 2. El sistema valida los datos registrados. 3. El actor graba. 4. El sistema registra los cambios en la ontología
Flujo Alternativo:
2.a. El sistema identifica que existen campos obligatorios vacíos. 1. El cursor se ubica en el primer campo obligatorio que se encuentre vacío 2. El actor continua el proceso de registro 2.b. El actor dice alguna palabra no incluida en la gramática 1. El sistema indica que se ha cometido un error 2. El actor continua el proceso de registro 4.a. No es posible registrar los cambios en la ontología 1. El sistema genera un mensaje de error 2. El actor debe repetir el proceso
Requisitos especiales:
Interface de usuario que permita interactuar a través de la voz, el teclado y un dispositivo apuntador.
Lista de tecnología:
Para la captura es posible que se utilicen dispositivos móviles
Frecuencia: Aleatoria

Figura Nº 32. Diagrama se Secuencia: Registrar personas desplazadas



Nombre: Consultar
Actor: Usuario
Personal involucrado e intereses: • Director del centro de investigación: Desea que la información arrojada por el sistema sea correcta y completa. • Usuario: Desea que el proceso de consulta sea sencillo y correcto
Precondición: Ninguna
Poscondición: El sistema responde correctamente a la consulta del usuario.
Flujo Principal de Éxito:
1. El actor registra los parámetros de la consulta. 2. El sistema valida los parámetros registrados. 3. El actor solicita la respuesta. 4. El sistema genera la respuesta a partir de la ontología
Flujo Alternativo:
2.a. El actor dice alguna palabra no incluida en la gramática 1. El sistema indica que se ha cometido un error 2. El actor continua el proceso de registro 4.a. No es posible generar la respuesta. 3. El sistema genera un mensaje de error 4. El actor debe repetir el proceso
Requisitos especiales:
Interface de usuario que permita interactuar a través de la voz, el teclado y un dispositivo apuntador.
Lista de tecnología:
Para la captura es posible que se utilicen dispositivos móviles
Frecuencia: Aleatoria

Figura Nº 33. Diagrama se Secuencia: Consulta



Teniendo en cuenta los escenarios del prototipo explicados anteriormente, a través de sus respectivos casos de uso y diagramas de secuencia, a continuación se presenta a nivel de ejemplo, la forma de aplicar la guía propuesta.

4.4 MODELO DE INTERACCIÓN MULTIMODAL

4.4.1 Aspectos generales:

Definir el orden de interacción: El orden de interacción se refleja en los diagramas de secuencia representados en las dos figuras anteriores, para su implementación se realizaron las siguientes acciones:

- Se identificó la información que la aplicación requiere del usuario para registrar a las personas desplazadas, a saber: Nombre, apellido, identificación, día de nacimiento, lugar de nacimiento, sexo, raza, fecha del registro, lugar de origen, nivel de educación, número de hijos y asentamiento en el que se encuentra.
- Se determinó que el usuario puede grabar la información cuando los datos están debidamente registrados, y que en cualquier momento puede interrumpir la acción cerrando el navegador o dirigiéndose a otra dirección. De igual manera puede solicitar ayuda pronunciando la palabra help, para que la aplicación presente un mensaje de ayuda, acorde al contexto, por medio de voz digitalizada. Esto se puede apreciar en los archivos lugar_colombia.mxml, date.mxml, y digito.mxml, cuando se invoca el elemento <vxml:help>
- Para definir el modelo del flujo de la interacción se construyeron los diagramas de secuencia de las figuras 31 y 32, a partir de sus las respectivas descripciones de los casos de uso.
- Se decidió que la reacción de la aplicación ante las respuestas correctas del usuario, consistirá en pasar al siguiente campo, esto se logra con el manejo de eventos xml, como se puede apreciar el archivo DatosPersonales.mxml, en las líneas que invocan <ev:listener>. Para las incorrectas se enviará un mensaje de error, acompañado de una posible respuesta correcta, utilizando el elemento <vxml:nomatch>. En ausencia de respuestas, a los diez segundos se repite la solicitud por medio de un mensaje de voz, a los 20 segundos se indica que no se ha escuchado la respuesta, posteriormente se queda en una espera infinita hasta que el usuario registre la información o cancele el proceso. Esto se logra haciendo uso del atributo timeout del elemnto <prompt>, en combinación con el elemento <noinput>. Si se presenta algún error, será capturado a través del elemento <vxml:error>.
- A partir de la información requerida, se identificó que para el nombre y los apellidos no se puede establecer una gramática, debido a su gran

diversidad; para la identificación se hará uso de una secuencia de dígitos; para el día de nacimiento y la fecha del registro se utilizará una gramática de fecha, para el lugar de nacimiento y el lugar de origen se utilizará una gramática que contenga lugares de Colombia, el número de hijos será un valor comprendido entre 0 y 20; el sexo, la raza, el nivel de educación, y el asentamiento tendrán gramáticas que serán descritas detalladamente más adelante, cuando se modele el componente de reconocimiento.

- Como resultado del registro de los desplazados, la aplicación emitirá un mensaje indicando si el objetivo se logró o no. En el caso de la consulta se generará una tabla.

*Definir de recursos tecnológicos para la realización de la aplicación***:**

- Debido al alto grado de exigencia de estos entornos de desarrollo, a nivel de software y de hardware, se decidió no utilizar ninguno de ellos.
- Lenguajes de programación: Se seleccionó XHTML + Voice, en combinación con java, para la construcción de las gramáticas: Java Speech Grammar Format JSGF, Las ontologías se definen con OWL, utilizando el editor Protege, el cual cuenta con un API robusto que facilita la programación del acceso a la ontología..
- Como Servidor de aplicaciones se escogió Apache Tomcat, para ser coherentes con el lenguaje de programación seleccionado.
- El Navegador multimodal que se seleccionó fue Opera por que incorpora la tecnología de voz desarrollada por IBM, la cual, al igual que el navegador puede ser utilizada sin incurrir en ningún costo. Su desventaja radica en el hecho de que la interacción se hace en ingés,

4.4.2 Funciones de la aplicación

Esta parte se detalla más adelante, cuando se aplique la guía para el modelado semántico.

4.4.3 Componente de sesión

- Para el caso del registro de las personas desplazadas se determinó que el nombre y el apellido se ingresarán exclusivamente a través del teclado. Los demás datos podrán ser registrados además del teclado por medio de la voz, teniendo en cuenta que el caso del sexo, la raza y el asentamiento adicionalmente se puede utilizar un dispositivo apuntador, como el ratón.
- El registro de personas desplazadas no requiere que se permita guardar una sesión en un dispositivo, para posteriormente continuarla en otro similar

o de características distintas. Por esta razón no se implementa ninguna estrategia para conservar la sesión del usuario.

- En el contexto del problema del desplazamiento forzado, los dispositivos que podrán ser usados para que la aplicación se ejecute, serán asistentes personales digitales PDA, con capacidad para captura de voz.
- Aunque el PDA permite múltiples modos en forma secuencial y concurrente, para efectos del sistema de registro sólo se tendrá en cuenta la interacción de un solo modo en un momento determinado.
- Con la intención de simplificar el prototipo, se construyó un servlet que sólo se ejecuta cuando el usuario final grabe los datos de la persona. Esto evita la necesidad de mantener un control sobre el manejo de las sesiones de usuario. Sin embargo no es una buena práctica, por que afecta la facilidad de uso de la aplicación.

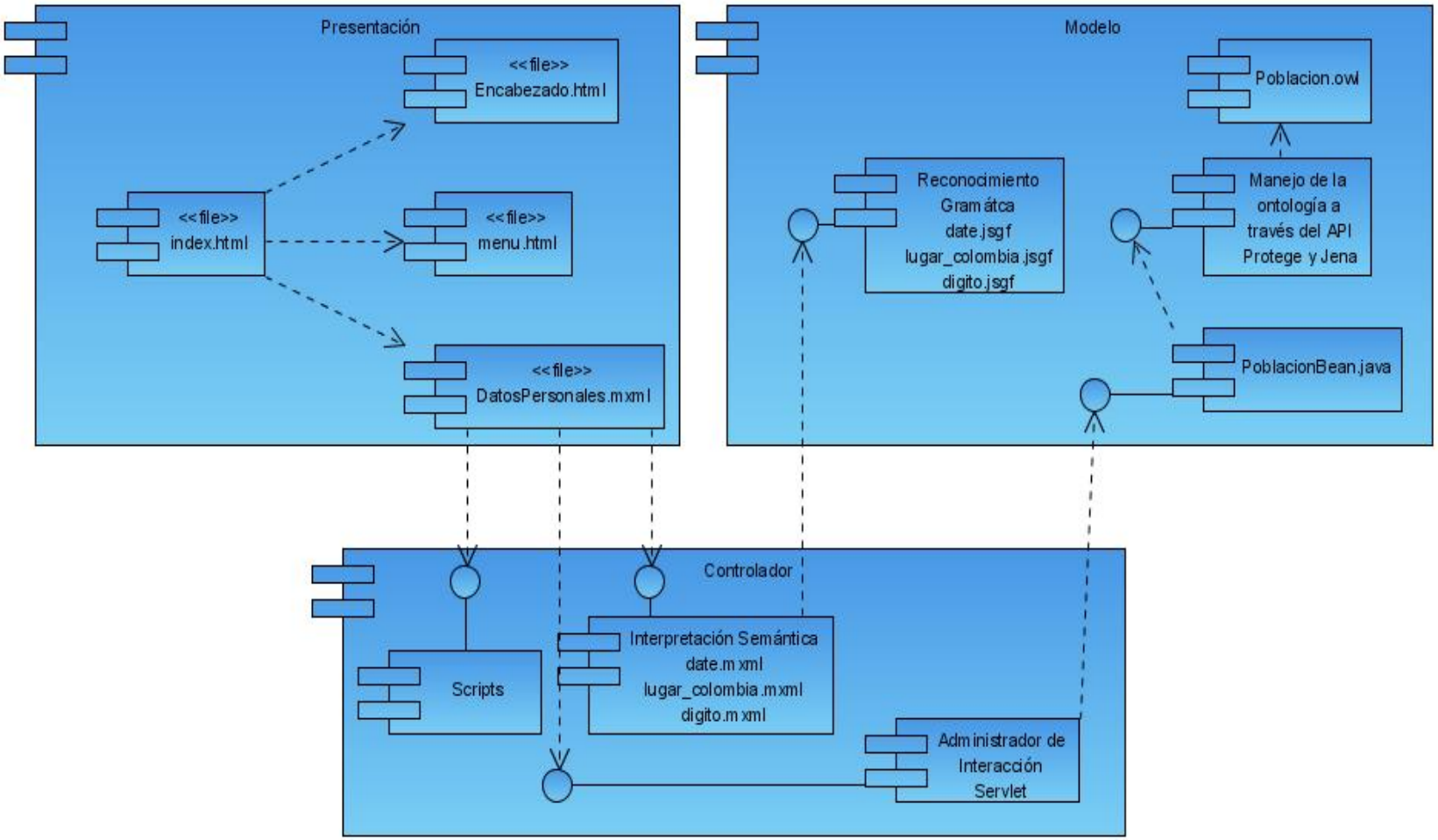
4.4.4 Administrador de interacción

Para comprender el modelado y construcción de este componente lógico que coordina datos y controla el flujo de ejecución, a partir de las diferentes interfaces de los componentes de las modalidades de entrada y salida, se recurre al diagrama de componentes de la figura Nº 33. En el se observa que la capa de presentación está conformada por archivos html, responsables de la página principal del portal y del menú. Además se encuentra el archivo DatosPersonales.mxml, escrito en XHTML + Voice, encargado de la interacción definida por las reacciones de la aplicación, que se han identificado previamente, como se detalla a continuación, describiendo la captura del número de identificación.

