# XMMVR: Especificación y Arquitectura para el desarrollo de aplicaciones de Interacción Multimodal en Escenarios 3D

Héctor Olmedo Rodríguez, David Escudero Mancebo, Arturo González Escribano, César González Ferreras, Valentín Cardeñoso Payo
ECA-SIMM, Departamento de Informática.
Escuela Técnica Superior de Ingeniería Informática (E.T.S.I.I.)
Universidad de Valladolid
Campus Miguel Delibes s/n
47011 - VALLADOLID
holmedo@infor.uva.es

## Resumen

Con la definición de un lenguaje de marcas para modelar escenas, comportamiento e interacción en base a la metáfora de película cinematográfica interactiva, esperamos tener un marco común para desarrollar aplicaciones que permitan interacción multimodal en escenarios 3D. Por ello, reutilizando lenguajes de marcas ya definidos como estándares para describir gráficos e interacción gráfica y vocal, hemos definido las bases de una arquitectura que nos permita integrar los componentes de este tipo de aplicaciones de interacción multimodal en entornos de Realidad Virtual.

**Palabras clave**: Realidad virtual, comportamiento, interacción vocal, interacción gráfica, interacción persona-ordenador, multimodalidad, sistemas de diálogo, avatar

## 1. Introducción

Es un hecho que los sistemas de Realidad Virtual (RV) incrementan significativamente el potencial de Interacción Hombre Máquina [1]. Si consideramos que los Sistemas de Diálogo (SD) aportan un canal complementario al canal gráfico como es el canal vocal o sonoro [2], podríamos ver la integración de ambos campos de investigación como una evolución natural de ambas tecnologías. Sin embargo no ha sido apenas explotada en sistemas comerciales, y aunque existen prototipos [3], se trata de un ámbito de trabajo por descubrir. La principal razón por la cual no existen apenas soluciones integradoras de RV y SD, es la juventud de estas áreas de trabajo, donde la mayoría de los esfuerzos se han centrado en mejorar de forma separada ambos campos, y no en estudiar las necesidades de interdependencia que se derivan de una propuesta integradora. Aquí se presenta una propuesta que combina RV-SD planteando una plataforma para desarrollo de aplicaciones basadas en mundos virtuales 3D que permita una interacción multimodal dirigida por diálogos.

Los ámbitos de la RV y de los SD se caracterizan por una relativa disponibilidad de prototipos realizados en laboratorios de investigación y de algún sistema comercial que, por lo general, han descuidado la necesidad de ajustarse a algún estándar de desarrollo o de especificación. El estándar en SD es VoiceXML [4] mientras que en RV hay un estándar de definición de escenas X3D [5] evolucionado de VRML [6]. La disponibilidad de estos estándares ha supuesto un marco de referencia para que los desarrolladores adapten sus sistemas, con las consiguientes aportaciones en cuanto a facilidad de uso en lo que se refiere a la definición de escenarios 3D y diálogos y a la portabilidad de módulos reutilizables. Aquí se presenta un marco de referencia que pretende ser un lenguaje de especificación de mundos 3D con integración de diálogos. La solución que aportamos respeta los estándares disponibles para RV y SD y sirve de vínculo entre ambos mundos, dando una coherencia argumental a la definición de escenas 3D con interacción hablada.

A lo largo de este artículo introduciremos en primer lugar los lenguajes de marcado para especificación de escena e interacción gráfica y

vocal para seguidamente describir el lenguaje XMMVR definido. A continuación definiremos la arquitectura necesaria para poder implementar este tipo de aplicaciones y las bases en las que hemos centrado su diseño, así como los elementos utilizados. Tras hacer una comparativa con otras propuestas actuales, esbozaremos una pequeña aplicación ejemplo para finalizar con las conclusiones y el trabajo a realizar en el futuro.

## 2. Interacción multimodal 3D y lenguajes de marcado

Añadir interacción vocal a los entornos virtuales con interacción gráfica aporta beneficios claros. Gracias a ello podemos emitir comandos manteniendo la libertad de manos y ojos. Además, los usuarios pueden referirse a objetos que no están presentes en la vista actual del mundo virtual, lo que hace que las acciones sean rápidas y su efecto inmediato. Pero existe una dificultad para la aproximación general a fusión multimodal que hace necesaria la definición de una arquitectura reutilizable para construir nuevos sistemas multimodales.