La parte gráfica de la interface se define por medio del código:

```
<tr>
    <td align="right">
      <b>Identification&#58; </b>
    </td>
    <td align="left">
       <input type="text" size="25" maxlength="10" id="identificacion"
              name="identificacion" ev:event="focus"
              ev:handler="#identificacion_vform" />  </td>  </tr>
```

Figura Nº 34. Diagrama de Componentes



Cuando el foco llega al objeto de entrada el control lo toma la siguiente sección de código, que invoca un subdiálogo, contenido en el archivo digito.mxml, el cual devuelve un valor que es asignado a la variable identificación.

```
<vxml:form id="identificacion_vform">
   <vxml:subdialog name="identificacion"
        src="subdialogos/digito/en_US/digito.mxml#formDigit">
   <vxml:param name="paramSubdialogObj" expr="objDigito" />
   <vxml:param name="paramPromptQuestion"
       expr="'What is the identification number?'" />
   <vxml:filled>
       <vxml:assign
           name="document.getElementById('identificacion').value"
           expr="identificacion.returnDigit" />
       </vxml:filled>
   </vxml:subdialog>
</vxml:form>
```

Ante las respuestas correctas del usuario, automáticamente se pasa al siguiente campo. Por ejemplo, para el caso en que el diga correctamente un número de identificación, se pasa a la captura de la fecha de nacimiento. Esto se logra con así:

```
…
<script id="performFocus1" type="text/javascript">
    if (docIsDoneLoading) {
       document.getElementById('fecha_nacimiento').focus();  }
</script>
…
<ev:listener ev:event="vxmldone" ev:handler="#performFocus1"
ev:observer="identificacion" ev:propagate="stop" />
…
```

Ante respuestas incorrectas, en el archivo digito.mxml se incluye el código:

```
<vxml:nomatch count="1">
    <vxml:value expr="paramSubdialogObj.noMatch1"/>
</vxml:nomatch>

<vxml:nomatch count="2">
   <vxml:value expr="paramSubdialogObj.noMatch2"/>
</vxml:nomatch>
```

En ausencia de respuestas se hace uso del elemento <noinput>, como se muestra a continuación:

```
<vxml:noinput count="1">
    <vxml:value expr="paramSubdialogObj.noInput1"/>
</vxml:noinput>

<vxml:noinput count="2">
   <vxml:value expr="paramSubdialogObj.noInput2"/>
</vxml:noinput>.
```

Por otra parte, en cualquier momento el usuario puede solicitar ayuda, para lo cual se hace uso del elemento <help>, así

```
<vxml:help count="1">
   <vxml:value expr="paramSubdialogObj.help1"/>
</vxml:help>

<vxml:help count="2">
   <vxml:value expr="paramSubdialogObj.help2"/>
</vxml:help>
```

En caso de generarse un error se controla a través del siguiente código, que emite un mensaje de voz manda el control del evento al plugin de voz de IBM.

```
<vxml:error>
    <vxml:value expr="paramSubdialogObj.error"/>
    <vxml:throw event="com.ibm.voicetools.exit"/>
</vxml:error>
```

Los valores de los parámetros de subdiálogo referenciados en cada una de éstas secciones de código se definen en el archivo digito.js así:

```
this.noMatch1 = 'Please say one or more digits. If you do not have the required information, say skip.';
this.noMatch2 = 'You must say one or more digits. If you do not have the required information, say skip.';
this.noInput1 = 'Please say one or more digits. If you do not have the required information, say skip.';
this.noInput2 = 'Can not hear you.  The connection might be bad.  Please hang up and call again.';
this.help1 = 'Please say one or more digits. If you do not have the required information, say skip.';
this.help2 = 'You must say one or more digits. If you do not have the required information, say skip.';;
this.error = 'An error has occurred in this application.  Please call back later.';
```

El tratamiento de los demás datos que permiten captura multimodal se hizo en forma similar. Adicionalmente, se incluyó un archivo de scripts, como se ve en el diagrama de componentes. En el se incluye la función Valida(), cuyo objetivo es identificar que los campos obligatorios han sido diligenciados, en caso de no ser así, el foco se asignará al primer campo obligatorio no diligenciado. También se incluye la función solo dígitos para asegurar que el número de identificación no contenga letras, al igual que el número de hijos. Para el caso del nombre y el apellido, se incluye la función tienedigitos(), para asegurar que dichos campos no contienen dígitos. Para las fechas se incluye la función ValidaFecha() encargada de verificar que la fecha registrada sea correcta.

En el momento en que el usuario presiona el botón submit, el control lo asume el servidor de aplicaciones Apache Tomcat, ejecutando el servlet Registro, donde se crea una instancia de la ontología y se graba el registro, según se describe en la sección del modelado semántico.

4.4.5 Componente de entrada:

A continuación se describe la forma como se aplicaron las recomendaciones de la guía al momento de construir cada una de las categorías del componente de entrada

De reconocimiento:

- Como se indicó anteriormente, se identificaron dos casos de uso: Registrar persona desplazada y consultar.
- Se hará uso de un PDA que permita captura por medio de la voz, adicional a la captura de su dispositivo apuntador, esto con el fin de facilitar el trabajo de la persona que realiza el registro de los desplazados.
- Las modalidades de entrada que se utilizan son la voz, el teclado, y un dispositivo apuntador como el ratón.
- El lenguaje de marcado para la interacción multimodal seleccionado es XHTML + Voice, por que se ajusta perfectamente a las modalidades establecidas para el prototipo.
- Para cada uno de los datos requeridos se definió la gramática como se muestra a continuación.

Tabla Nº 3. Datos requeridos

Dato de entrada	Gramática y restricciones
Nombre y apellido	No se aceptan dígitos
Identificación	Combinación de dígitos. No se aceptan letras

Fecha de Nacimiento y Fecha de Registro	año-mes-día. Donde el año tomará valores entre 1910 y 2009, los meses entre 1 y 12, o su nombre correspondiente, y los días entre 1 y 31. Se tienen en cuenta los años bisiestos, y la cantidad de días de cada mes.
Lugar de nacimiento y de origen	Lugar de Colombia. No se aceptan dígitos
Sexo	Únicos valores aceptados: Femenino y masculino
Raza	Únicos valores aceptados: Blanco, Negro y mestizo
Nivel de educación	Únicos valores aceptados: Ninguno, primaria, secundaria, universitario.
Número de hijos	Un número de 0 a 20
Asentamiento	Únicos valores aceptados: Ninguno, primero, segundo, tercero.

La gramática se construyó utilizando Java Speech Grammar Format, por que permite la combinación con scripts, lo que facilita el trabajo de reconocimiento. Se implementa un método del singleton responsable del acceso a la ontología, que genera en forma automática la gramática de los lugares válidos en Colombia, cada vez que se carga el sistema, cuyo resultado se conserva el archivo lugar_colombia.jsgf. Adicionalmente, se toma el archivo date.jsgf creado y distribuido por la IBM en calidad de ejemplo, como referente para la gramática en el momento de la captura de las fechas de nacimiento y de registro. En el caso de las gramáticas que aceptan un número pequeño de valores, su definición se realizó como se muestra a continuación:

```
<vxml:grammar>
    <![CDATA[
        #JSGF V1.0;
        grammar tipo_raza;
        public <raza> = white | black | mixed ;
    ]]>
</vxml:grammar>
```

De interpretación:

- Todos los datos capturados por el prototipo cuentan con representaciones únicas, cada una de las cuales corresponde a su vez a un solo significado. Excepto en el caso de la fecha, para el cual se presenta una situación particular, que consiste en que el año puede ser enunciado indicando solo

sus dos últimos dígitos, por ejemplo al enunciar el año 66 se está haciendo referencia al año 1966, mientras que si se indica un valor menor que diez, se estaría haciendo referencia a la presente década. Por otra parte, el mes puede indicarse con su nombre o con un número entre 1 y 12.

- La siguiente tabla relaciona cada una de las formas de representación identificada para la fecha con su respectivo significado, teniendo en cuenta la modalidad de entrada

Tabla Nº 4. Formas de representación de la fecha

Forma de representación según Modalidad		Significado
Voz	Texto	
Para el año		
Pronuncia el año indicando solo sus dos últimos dígitos con un valor entre 0 y 9	Se escribe un valor entre 0 y nueve con uno o dos dígitos así: 0, 1 o 01, 2 o 02, ...	Se asume que se hace referencia al año 2000 incluyendo el valor capturado. Así: 2001, ...
Pronuncia el año indicando solo sus dos últimos dígitos con un valor entre 10 y 99	Se escribe un valor entre 10 y 99 con dos dígitos así: 10, 11, 1 2, ...	Se asume que se hace referencia al año 1900 incluyendo el valor capturado. Así: 1910, ...
Para el mes		
Pronuncia un número entre 1 y 12	Se escribe un valor entre 1 y doce con uno o dos dígitos así: 1 o 01, 2 o 02, ...	Se asume que se hace referencia al mes que corresponde al número digitado. Su representación interna es numérica
Se pronuncia el nombre del mes		Se asume que se hace referencia al mes que corresponde al número digitado. Su representación interna es numérica
Para el día		
Pronuncia un número entre 1 y 31 en su forma ordinal, ejemplo: primero, segundo, ...		Se asume que se hace referencia al día correspondiente del mes.