Las tres componentes de la interacción multimodal para entornos 3D son: la especificación tridimensional que básicamente consiste en modelar objetos del entorno virtual que pueden ser estáticos y/o dinámicos; la interacción gráfica (GUI) basada en teclado y ratón como la conocemos hasta ahora y que siempre gira en torno al modelo de eventos y en base a espacios de acción o action spaces [7] que son aproximaciones metafóricas para estructurar los interfaces de usuario tridimensionales (dos fundamentales, cinco estructurales y cinco navegacionales); y por último, la interacción vocal (VUI) en la que son posibles cuatro metáforas de interacción: proxy o delegado, divinidad, telekinesis o agente interfaz [8]. Elegir la metáfora de interacción vocal adecuada a nuestro mundo es tarea difícil y más especificar un lenguaje que englobe ésta dentro del marco definido.

Existen numerosos lenguajes de marcas para especificar interacción vocal, escenas y comportamiento por separado pero también hay otros lenguajes de marcado que denominaremos híbridos. Ejemplos de lenguajes de marcado para especificar interacción vocal son VoiceXML [4], SALT [9] y X+V [10]. Entre los lenguajes de marcado para especificar escenas tenemos VRML [6] y X3D [5]. Las limitaciones de estos dos lenguajes para especificar el comportamiento de los elementos integrantes de la escena ha llevado a definir lenguajes de marcado para especificar comportamiento, como por ejemplo Behavior3D [11] o VHML [12]. Entre los lenguajes de marcas híbridos podemos citar MIML [13] que permite integrar información de discurso, gesto y el contexto de una aplicación dada usando una aproximación de parseo o procesamiento sintáctico/semántico y MPML-VR que es una extensión de MPML (Multimodal Presentation Markup Language) un lenguaje de marcas diseñado para presentaciones multimodales que usa VRML 2.0 para poder presentar espacios tridimensionales a través de un agente antropomórfico o avatar de aspecto humano [14]. Además de los lenguajes de marcado vistos existen otros que también buscan definir especificaciones para aplicaciones concretas [15]. Siguiendo esta corriente y con el objetivo de definir un lenguaje de marcas para la especificación de mundos virtuales con interacción multimodal, llegamos a proponer el lenguaje XMMVR que describiremos en el siguiente apartado.

## 3. El lenguaje XMMVR

El eXtensible markup language for MultiModal interaction with Virtual Reality worlds o XMMVR es una propuesta de definición de un lenguaje de marcas para definir escena, comportamiento e interacción en el que consideraremos cada mundo o película interactiva como un elemento *"xmmvr"* basándonos en la metáfora de película cinematográfica. Podríamos decir que es un lenguaje de marcas híbrido porque la idea es utilizar otros lenguajes tales como VoiceXML o X+V para interacción vocal y X3D o VRML para descripción de escena que quedarían embebidos en éste.

De esta manera, el procesamiento de los ficheros XML válidos para el DTD de XMMVR permitirá enlazar con los programas y ficheros necesarios para hacer funcionar el mundo especificado. Nuestro sistema XMMVR va a ser dirigido por eventos, por ello habrá que definir una mínima lista de eventos y no va a existir línea de tiempos. Un elemento *"xmmvr"* está formado

principalmente por el reparto de actores *"cast"* y la secuencia de escenas *"sequence"* que marcan el transcurrir del mundo. Además nos reservamos un elemento *"context"* para uso futuro.

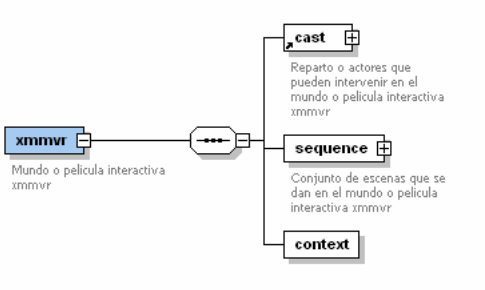


Figura 1. Elemento xmmvr

El elemento *"cast"* o reparto será el conjunto de actores que intervendrán en el mundo o película *"xmmvr"* es decir, cada uno de los elementos que tienen una apariencia gráfica especificada por un fichero VRML y un comportamiento que permite una interacción con el usuario. Al usuario lo consideraremos como un espectador sin presencia en el mundo pero que interactúa con los actores de éste, por tanto estamos utilizando la metáfora del proxy o delegado de la que hablamos anteriormente para especificar la interacción vocal y puesto que nos basamos en la metáfora de la película cinematográfica interactiva, ésta sería una evolución de una metáfora fundamental de interacción gráfica: la metáfora de teatro.

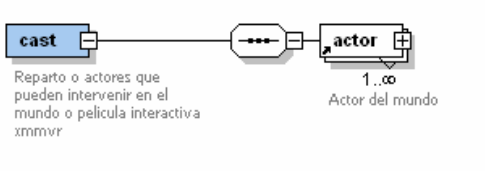


Figura 2. Elemento cast

Diremos que un *"actor"* es todo elemento que puede formar parte del mundo definido y que tendrá comportamientos propios que especificaremos con la etiqueta *"behavior"*.

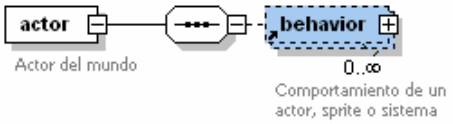


Figura 3. Elemento actor

Cada comportamiento *"behavior"* se definirá como una pareja de *evento* y *lista de acciones* que puede tener cada actor ante una determinada *condición*.

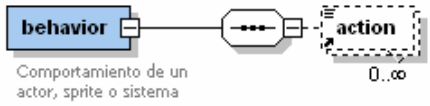


Figura 4. Elemento behavior

Un evento puede ser provocado por el usuario debido a una interacción gráfica *"GUI"* o a una interacción vocal *"VUI"*. Asimismo existen eventos del sistema que sirven para definir la interacción con otro actor del mundo *"ACT"* o de interacción con el mundo o sistema *"SYS"*. La lista de acciones serán una o varias acciones que se generan ante un evento y pueden ser también de carácter gráfico *"GUI"*, vocal *"VUI"*, de interacción con otro actor del mundo *"ACT"* o de interacción con el mundo o sistema *"SYS"*.
Además de lo anterior, tenemos que especificar la secuencia de escenas *"sequence"* en la que tendremos una o más escenas que se presentarán por defecto en el orden en el que se escribieron.

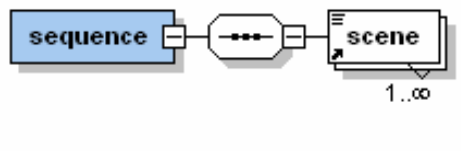


Figura 5. Elemento sequence

En todo mundo *"xmmvr"* consideraremos que ocurre al menos una escena *"scene"* y para que haya interacción entre el usuario y esa escena del mundo, deberá existir al menos un *"actor"* que habite el mundo definido.
Con todas estas premisas hemos definido un DTD [16] y podremos desarrollar aplicaciones para interacción multimodal con mundos virtuales en base a archivos XML válidos para dicho DTD que a su vez serán la especificación de las escenas, el comportamiento y la interacción en dichos mundos virtuales.

## 4. La plataforma XMMVR

El objetivo general es disponer de un marco y un lenguaje para modelar aplicaciones de interacción hombre-máquina multimodales (interacción gráfica e interacción vocal) en entornos 3D.
La construcción de una aplicación concreta utilizando nuestro marco de trabajo consiste en especificar un mundo virtual, las secuencias de diálogo, las acciones que se generan y su relación con los elementos del mundo. Las secuencias de diálogo se especifican empleando VoiceXML y los mundos virtuales utilizando VRML. Para describir el comportamiento del mundo debemos especificar la estructura de componentes que forman el gestor de mundo y la correspondencia entre los diálogos con el usuario y las acciones de alto nivel incluídas en el gestor de mundo. Todas estas especificaciones se expresarán en un documento estructurado mediante XML conforme al DTD de XMMVR.
Esto nos permitirá desarrollar una aplicación "embebida" en un navegador web que permita a un usuario controlar un mundo virtual desarrollado en VRML a través de la voz. Para ello usamos el micrófono del ordenador donde se ejecuta la aplicación que enviará las órdenes a un gestor de diálogo basado en VoiceXML. A la vez podremos interactuar con dicho mundo virtual a través del teclado/ratón del ordenador donde se está ejecutando. La arquitectura a implementar ha de basarse en los modelos definidos para las arquitecturas VRML-EAI y VoiceXML integrándose con el gestor de diálogo del grupo ECA-SIMM, ATLAS de IBERVOX [17].