- Aunque el presente proyecto no contempla la generación de las correspondientes anotaciones haciendo uso de EMMA, si se tuvieron en cuenta las situaciones descritas en la tabla Nº 4, en el momento de definir

la gramática, como se puede observar en la siguiente sección de código, que interpreta el año según lo descrito en la tabla.

```
<yr> =  nineteen ( <NULL> | hundred [and] ) <numbers10to99>
          {  $ = "19" + $numbers10to99;  }
        | <numbers10to99digit>
          {  $ = "19" + $numbers10to99digit; }
        | nineteen [and] <2dNum1to89>
          {  $ = "19" + $2dNum1to89; }
        | two thousand (zero|oh|nil|null) <numbers1to9>
          {  $ = "20" + "0" + $numbers1to9; }
        | two thousand
          {  $ = "2000"; }
        | two (zero|oh|nil|null)  <numbers1to9>
          {  $ = "20" + "0" + $numbers1to9; }
        | <numbers10to99>
          {  $ = "19" + $numbers10to99; }
```

De integración: Esta responsabilidad fue asignada al componente de interacción, aplicando las recomendaciones de la guía:

- Como se indicó previamente, solo el nombre y el apellido será capturado únicamente por medio del teclado, los demás datos podrán ser registrados además del teclado por medio de la voz, teniendo en cuenta que el caso del sexo, la raza y el asentamiento adicionalmente se puede utilizar un dispositivo apuntador, como el ratón.
- La integración se realiza en el momento de la validación del formulario, y a través del servlet que es el encargado de finalmente grabar los datos registrados en la ontología.
- La generación de las correspondientes anotaciones haciendo uso de EMMA, se convierte en material de estudio para futuros trabajos.

4.4.6 Componente de salida:

Los resultados obtenidos al aplicar las recomendaciones de la guía al momento de construir éste componente fueron:

- En el caso de uso registrar desplazado, las salidas están representadas por los mensajes de solicitud de ingreso de respectivo dato, y por un mensaje final que indica si el resultado del registro fue exitoso o no. Como se mencionó anteriormente, dichas salidas son implementadas por medio del

elemento <prompt> que proporciona XTHML + Voice. Para el caso de uso consulta, su resultado se representa únicamente a través de una tabla.

- Las modalidades que se utilizarán para la salida, según lo descrito anteriormente, serán la voz y la parte gráfica.
- Debido a que las salidas implementadas no aplican multimodalidad concurrente, no se hace necesario ningún tipo de sincronización.
- Los mensajes de salida son cortos, concretos y específicos. Cuando se solicite un datos de entrada se hará en forma de pregunta.
- Establecer las características de estilo para cada salida según su modalidad.
- La presentación de la aplicación se implementó a través de una interface gráfica que permite la interacción por medio de la voz, el teclado y un dispositivo apuntador, gracias a las capacidades que presenta el lenguaje XHTML + Voice..

Componente de Estilos: Las pausas en la voz, y las inflexiones se implementaron haciendo uso de los signos de puntuación, gracias a las características que posee el plugin de IBM al interpretarlos como elementos de estilo en el momento de la salida por medio de la voz.

Componente de Presentación: De acuerdo a las modalidades definidas se decidió combinar la voz con el formato del formulario de captura, elaborando la parte gráfica con html y la voz con XHTML + Voice, como se detalló en la parte inicial del administrador de interacción. Al tomar el foco el respectivo control, por medio del elemento <prompt> se presenta un mensaje que solicita la información pertinente, como se muestra a continuación:

```
<vxml:prompt timeout = "10s" >
        What is the level of education?
</vxml:prompt>
```

Debido al contexto del problema, no se hizo necesario coordinar salidas a través de diferentes medios, quedando la implementación de SMIL para trabajos futuros.

4.5 MODELO SEMÁNTICO

4.5.1 Diseño de la ontología

Para definir nuestra ontología utilizamos hacemos un desarrollo conducido por los pasos establecidos para le modelado semántica. En la Web existen varias herramientas para la edición de ontologías expuestas en el anteproyecto de este trabajo. Entre las mas conocidas y populares se encuentra el Protegé. Nuestra implementación utiliza esta herramienta pues mas que un simple editor es un poderoso IDE con conexión a razonadores DIG.

Para explicar el proceso hagamos los siguientes pasos:

Paso 1. Determinar el dominio y el alcance de la ontología.

En el modelo del negocio está plasmado de manera general cuál es el dominio a representar. El problema del desplazamiento como fenómeno social y con los factores de riesgo que contiene, incluye tres estados de las personas:

Desplazado - Individuo cuyo traslado de territorio de residencia está motivado por razones relacionadas con el conflicto armado colombiano.
Vulnerable - Individuo que reside en un área del territorio nacional que el gobierno colombiano considera como área de conflicto permanente por la presencia de grupos regulares al margen de la ley (Guerrilla, Paramilitares)
Normal – Individuo que reside en un área del territorio nacional considerada como territorio de paz. En la misma el estado colombiano tiene el monopolio de las armas y hace cumplir la constitución.

La finalidad de la representación de este conocimiento es determinar la composición demográfica de la población desplazada y las amenazas que este fenómeno representa para la estabilidad social y política de la ciudad.

Paso 2. Considerar la reutilización de ontologías existentes

El fenómeno del desplazamiento forzado por razones de conflicto armado (hasta donde hemos investigado) no ha sido modelado de un modo semántica. Las ontologías han abarcados temas de la ciencias médicas, de la geografía, de los alimentos, etc., pero poco han mirado hacia los aspectos sociales. Por estas razones el re-uso en nuestro proceso se ha visto limitado a la utilización de los elementos del Dublín Core Group para metadatos.

Paso 3. Enumerar términos importantes en la ontología

A parte de los tres elementos descrito, el fenómeno del desplazamiento tiene, por supuesto, una relación con el territorio nacional colombiano. Las clasificación del territorio como un posible originador de desplazados o no es un aspecto relevante. Ahora bien, para representar el territorio nacional debe tomarse en cuenta la división política administrativa del país.

Departamento:

> Son entidades territoriales que tienen autonomía para la administración de los asuntos seccionales y la planificación y promoción del desarrollo económico y social dentro de su territorio en los términos establecidos por la Constitución. Los departamentos ejercen funciones administrativas, de coordinación, de complementariedad de la acción municipal, de

intermediación entre la Nación y los Municipios y de prestación de los servicios que determinen la Constitución y las leyes.[2]

Municipalidad:

Municipio: Es la entidad territorial fundamental de la división político-administrativa del Estado, con autonomía política, fiscal y administrativa dentro de los límites que le señalen la Constitución y las leyes de la República.

Corregimiento Departamental: Es una división del departamento, la cual incluye un núcleo de población. Según esta misma norma, los ahora corregimientos departamentales no forman parte de un determinado municipio.

Área Metropolitana: Es una entidad administrativa, formada por un conjunto de dos o más municipios integrados alrededor de un municipio núcleo o metrópoli, vinculados entre sí por estrechas relaciones de orden físico, económico y social, que para la programación y coordinación de su desarrollo y para la racional prestación de sus servicios públicos requiere una administración coordinada. Las áreas metropolitanas están dotadas de personería jurídica de derecho público, autonomía administrativa y patrimonio y autoridades propias.

Distritos

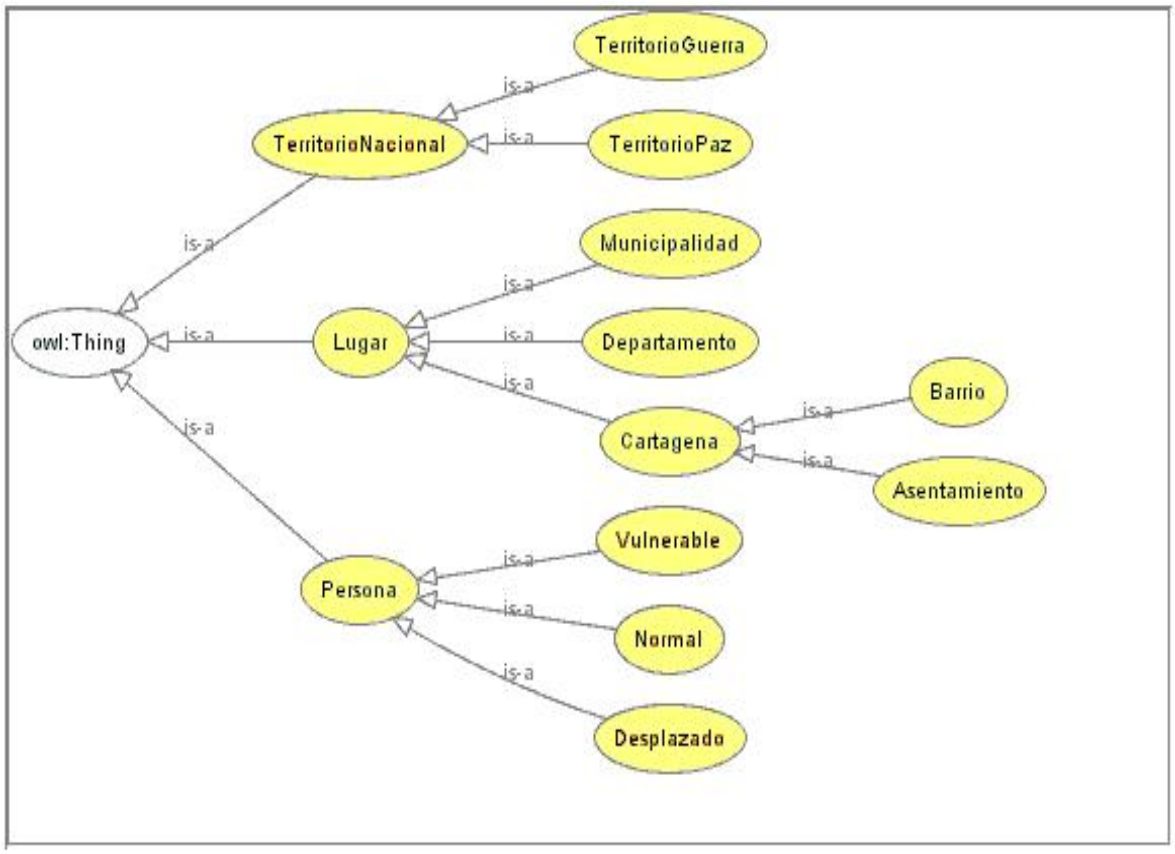
1. Atlántico: Barranquilla, Distrito Especial, Industrial y Portuario, desde 1993.
2. Bolívar: Cartagena, Distrito Turístico y Cultural, desde 1991.
3. Cundinamarca: Bogotá, Distrito Especial, desde 1954 hasta 1991 y en adelante Distrito Capital.
4. Magdalena: Santa Marta, Distrito Turístico, Cultural e Histórico, desde 1991.

Paso 4. Definir las clases y su jerarquía

Con los elementos determinados podemos definir una taxonomía para representar nuestro dominio de conocimiento.

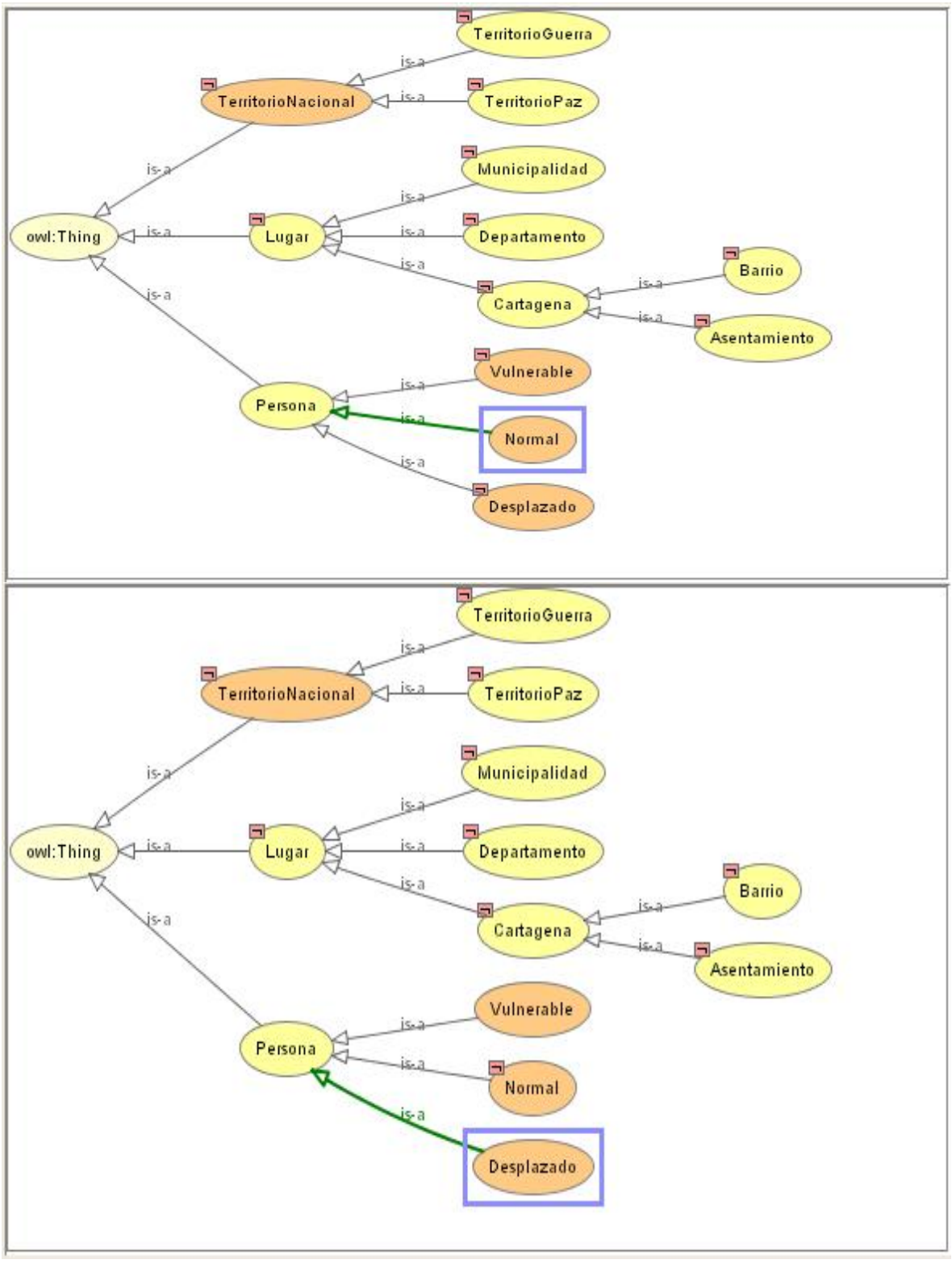
[2] Instituto Geográfico Agustín Codazzi. [en línea]
http://www.igac.gov.co:8080/igac_web/contenidos/division_politico_administrativa.jsp?idMenu=109

Figura Nº 35. Taxonomía



El modelo inicial representa la relación *is-a* entre las clases. En OWL todas las clases hereden de otra, por eso la clases *Persona*, *Lugar* y *TerritoriNacional* heredan de la clase abstracta *Thing*. La clase Persona es la razón de estudio. Sus descendientes son personas que se diferencian entre sí por al menos una característica. Estas tres clases en su definición incluyen elementos relacionados con la geografía nacional y las características del área. Esto genera la aparición de la clase *TerritorioNacional* con su correspondiente especificación en *TerritorioPaz* y *TerritorioGuerra*. Por norma el territorio nacional colombiano esta estructurado según INAC y sus elemetos fundamentales son *Departamento* y *Municipalidad*.

Figura Nº 36. Disjunción en subclases de Persona.



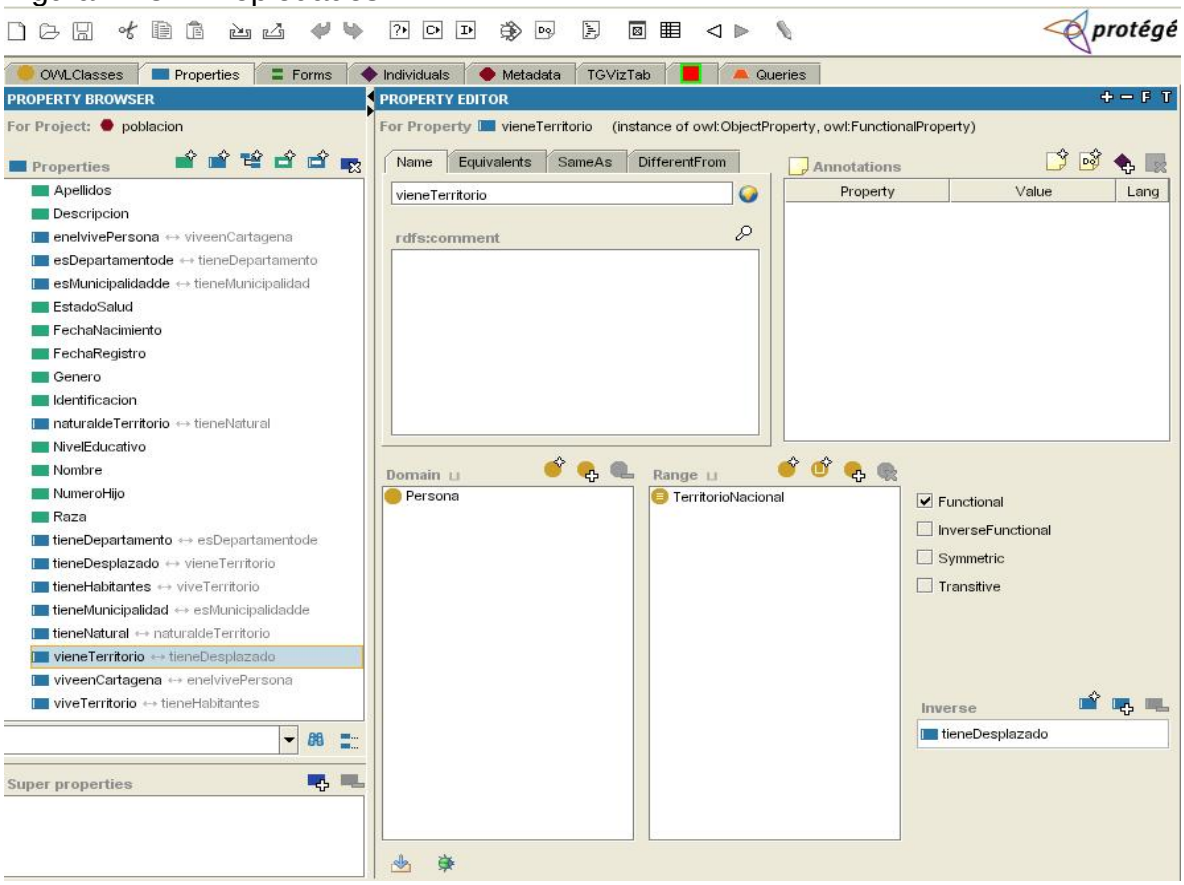
El problema a estudiar está orientado hacia la ciudad de Cartagena de Indias, por lo a pesar de estar clasificado como un territorio nacional, es una instancia que aquellos individuos que la posean como territorio de habitación y sean desplazados, requieren de mas información asociada a este territorio. Por este motivo Cartagena es una clase que se describe por Barrios y Asentamientos.

Aunque la mayoría de las clases hermanas son disjuntas y por ende lo son entre ramas de la jerarquía, el fenómeno del desplazamiento en general no incluye solo a Cartagena. Bajo este análisis y con la realidad que los grupos al margen se mueven por diferentes áreas, una persona Desplazada puede verse amenazada nuevamente por los factores del conflicto. Por este motivo las clases Vulnerable y Desplazado no son disjuntas.

Paso 5. Definir las propiedades de las clases

Las propiedades más sencillas de determinar son las propiedades “intrínsecas”. Las clases individuos de las clases presentadas requieren: Nombre, Apellidos, Descripción. El problema requiere conocer la cantidad de hijos de las personas, su estado de salud, etc.