### 4.1. Arquitectura XMMVR

Para implementar una arquitectura que integre la plataforma descrita basada en un applet sobre un navegador Internet Explorer que además permita a nuestro navegador VRML, CORTONA [22] mostrar el estado del mundo sobre el que interactuamos, hemos desarrollado varios paquetes Java basados en el API EAI [24] pero integrando desarrollos anteriores realizados por el grupo ECA-SIMM y que con la ayuda de un servlet sobre un servidor Apache Tomcat para realimentar el navegador vocal de nuestro sistema de diálogo, nos permite tener una arquitectura que soportará las aplicaciones definidas sobre un PC normal con el único requisito hardware de tener activo un micrófono y unos altavoces conectados a su tarjeta de sonido.

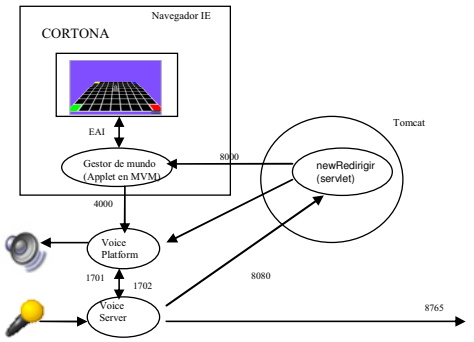


Figura 6. Esquema de integración del marco de trabajo con la plataforma vocal

Podríamos decir que nuestra arquitectura aglutina varias tecnologías basadas en XML: VoiceXML, VRML/X3D y el propio XMMVR de manera que podremos reutilizar documentos de cada una de estas tecnologías, obteniendo de esta manera un ahorro en tiempo de desarrollo de aplicaciones de este tipo.

### 4.2. Control de flujo

Un agente inteligente que tenga que desarrollar un objetivo de alto nivel en un escenario 3D dado, tendrá que resolver múltiples acciones elementales cuya ejecución va a depender de la semántica de la tarea concreta y del estado del mundo. Para ello, a lo largo de diferentes proyectos del grupo

ECA-SIMM, se han definido elementos que van a permitir desarrollar la plataforma definida. Ésta se encargará del control de flujo de eventos y acciones a realizar en nuestras aplicaciones. Para ello especificaremos unos comportamientos definidos en los correspondientes ficheros XML válidos para el DTD de XMMVR. Pero antes haremos un inciso para definir el comportamiento en mundos virtuales.

El comportamiento simple es aquel en el cual los cambios en el estado de los objetos dependen exclusivamente de eventos internos generados por nodos descritos dentro del programa VRML, sin la intervención de ningún lenguaje ajeno a VRML. Se puede decir que éste tipo de comportamiento es propio de VRML.

La propagación de los eventos se realiza enrutando o dirigiendo los eventos de un nodo a otro(s). De esta forma un solo evento puede generar lo que se conoce como "cascada de eventos", la cual va generando respuestas o cambios en los nodos a los que se va dirigiendo.

Dado que el comportamiento simple está restringido a dirigir datos de un nodo a otro, mediante instrucciones dadas sólo en VRML, se tienen limitaciones como las siguientes: no es posible mediante este mecanismo determinar si una puerta está cerrada o abierta, o mover un objeto a lo largo de una trayectoria definida por una ecuación matemática. Para llevar a cabo éstas u otras operaciones más complejas, se debe utilizar un lenguaje de programación de propósito general.

Se entiende por comportamiento complejo en un mundo VRML, aquel en el cual los cambios en el estado de los objetos dependen de un programa escrito en un lenguaje ajeno a VRML [18].

Los cambios dinámicos en la escena pueden ser estimulados por las acciones programadas en un script, paso de mensajes, comandos del usuario o protocolos de comportamiento. Esto brinda la posibilidad de interacción con otros lenguajes (específicamente los lenguajes Java y Javascript). Para ello se emplea el nodo Script o una interfaz como el EAI. De esta manera es posible asignar comportamientos más elaborados como se describe en lo sucesivo.

### 4.3. Visión global de capas

La solución planteada para definir nuestra plataforma de control de flujo pasa por dividir nuestro sistema en varias capas superpuestas donde cada una da un servicio a la capa inmediatamente superior. Repasaremos las capas que conforman nuestro gestor y las situaremos entre dos capas: una superior y otra inferior para poder encuadrar nuestro gestor de acciones en un contexto y que sea más fácil de comprender. Seguiremos un orden de arriba hacia abajo en el nivel de abstracción:

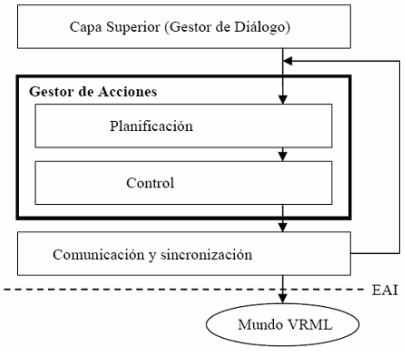


Figura 7. Visión global de capas del gestor del mundo

- Capa superior: Esta capa se correspondería con un gestor de diálogo que generaría eventos procedentes de una interacción vocal con el usuario y los proporcionaría como entrada para la capa inmediatamente inferior, que es la entrada a nuestro gestor de acciones (vocal y gráfica).
- Gestor de acciones: Se identifica con un subsistema que recibe eventos, los traduce en series de acciones y planifica su ejecución. Más tarde procesa las acciones interactuando con el mundo virtual a través de la interfaz suministrada por el Títere que detallaremos a continuación. Esto ha de hacerlo sin depender lo más mínimo de las características del mundo virtual que se vaya a controlar. A su vez, se subdivide en dos capas:

  Planificación: Es la capa que provee un mecanismo de acceso unificado para la entrada de eventos de diferentes fuentes. Una vez dentro, estos eventos se traducen en unas acciones complejas y cada una de estas se subdivide en una serie de acciones simples. Dichas series de acciones pueden ser ejecutadas de forma paralela

pero las acciones simples procedentes de una misma acción compleja se ejecutan de forma secuencial. Además esta capa implementa una arquitectura para el tratamiento de "anti-acciones" que son mandatos que anulan acciones simples pendientes de procesarse.

Control: La entrada de esta capa son las acciones simples procedentes del nivel superior. Aquí se ejecutan estas acciones teniendo en cuenta el estado del mundo. Llamamos estado del mundo al conjunto formado por los estados de todos los elementos dinámicos que componen el mundo virtual. Tras la ejecución de una acción simple el estado del mundo se actualizará con los nuevos estados

Comunicación y sincronización [19]: Base para ejecutar las acciones sobre el mundo virtual con un nivel superior de abstracción. Esto nos evita tener que conocer la interfaz EAI mediante la cual Java se comunica con un mundo en VRML y nos provee de un mecanismo de sincronización para la interacción con los objetos del mundo. Esta capa, fue redefinida para admitir la posibilidad de que un elemento virtual genere eventos, al pinchar con el ratón sobre su representación visual en el mundo (títere clickable) [20]. En ésta podemos también definir dos subniveles:

Títeres: Una capa de mayor abstracción, trabaja con la capa inferior a través del EAI para ofrecer una interfaz un poco más compleja y que permita realizar acciones que aunque simples, están desacopladas del VRML en lo que a las capas superiores se refiere.

Nodos VRML: La capa inferior que se ocupa únicamente de mantener la representación gráfica al usuario. Presenta su interfaz a la capa superior a través del EAI.

Se considera que un Títere se compone de dos naturalezas separadas: Por una parte es un objeto tridimensional desarrollado en el lenguaje de modelado VRML. Este objeto tan solo es una imagen sin ningún tipo de comportamiento o capacidad de realizar acciones. Por hacer un símil sería como la marioneta de un títere. Por otro lado es una clase escrita en Java a la que está asociado el objeto del mundo virtual y que es la encargada de proporcionar funcionalidad y dotar de capacidad de realizar acciones a dicho objeto. Continuando con el símil podríamos decir que esta clase sería los hilos que manejan la marioneta.

Para poder ceñirnos a esta filosofía debemos definir los nodos VRML de nuestras aplicaciones en base a un formato definido es decir, que no basta conocer la especificación VRML, debemos ceñirnos a las normas fijadas para la integración con XMMVR.

### 4.4. Manejo de eventos

A nuestro subsistema entran eventos. Estos eventos se traducen en las acciones que llevan asociadas según el fichero XML válido para el DTD de XMMVR. Estas acciones, si son de alto nivel, se descomponen en función de los otros dos tipos de acción menos abstractos. Una vez que los eventos son traducidos en sucesiones de acciones menos complejas, se ejecutan estas acciones sobre los títeres pertinentes teniendo en cuenta los estados actuales de los títeres implicados y modificando su estado almacenado posteriormente. Si el usuario pincha en la representación visual sobre uno de los títeres, este producirá su evento asociado y lo realimentará en el subsistema.