Figura Nº 37. Propiedades

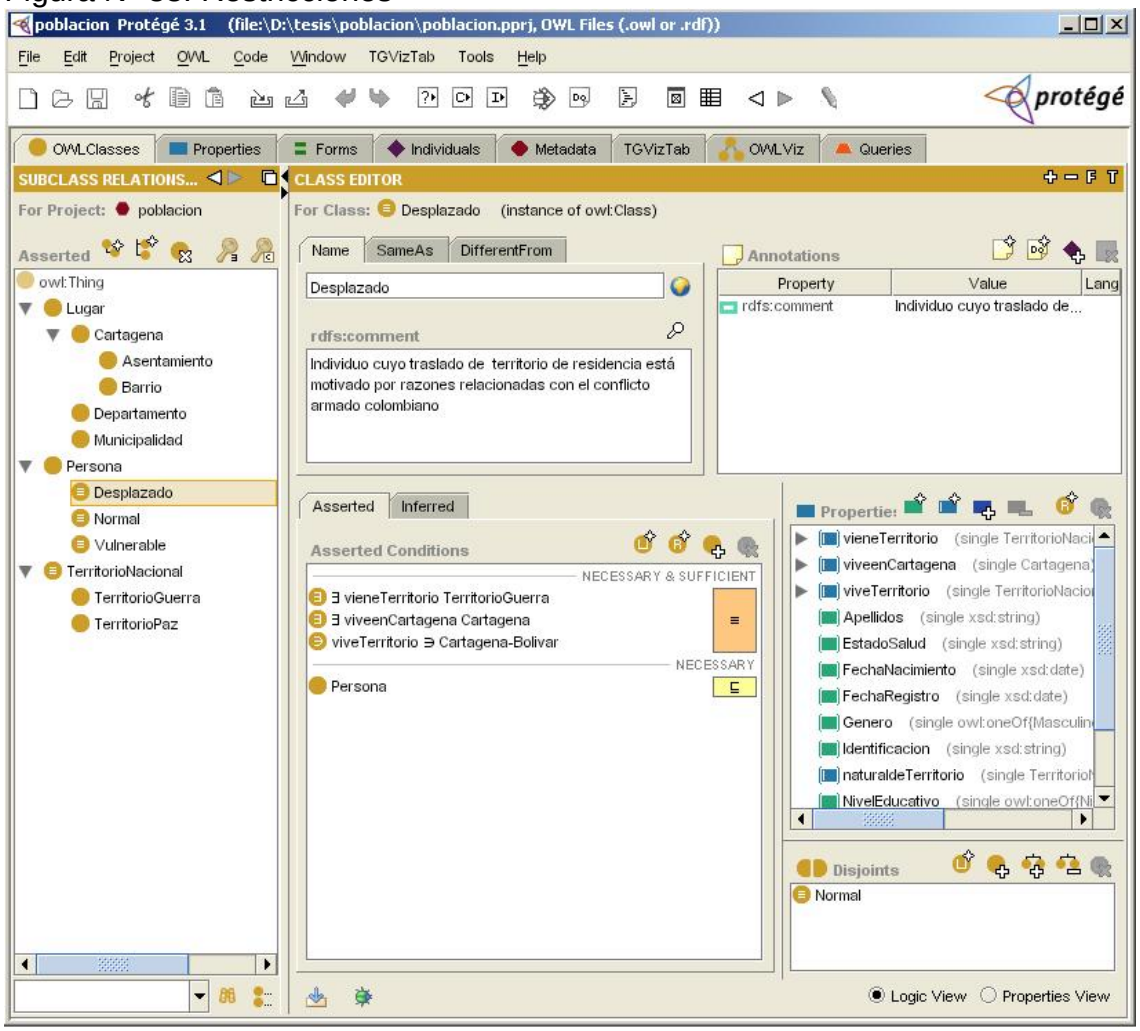


Las tres características determinantes son: donde vive, si viene desplazado de algún sitio y si vive en Cartagena de Indias en algún bario a Asentamiento. Estas son propiedades que van a relacionar individuos. Una buena práctica ya mencionada es definir para estas propiedades, propiedades inversas.

Paso 6. Definir las restricciones de las propiedades y las restricciones de las clases

Las propiedades en nuestro modelo solo aceptan un valor por lo que estas deben ser establecidas como Funcionales. En este punto, la ontología está casi lista. Solo necesita de los elementos que le van a permitir al razonador inferir conocimiento.En Protegé a este proceso se realiza estableciendo para las clases sus condiciones necesarias y suficientes.Las condiciones necesarias y suficientes de nuestras tres clases principales son:

Figura Nº 38. Restricciones



Normal – Un individuo *Normal viveTerritorio TerritorioPaz*

Vulnerables – Un individuo *Vulnerable viveTerritorio TerritorioGuerra*
Desplazado – Un individuo *Desplazado vieneDesplazado TerritorioGuerra* y *viveTerritorio tiene el valor Cartagena-Bolivar* y *viveCartagena Cartagena*

Las propiedades puedes tener diferentes restricciones que describen el tipo de valor, valores admitidos, el número de los valores (cardinalidad), y otras características de los valores que las propiedades pueden tomar.

Para definir estas restricciones se debe tomar en cuenta:

- Funcionalidad. Esta puede determinar si la propiedad admite solo uno o varios valores.
- Tipo de datos. Los tipos de datos son muy sencillos para cualquier conocedor de las herramientas de computación. Una persona familiarizada con el Excel conoce que tipos de datos se pueden representar. Técnicamente estos tipo de datos están preestablecidos en el RDF.
- Dominio y rango. Estos conceptos, explicados en la capa ontológica, son los que van a establecer las relaciones entre las clases a través de las propiedades. Esto es la esencia del RDF.

Las restricciones de las clases son las encargadas de determinar si un individuo es o no miembro de la clase durante el proceso de razonamiento.

4.5.2 Diseño de la interacción lógica

La implementación del modelo semántica además de la ontología ya definida requiere de un razonador. Nuestro desarrollo se hace tomando el RacerPro 1.8.1 en una versión educativa por 180 días. El Racer nos permite a través de la API del Protegé para la conexión con razonadores DIG verificar que nuestra ontología es consistente y resuelve las preguntas básicas establecidas. El modelo se prueba desde una clase en Java y se inserta en la aplicación JSP desarrollada.

Figura Nº 39. RacerPro

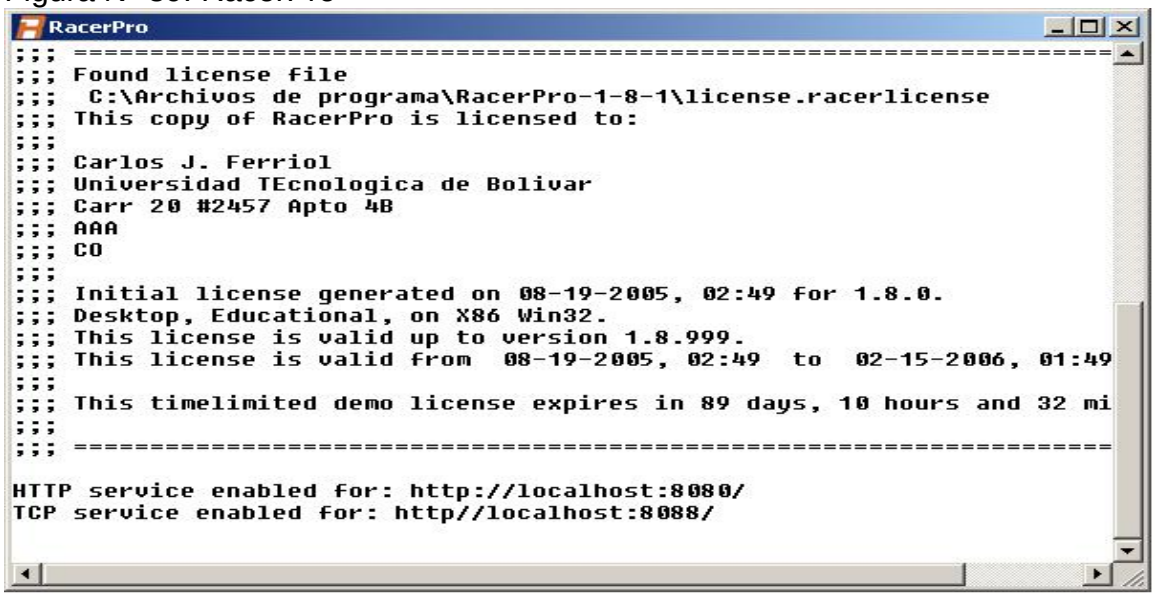
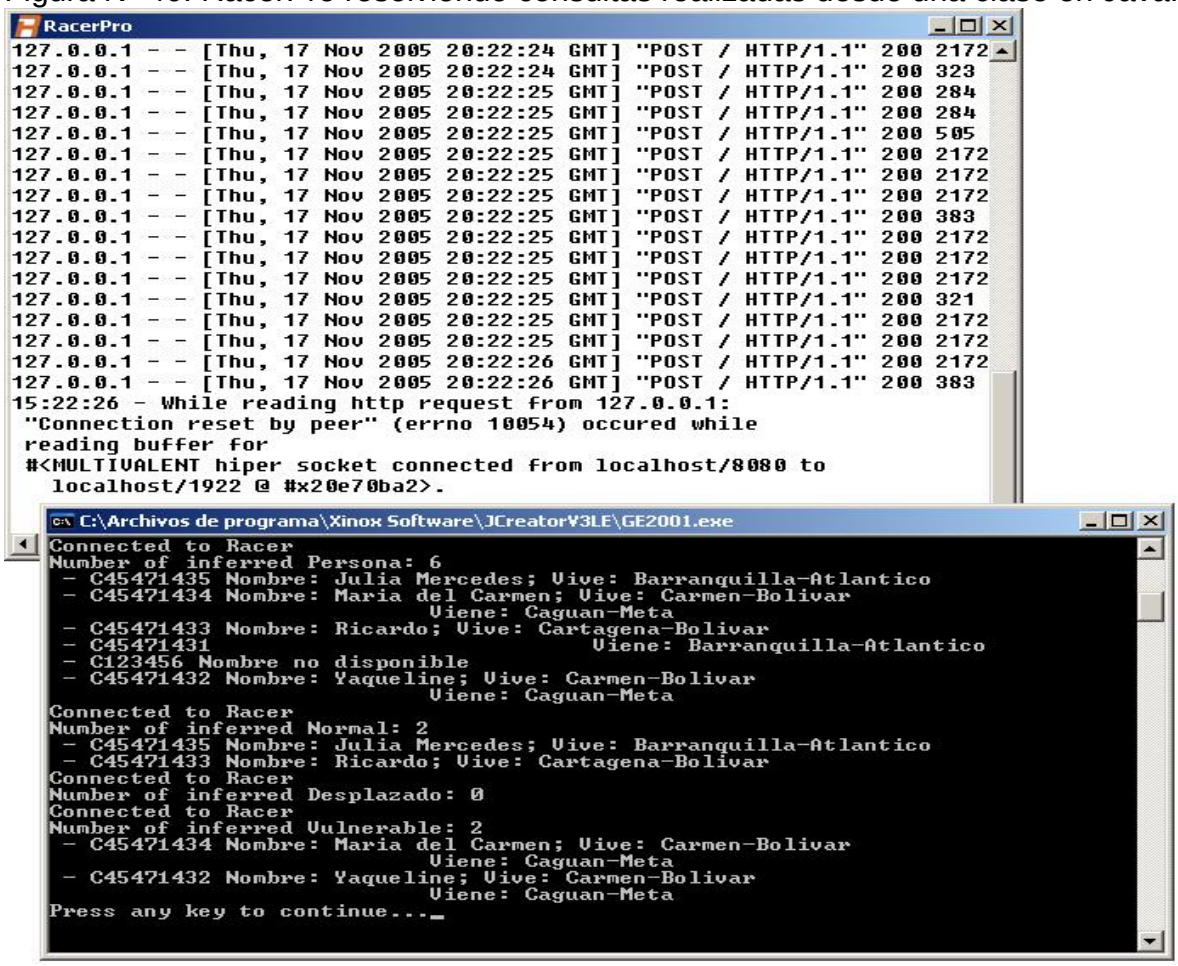


Figura Nº 40. RacerPro resolviendo consultas realizadas desde una clase en Java.



4.6 MODELO DE IMPLEMENTACIÓN

La implementación de la aplicación se realiza a partir del diagrama de componentes representado en la figura N 33. Como se puede apreciar, la vista de la aplicación se manejo por medio de formularios HTML y XHTML + Voice, específicamente con los archivos:

Index.htm: Es la página principal del portal
Menuportal.htm: Contiene el menú principal
Generalidades.htm: Presenta información general sobre la ciudad de Cartagena
Desplazamiento.htm; Muestra el problema de investigación para el Centro de investigación de la Universidad de San Buenaventura Cartagena
Cein.htm: Presenta información referente al Centro de investigación de la Universidad de San Buenaventura Cartagena
DatosPersonales.mxml: Se encuentra escrito en XHTML + Voice, corresponde a la interface de captura de los datos de la persona desplazada.
Consulta.htm: Presenta el resultado de la consulta a la ontología.

El control de la aplicación se realiza a través de scripts, servlets y el componente de interpretación semántica, como se describe a continuación

Funciones.js: Contiene una serie de funciones que permiten la validación de los datos de entrada.
Registro.java: Es el servlet que permite grabar los datos de las personas en la ontología

A su vez el componente de interpretación semántica, está conformado por los siguientes archivos escritos en XHTML + Voice,

Date.mxml: controla la captura de datos tipo fecha
Lugar_colombia.mxml: controla al captura de datos correspondientes a lugares válidos de Colombia
Digito.mxml: Controla la captura de datos numéricos digito por dígito.

El modelo de la aplicación está representado por el componente de reconocimiento, constituido por las gramáticas, y el modelo semántico de la aplicación. Se encuentra estructurado de la siguiente manera:

Date.js: Define la gramática para las fechas.
Lugar_colombia.mxml: Define la gramática para los lugares de Colombia
Digito.mxml: Define la gramática para valores registrados dígito por dígito.
Población.java Permite el acceso a la ontología para generar las consultas a través del razonador.
Poblacion.owl: Contiene la ontología

Poblacion.jar: Contiene las clases e interfaces necesarias para el manejo de la ontologìa
OWLPoblacionDoc: Documentación de la ontología población.owl en formato HTML.

4.7 EVALUACIÓN DEL PROTOTIPO

El prototipo fue evaluado permanentemente, utilizando la técnica de caja blanca, con el fin de identificar los errores de tipo lógico que presentaba. Las pruebas se realizaron a partir de los casos de uso, convirtiéndolos en casos de prueba, y haciendo énfasis en las gramáticas y restricciones que se definieron al construir el componente de reconocimiento.

Como situación particular, se pudo observar que no siempre lo que se decía era lo mismo registrado por la aplicación, sobre todo en el caso del las fechas de nacimiento, de registro, los lugares de nacimiento y de origen. Inicialmente se pensó que era resultado de una mala definición de la gramática, sin embargo, después de muchas pruebas realizadas por diferentes personas, se pudo observar la gran influencia que tiene la buena pronunciación, y el mínimo nivel de interferencia presente en el momento de la captura.

El manejo de las sesiones se deja bajo la responsabilidad del servidor de aplicaciones seleccionado, el cual debe permitir especificar el alcance de la sesión, según sea necesario, indicando si corresponde a una sesión de página, petición, sesión HTTP, o aplicación en general. Para identificar dicho alcance se sugiere tener en cuenta las reacciones de la aplicación ante las respuestas del usuario (correctas e incorrectas) o en ausencia de éstas.

CONCLUSIONES

Como resultado de la tesis se obtuvo una guía para el modelado y desarrollo de aplicaciones web multimodales que incluyen modelado de datos basado en web semántica, soportada por un estudio del estado del arte de los lenguajes de marcado para el desarrollo de aplicaciones multimodales, y un estudio aplicado de las capas ontológica y lógica de la web semántica, que fue probada a través de la construcción de un prototipo de portal de información sobre el problema de desplazamiento forzado en la ciudad de Cartagena, que incluye modelado semántico de los datos hasta la capa de reglas de la web semántica.

Teniendo en cuenta la plataforma de interacción multimodal que propone el Consorcio W3C, los lenguajes de marcado se clasificaron según dos criterios: La modalidad (Entrada por escritura manual: InkML, Entrada con Voz y DTMF: XHTML + Voice, VoiceXML, SALT, TalkML, VoxML, SpeechML, Salida con audio: SSML) y la Funcionalidad que representa en la interacción con el usuario final (Construcción de la interface de usuario: XForms, Descripción de la interpretación de la entrada de Usuario EMMA, Sincronización e integración de elementos multimedia SMIL). Algunos de estos lenguajes, como es el caso de VoiceXML, han llegado a un buen nivel de desarrollo, mientras que otros como InkML y EMMA, aún se encuentran en evolución, lo que se convierte en un inconveniente en el momento de utilizarlos.

Actualmente existen dos propuestas para el manejo de la interacción multimodal, la primera liderada por el Consorcio W3C, apoyada por entidades como IBM, y la segunda por dirigida por SALT Forum, apoyado por Microsoft. Se tuvo como referente la primera, debido a que presenta compatibilidad con todos los lenguajes de marcado existentes para la interacción multimodal, y además no limita la parte de programación a una tecnología específica, como es el caso de la segunda, que requiere de ASP (Active Server Pages) o tecnología .NET para el desarrollo de aplicaciones.

Al analizar a profundidad la capa ontológica y la capa lógica del modelo semántico se logró un mejor entendimiento de éste, comprendiendo el papel que desempeña cada una de ellas para la Web Semántica. En la capa ontológica se declara la información que servirá de base para el razonamiento de los agentes. La capa lógica va a determinar el mecanismo de inferencia. Para esta inferencia son suficientes, por el momento, los razonadores DIG. Las ontologías distribuídas son un gran adelanto hacia el deseo de compartir el conocimiento entre toda la comunidad digital. Aunque el modelo de Tim Berners-Lee empieza a tener contradictores, es un modelo sencillo que permite desarrollar las aplicaciones para la Web semántica.

Fundamentado en el patrón de diseño denominado Modelo Vista Controlador MVC, la guía se estructuró en dos partes, la primera destinada a definir el Modelo de la interacción multimodal, y la segunda dedicada al Modelado semántico. Esto se hace con el fin de aplicar los principios de modularidad e independencia funcional, que plantea la ingeniería de software, al realizar las tareas de modelar y desarrollar aplicaciones. A su vez, la primera parte de la guía tomó como referente la plataforma de interacción multimodal que propone el Consorcio W3C, para facilitar su compresión y aplicación. En la segunda se usó el documento de Natalya F. Noy and Deborah L. McGuinness, titulado Desarrollo de Ontologías-101: Guía Para Crear Tu Primera Ontología.

El prototipo construido permite el registro de personas desplazadas a través de una interface multimodal, y sirvió como ejemplo al momento de ilustrar e implementar la guía propuesta. Sin embargo, debido a las características propias del navegador y del componente de voz de IBM, la captura de los datos no siempre corresponde a lo que se desea registrar, sobre todo en aquellos datos cuya gramática es más compleja. Por otra parte, el no contar con un componente de voz en Español se convirtió en un gran inconveniente en el contexto del problema de desplazamiento forzado interno en la ciudad de Cartagena.

Al fusionar las características propias de la interacción multimodal con las ventajas de la web semántica, se logra poner a disposición de cualquier persona interesada, un conocimiento compartido en cualquier momento, en cualquier lugar y a través de cualquier dispositivo, independientemente de las limitaciones intrínsecas al entorno de ejecución de la aplicación, o del usuario final, puesto que le brinda la posibilidad de interantuar con diferentes modalidades ya sea en forma suplementario o complementaria. De igual manera, la fusión de estos dos tipos de tecnologías permite generar interfaces multimodales inteligentes capaces de corregir y mejorar el proceso de reconocimiento de la modalidad de entrada, a partir de consultas específicas que se hacen al modelo semántico representado a través de la ontología.

Aunque la tecnología sirve de apoyo para el tratamiento de problemáticas sociales, es muy compleja la solución de un problema tan crítico como el desplazameinto forzado en todo el país, debido a los componentes políticos, económicos y culturales que generan las dinámicas sociales. Este trabajó permitió a los investigadores comprender mejor dicha problemática, que desafortunadamente se ha convertido en algo normal en la sociedad colombiana.

TRABAJOS FUTUROS

Se hace necesario incluir en la guía un desarrollo más detallado del componente de Entorno y sistema, involucrando el trabajo del grupo del Consorcio W3C denominado Device independence Work Group, en combinación con la implementación del lenguaje EMMA, que permita una interpretación e integración de la voz y la escritura manual, capturada y representada por medio del lenguaje InkML. De igual manera, la guía se puede complementar, incluyendo salidas que requieran múltiples modalidades sincronizadas, para aplicar las ventajas que representan los lenguajes de Marcado SSML, SVG y SMIL. En el caso del modelado semántico, se requiere incluir la aplicación de un lenguaje de consulta a la ontología.

De igual manera, es evidente la necesidad de contar con un componente para el manejo de la voz en español, sobre todo para del desarrollo de soluciones a problema tan particulares, como lo es el desplazamiento forzado interno en la ciudad de Cartagena.

Nuevos modelos de la Web semántica (2 torres) son presentados a la comunidad y aunque a nivel horizontal mantiene la estructura del modelo clásico, es necesario validar este modelo y adicionarle a los estudios ya realizados los nuevos componentes.

Al Centro de Investigación de la Universidad de San Buenaventura Cartagena se le recomienda constuir un sistema de administación del conocimiento, a partir de la ontología desarrollada, que permita interacción multimodal, facilitando los procesos de captura de información en territorios de difícil acceso.

BIBLIOGRAFÍA

[1] ATIS Telecom Glossary 2000. Dual-Tone Multifrequency (DTMF) [en línea] <http://www.atis.org/tg2k/_dual-tone_multifrequency_signaling.html> (Consulta: 8 de junio de 2005)

[2] ALEOTTI, Jacopo. BOTTAZZI, Stefano. et all. A multimodal user interface for remote object exploration in teleoperation systems [en línea] <http://rimlab.ce.unipr.it/publications/HUROIN2002.pdf> (Consulta: 8 de enero de 2005)

[3] AXELSSON, Jonny y CROSS Chris. Michael. Et all. Multimodal Architecture and Interfaces. [en línea] http://www.w3.org/TR/mmi-arch/ (Consulta: 5 de mayo de 2005)

[4] _____. XHTML 2.0. [en línea] <http://www.w3.org/TR/2005/WD-xhtml2-20050527/Overview.html#toc> (Consulta: 18 noviembre, 2004)

[5] _____. XHTML+Voice Profile 1.0. [en línea] <http://www.w3.org/TR/xhtml+voice/> (Consulta: 21 noviembre, 2004)

[6] _____. Chris. XHTML+Voice Profile 1.1. . [en línea] http://www-306.ibm.com/software/pervasive/multimodal/x+v/11/spec.htm (Consulta: Abril 10 de 2005)

[7] BECHHOFER, Sean. OilEd 3.4 Manual. [en línea] <http://oiled.man.ac.uk/docs/Manual.pdf> (Consulta: 21 diciembre, 2004)

[8] BECKET, Dave. RDF/XML Syntax Specification. [en línea] <http://www.w3.org/TR/rdf-syntax-grammar/> (Consulta: 22 noviembre, 2004)

[9] BELL LABORATORIES. SABLE: A Synthesis Markup Language (version 1.0). .[en línea] http://www.bell-labs.com/project/tts/sable.html (Consulta: Abril 2 de 2005)

[10] BELROSE, Guillaume. Introduction to TalkML [en Línea] <http://www.w3.org/Voice/TalkML/> (Consulta: Marzo 14 de 2005)

[11] BERNERS-LEE, Tim. Semantic Web Road map. [en línea]. <http://www.w3.org/DesignIssues/Semantic.html> (Consulta: 9 diciembre, 2004)

[12] _____. Semantic Web – XML2000. [en línea] <http://www.w3.org/2000/Talks/1206-xml2k-tbl/Overview.html> (Consulta: 9 noviembre, 2005)

[13] _____. Web for real people.[en línea] <http://www.w3.org/2005/Talks/0511-keynote-tbl/> (Consulta: 2 noviembre, 2005)

[14] BODEL, Michael. JOHNSTON, Michael. Et all. W3C Multimodal Interaction Framework [en línea] http://www.w3.org/TR/mmi-framework/ (Consulta: 5 de mayo de 2005)

[15] BOLEY, Harold y GROSOF Benjamín. The RuleML Design. [en línea]. <http://www.ruleml.org/indesign.html> (Consulta: 5 enero, 2005)

[16] BOZSAK, Erol. EHRIG Marc, Siegfried HANDSCHUH, Andreas HOTHO, et all. KAON - Towards a large scale Semantic Web. [en línea] http://www.aifb.uni-karlsruhe.de/ ~sst/Research/Publications/dexa2002.pdf (Consulta: 5 de noviembre, 2005)

[17] BULTERMAN, Dick. GRASSEL, Guido et al, Synchronized Multimedia Integration Language (SMIL 2.1) [en línea]. http://www.w3.org/TR/2005/WD-SMIL2-20050201/ (Consulta: 5 de noviembre, 2005)

[18] BURNETT, Daniel C. WALKER, Mark. HUNT Andrew. Speech Synthesis Markup Language (SSML) Version 1.0 [en línea] http://www.w3.org/TR/speech-synthesis/ (Consulta: Abril 8 de 2005)

[19] CARRILLO, Eduardo. Framework for ubiquitous and voice enabled web applications development: A Transport Information Systems Test-Bed. [en línea] <http://www.tdx.cesca.es/TESIS_UV/AVAILABLE/TDX-0530105-170506//CARRILLO.pdf> (Consulta: 2 diciembre, 2004)

[20] CASTELLS, Pablo. Aplicación Técnica de la Web Semántica. [en línea] <http://giig.ugr.es/~mgea/coline02/Articulos/pcastells.pdf> (Consulta: 10 diciembre, 2004)

[21] _____ La Web Semántica. . [en línea] <http://www.ii.uam.es/~castells/publications/castells-uclm03.pdf> (Consulta: 9 diciembre, 2004)

[22] CHEE, yi-Min y PRASAD, Said. Requirements for the Ink Markup Language. [En línea] <http://www.w3.org/TR/inkreqs/> (Consulta: febrero 2 de 2005)

[23] CHOU, Wu. DAHL, Deborah A. Et al. Extensible MultiModal Annotation markup language [En línea] http://www.w3.org/TR/2004/WD-emma-20040901/ (Consulta: Abril 10 de 2005)

[24] DAHL, D. Summary of SALT 1.0 Elements [en línea] http://www.saltforum.org/devforum/tutorials/tutorials/SALTSummary2.asp (Consulta: Marzo 31 de 2005)

[25] DUBINKO, Micah. SCHNITZENBAUMER, Sebastián. Et al. XForms Requirements http://www.w3.org/TR/2000/WD-xhtml-forms-req-20000329 (Consulta: Abril 19 de 2005)

[26] _____. XForms 1.0: Data Model [en línea] http://www.w3.org/TR/2000/WD-xforms-datamodel-20000406/ (Consulta: Abril 19 de 2005)

[27] _____. XForms 1.0: Data Model [en línea] http://www.w3.org/TR/2000/WD-xforms-datamodel-20000815/ (Consulta: Abril 19 de 2005)

[28] DUMBILL. Edd. Building the Semantic Web, Marzo 07 de 2001. [en línea] http://www.xml.com/pub/a/2001/03/07/buildingsw.html (Consulta: 10 de noviembre, 2005)

[29] DUTOIT Thierry. A Short Introduction to Text-to-Speech Synthesis. [en línea] http://tcts.fpms.ac.be/synthesis/introtts.html (Consulta: 4 de abril, 2005)

[30] EISENZOPF, Jonathan. VoiceXML Developer Series, Introduction. [en línea] <http://www.developer.com/voice/article.php/1570051> (Consulta: Marzo 14 de 2005)

[31] EURESCOM. Eurescom MUST project Multimodal, multilingual information services for small mobile terminals.
[en línea] http://www.loria.fr/projets/JEP-TALN/actes/TALN/conf_assoc/Ecrit_Oral01.pdf (Consulta: 8 de enero, 2005)

[32] FROUMENTIN, Max, Ink Markup Language. [en línea] <http://www.w3c.org/2002/mmi/ink> (Consulta: febrero 20 de 2005)

[33] GARSHOL, Lars Marius. Topic maps, RDF, DAML, OIL. [en línea]. <http://www.ontopia.net/topicmaps/materials/tmrdfoildaml.html> (Consulta: 19 diciembre, 2004)

[34] GRUBER Thomas., A Translation Approach to Portable Ontology Specifications, Knowledge Acquisition, 5 (2), 199-220, 1993b. . [en línea]

<http://www-ksl.stanford.edu/kst/what-is-an-ontology.html> (Consulta: 9 diciembre, 2004)

[35] GUARINO, Nocola. Understanding, Building, and Using Ontologies. [en línea] <http://www.loa-cnr.it/Papers/FOIS98.pdf>(Consulta: 9 diciembre, 2004)

[36] HAASE, Peter. A Comparison of RDF Query Languages. [en línea] <http://www.aifb.uni-karlsruhe.de/WBS/pha/rdf-query/> (Consulta: 5 enero, 2005)

[37] HAYES, Patrick. REF Semantics. [en línea] <http://www.w3.org/TR/rdf-mt/> (Consulta: 22 noviembre, 2004)

[38] HEFLIN, Jeff. OWL Web Ontology Language Use Cases and Requirements. [en línea]. <http://www.w3.org/TR/webont-req/> (Consulta: 27 diciembre, 2004)

[39] HIRTLE, David. SWRL RuleML Accessing SWRL Properties as "Foreign" Atoms. [en línea]. <http://www.ruleml.org/swrl/> (Consulta: 5 enero, 2005)

[40] HORROCKS, Ian. DAML+OIL: a Description Logic for the Semantic Web. [en línea] <http://www.cs.man.ac.uk/~horrocks/Publications/download/2002/ieeede2002.pdf> (Consulta: 19 diciembre, 2004)

[41] _____.y PATEL-SCHNEIDER, Peter F. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. [en línea]. <http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/> (Consulta: 5 enero, 2005)

[42] HUNT, Andrew. JSpeech Markup Language [en lìnea] http://www.w3.org/TR/2000/NOTE-jsml-20000605/ (Consulta: Abril 2 de 2005)

[43] _____. Speech Synthesis Markup Requirements for Voice Markup Languages [en línea] http://www.w3.org/TR/1999/WD-voice-tts-reqs-19991223/ (Consulta: Abril 2 de 2005)

[44] _____. Speech Recognition Grammar Specification [en línea] http://www.w3.org/TR/speech-grammar/ (Consulta: Abril 2 de 2005)

[45] HUNTER, Jane. DRENNAN, John. LITTLE, Suzanne. Realizing the Hydrogen Economy through Semantic Web Technologies. [en línea] http://maenad.dstc.edu.au/papers/2004/ieee-is.pdf (Consulta: 8 de enero de 2005)

[46] IBM. Ontology Management System. [en línea] http://www.alphaworks.ibm.com/tech/snobase (Consulta: 2 de abril de 2005)

[47] _____. WebSphere Voice Application Access. [en línea] http://www-306.ibm.com/software/pervasive/ws_vaa/ (Consulta: 2 de noviembre de 2005)

[48] INTERNATIONAL BUSINESS MACHINE. Why IBM Leadership in Multimodal. [en línea] <http://www-306.ibm.com/software/pervasive/multimodal/> (Consulta: 10 noviembre, 2004)

[49] ISHIKAWA, Masayasu, XForms - The Next Generation of Web Forms [En línea] http://www.w3.org/MarkUp/Forms/ (Consulta: Abril 8 de 2005)

[50] JENA. Semantic Web Framework. [En línea] http://jena.sourceforge.net/index.html (Consulta: Abril 11 de 2005)

[51] JRDF [En línea] http://jrdf.sourceforge.net/index.html (Consulta: Abril 11 de 2005)

[52] KALYANPUR, Aditya y SIRIN, Evren. Hypermedia Inspired Ontology Engineering Environment: SWOOP**.** [en línea]. <http://www.mindswap.org/papers/SWOOP-Poster.pdf> (Consulta: 21 diciembre, 2004)

[53] KLYNE, Graham y CARROLL Jeremy. Resource Description Framework (RDF): Concepts and Abstract Syntax. [en línea] <http://www.w3.org/TR/rdf-concepts/> (Consulta: 22 noviembre, 2004)

[54] KNOWLEDGE AND INFORMATION MANAGEMENT. The KIM Platform Introduction. [en línea] <http://www.ontotext.com/kim/introduction.html> (Consulta: 20 diciembre, 2004)

[55] _____. The KIM Platform for Knowledge & Information Management. [en línea]. <http://www.ontotext.com/kim/> (Consulta: 20 diciembre, 2004)

[56] KOIVUNEN, Marja Riitta y MILLER Eric. W3C Semantic Web Activity. [en línea] <http://www.w3.org/2001/12/semweb-fin/w3csw> (Consulta: 21 noviembre, 2004)

[57] _____. Accessibility Features of SMIL [En línea] http://www.w3.org/TR/SMIL-access/ (Consulta: Abril 10 de 2005)

[58] KUCHLING, A.M. Introduction to the Semantic Web and RDF [en línea] <http://www.amk.ca/talks/2004-12-02/> (Consulta: 2 noviembre, 2005)

[59] LARSON, James A. Commonsense Guidelines for Developing Multimodal User Interfaces. [en línea] <http://www.larson-tech.com/MMGuide.html> (Consulta: 15 de julio de 2005)

[60] _____. y RAMAN, T. V. W3C Multimodal Interaction Framework. [en línea] <http://www.w3.org/TR/mmi-framework/> (Consulta: 12 noviembre, 2004)

[61] LASSILA, Ora y SWICK Ralph R. Resource Description Framework (RDF) Model and Syntax Specification. [en línea] <http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/#model> (Consulta: 22 noviembre, 2004)

[62] LILLEY Chris, JACKSON Dean. Scalable Vector Graphics (SVG) XML Graphics for the Web. [en línea] <http://www.w3.org/Graphics/SVG/> (Consulta: 22 mayo, 2005)

[63] LUCENT TECHNOLOGIES. BELL LABS INNOVATIONS. AT&T, Lucent Technologies, and Motorola create VXML Forum; companies seek open standard to promote voice access to web services. [en línea] http://www.lucent.com/press/0399/990302.bla.html (Consulta: Marzo 15 de 2005)

[64] MAES, Stéphane H., POTTER, Stephen. Requirements for EMMA [En línea] http://www.w3.org/TR/EMMAreqs/ (Consulta: Abril 8 de 2005)

[65] MANOLA, Frank y MILLER Eric. RDF Primer. [en línea] <http://www.w3.org/TR/2004/REC-rdf-primer-20040210/#collections> (Consulta: 22 noviembre, 2004)

[66] MCBRIDE, Brian. RDF Semantics. [en línea] <http://www.w3.org/TR/rdf-mt/> (Consulta: 19 diciembre, 2004)

[67] McGLASHAN, Scott y BURNETT, Daniel.Voice Extensible Markup Language (VoiceXML) Version 2.0 [En línea] < http://www.w3.org/TR/2004/REC-voicexml20-20040316/> (Consulta: febrero 2 de 2005)

[68] MCGUINNESS, Debora L. OWL Web Ontology Language Overview. [en línea]. <http://www.w3.org/TR/owl-features/> (Consulta: 27 diciembre, 2004)

[69] MICROSOFT. Speech Server. [en línea] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnanchor/html/netspeechanchor.asp (Consulta: 2 de noviembre de 2005)

[70] MOTOROLA INC, VoxML 1.1 Language Reference. [en línea] www.w3.org/Voice/1999/VoxML.pdf (Consulta: Marzo 16 de 2005)

[71] NOY, Natalya F. MCGUINNESS, Deborah L. Ontology Development 101: A Guide to Creating Your First Ontology

[72] OPENRDF. Sesame. [En línea] http://openrdf.org/ (Consulta: 2 de abril de 2005)

[73] OWL Web Ontology Language Guide [en línea] <http://www.w3.org/TR/2004/REC-owl-guide-20040210/> (Consulta: 2 de noviembre de 2005)

[74] PAT Hayes, <http://robustai.net/papers/Monotonic_Reasoning_on_the_Semantic_Web.html> (Consulta: 2 de noviembre de 2005)

[75] PIERACCINI, Roberto. CHOU, Wu. et al. MMA: Extensible MultiModal Annotation markup language [en línea] http://www.w3.org/TR/emma/ (Consulta: Abril 8 de 2005)

[76] PRUD'HOMMEAUX y Eric. SEABORNE, Andy. SPARQL Query Language for RDF [en línea]. <http://www.w3.org/TR/rdf-sparql-query/> (Consulta: 5 enero, 2005)

[77] RAGGETT, Dave. Voice Browsers. [en línea] http://www.w3.org/Mobile/1998/Workshop/Slide/VoiceBrowsers/slide17.htm (Consulta: Abril 2 de 2005)

[78] RAMAN, T. V. XForms Components. [en línea] <http://safariexamples.informit.com/0321154991/bookse3.html> (Consulta: 5 diciembre, 2004)

[79]_____. XForms XML Powered Web Forms [en línea] http://safariexamples.informit.com/0321154991/bookse58.html (Consulta: 26 de julio de 2005)

[80] REITHINGER, Norbert y SONNTAG, Daniel. An Integration Framework for a Mobile Multimodal Dialogue System Accessing the Semantic Web. [en Línea]. <http://www.dfki.de/~bert/interspeech2005.pdf > (Consulta: noviembre 6 de 2005)

[81] ROBIN, Michael y LARSON, Jim. Voice Browsers. An Introduction and Glossary for the reqirements drafts. [en Línea]. <http://www.w3.org/TR/1999/WD-voice-intro-19991223/> (Consulta: Marzo 14 de 2005)

[82] SALT FORUM PRESS RELEASE. Industry Leaders Join to Accelerate Widespread Adoption of Speech and Graphical Interaction With HTML, xHTML and XML-Based Applications and Web Services. [en línea] http://www.saltforum.org/press/011015.asp (Consulta: Marzo 31 de 2005)

[83] _____. The SALT Forum Welcomes Additional Technology Leaders as Contributors [en línea] http://www.saltforum.org/press/020131.asp (Consulta: Marzo 31 de 2005)

[84] _____. Speech Application Language Tags (SALT). Technical White Paper [en línea] http://www.saltforum.org/whitepapers/whitepapers.asp (Consulta: Marzo 31 de 2005)

[85] _____. SALT Forum Publishes Speech Application Language Tags Specification Version 1.0 [en línea] http://www.saltforum.org/press/020715.asp (Consulta: Marzo 31 de 2005)

[86] _____. SALT Forum Contributes Speech Application Language Tags Specification Version 1.0 to World Wide Web Consortium [en línea] http://www.saltforum.org/press/020813.asp (Consulta: Marzo 31 de 2005)

[87] _____. Salt Forum advances Mobile Content Delivery with Enhancements to Scalable Vector Graphics Specification [en línea] http://www.saltforum.org/press/030624.asp (Consulta: Marzo 31 de 2005)

[88] SCHNITZENBAUMER, Sebastián. WEDEL, Malte. RAGGETT, Dave. XHTML™ Extended Forms Requirements. [en línea] http://www.w3.org/TR/1999/WD-xhtml-forms-req-19990906 (Consulta: Abril 19 de 2005)

[89] SMIL PRESS. W3C Issues First Public Draft of Synchronized Multimedia Integration Language (SMIL) [En línea] http://www.w3.org/Press/SMIL (Consulta: Abril 10 de 2005)

[90] Speech Application Language Tasgs (SALT) 1.0 Specification [en línea] <http://www.saltforum.org/saltforum/downloads/SALT1.0.pdf> (Consulta: Marzo 14 de 2005)

[91] SPROAT, R., HUNT, A et al. SABLE: a standard for TTS MarkuP. [en línea] .http://www.bell-labs.com/project/tts/sabpap/sabpap.html (Consulta: Abril 2 de 2005)

[92] STANFORD MEDICAL INFORMATICS. Protégé Education. [en línea]. <http://protege.stanford.edu/> (Consulta: 20 diciembre, 2004)

[93] STÉPHANE, H. Maes y VIJAY Saraswat. Multimodal Interaction Requirements. [en línea] <http://www.w3.org/TR/mmi-reqs/> (Consulta: 12 noviembre, 2004)

[94] TECHNOLOGY REPORTS. VoxML Markup Language. [en Línea] <http://xml.coverpages.org/voxML.html> (Consulta: Marzo 16 de 2005)

[95] THIERRY, Michel. Synchronized Multimedia Activity Statement. [En Línea]. http://www.w3.org/AudioVideo/Activity.html (Consulta: Abril 10 de 2005)

[96] VUJOSEVIC, Srdjan y LABERGE, Robert. VoxML: Ger your Database Talking. [en línea] <http://www.webtechniques.com/archives/2001/02/laberge/> (Consulta: Marzo 15 de 2005)

[97] WALKER, Mark R. HUNT, Andrew. Speech Synthesis Markup Language Specification for the Speech Interface Framework [en línea] http://www.w3.org/TR/2000/WD-speech-synthesis-20000808 (Consulta: Abril 2 de 2005)

[98] WIKIPEDIA. Application programming interface [en línea]. http://en.wikipedia.org/wiki/Application_programming_interface (Consulta: Abril 2 de 2005)

[99] W3C. Device Independence, Accessibility and Multimodal Interaction. [en línea http://www.w3.org/2005/04/di_mmi_wai.html. (Consulta: 2 de noviembre de 2005)

[100] _____. Uniform Resource Identifier (URI) Activity Statement. [en línea <http://www.w3.org/Addressing/Activity> (Consulta: 2 de marzo de 2005)

[101] _____. Extensible Markup Language (XML). [en línea <http://www.w3.org/XML/> (Consulta: 13 de marzo de 2005)

[102] _____. Document Object Model (DOM). [en línea <http://www.w3.org/DOM/> (Consulta: 2 de marzo de 2005)

[103] _____. Cascading Style Sheets. [en línea <http://www.w3.org/Style/CSS/> (Consulta: 15 de marzo de 2005)

ANEXOS

CRONOGRAMA DE ACTIVIDADES

Actividad	Enero - Junio				Julio				Agosto				Septiembre				Octubre				Noviembre			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Revisión bibliográfica (Multimodalidad y Web Semántica)																								
Construcción anteproyecto																								
Construcción estado del arte de los lenguajes de marcado para el desarrollo de aplicaciones multimodales																								
Estudio aplicado de las capas ontológica y lógica de la web semántica.																								
Construcción guía para el modelado y desarrollo de aplicaciones web multimodales que incluyan modelado de datos basado en web semántica																								
Modelado del prototipo																								
Construcción del prototipo																								
Evaluación del prototipo																								
Corrección del prototipo																								
documentación																								
Entrega documento final																								

PRESUPUESTO

Debido a que los costos son asumidos por los investigadores, se hará uso de software libre, y no se incurrirá en costos de adquisición de equipos de cómputo. Los valores presentados a continuación se encuentran expresados en pesos colombianos.

Rubro	Costo
Acceso a Internet	240.000
Bibliografía	500.000
Materiales	500.000
Total	1.240.000

RECURSOS NECESARIOS

- Hardware
    - Dos computadores con velocidad de procesamiento 2.4 GHz, 256 MB de RAM, y disco duro de 40 GB
    - Conexión permanente a Internet
    - Impresora
    - Micrófono
    - Parlantes
- Software de desarrollo
    - Java
    - Java Server Page
    - Apache Tomcat
    - Protégé
    - RacerPro
    - Lenguajes de Marcado: XHTML + Voice, HTML
    - ECMAScript
    - Java Speech Grammar Format
    - Visual Paradigm (Modelamiento en UML)
    - Opera