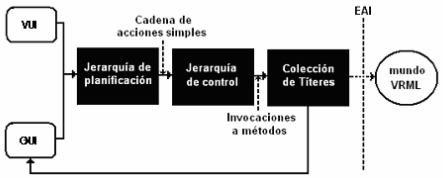


Figura 8. Vista global del subsistema

Partiendo de una visión tan clara del caso que nos ocupa, tendremos una gran parte del camino hecho a la hora de decidir la arquitectura software que vamos a dar al subsistema. Además de esta forma tenemos ya explicado el problema de una forma natural, legible para alguien no muy familiarizado con los estándares de análisis y diseño en la ingeniería del software.

Desde un primer momento se vio la jerarquía de planificación de los eventos como una serie de problemas de productores/consumidores. Si se hubiera elegido una estructura de clases estáticas la ejecución de acciones simultáneas sobre el mundo se hubiera visto dificultada en exceso. El subsistema quedaría bloqueado a la espera del procesamiento de un evento y no hubiera podido satisfacer la demanda de ejecución en tiempo próximo al real. Además, no hubiera sido posible el procesamiento de "anti-acciones" es decir, eventos excepcionales que pueden inhibir la

ejecución de acciones simples que están a la espera de ejecutarse.
Las hebras (hilos) y las colas son los componentes básicos para resolver problemas del tipo productores/consumidores pero esto siempre implica que deben cumplirse unas normas para evitar el bloqueo y poder acceder a los recursos críticos de forma segura. Para ello, en vez de utilizar cualquier estructura predefinida en Java, se ha implementado una clase ColaEventos que además de estar diseñada para satisfacer nuestras necesidades concretas, está provista de los mecanismos de sincronización para asegurar un correcto funcionamiento. Esto se ha conseguido implementando una clase Cola que provee los mecanismos de inserción y eliminación de elementos sobre un array de elementos.
El esquema de esta capa de planificación, visto como una serie de problemas de productores consumidores es como se muestra en la figura siguiente.

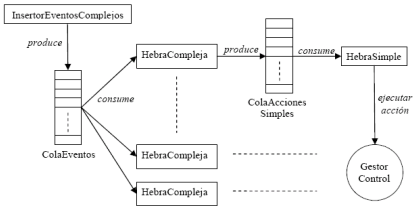


Figura 9. Planificación como Productores/Consumidores

## 5. Otros trabajos relacionados

El grupo MMI del W3C ha marcado las líneas a seguir por la arquitectura de una aplicación de interacción multimodal en la *Multimodal Architecture and Interfaces* [21]. Esta arquitectura se basa en un *Runtime Framework*, que provee de la estructura básica y controla la comunicación entre los demás elementos o *Constituents*. El *Delivery Context Component*, que es un subcomponente del Runtime Framework, provee información acerca de las capacidades de la plataforma. El *Interaction Manager* que es el siguiente subcomponente del Runtime Framework, coordina las diferentes modalidades, sería el Controlador del paradigma MVC y en la arquitectura XMMVR presentada se asemejaría con el gestor del mundo. El *Data Component* es el último subcomponente del Runtime Framework y es el proveedor del modelo de datos común, representaría el Modelo en el paradigma MVC y podríamos equipararlo al XML válido para el DTD de XMMVR en nuestra propuesta. Los *Modality Components*, aportan capacidades de interacción específicas, serían las Vistas en el paradigma MVC y los ficheros VRML y VoiceXML para la especificación de interacción gráfica y de escena y para la especificación de interacción vocal respectivamente en nuestra propuesta. Asímismo, podremos asemejar el API EAI y el servlet de los que hablamos al presentar nuestra arquitectura con los Modality Component API propuestos en la Multimodal Architecture and Interfaces.

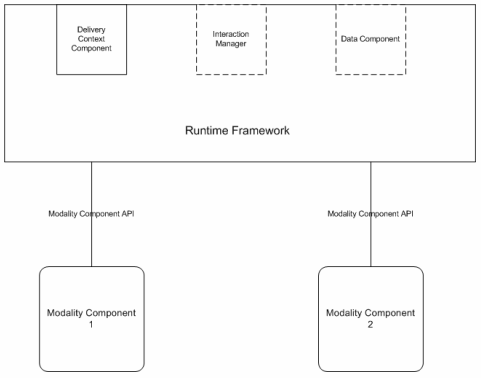


Figura 10. Multimodal Architecture and Interfaces

Además de este modelo arquitectónico, el grupo de interacción multimodal del W3C está desarrollando otra serie de estándares como por ejemplo un lenguaje XML para transportar los datos de entrada de las distintas modalidades al gestor de interacciones, EMMA o Extensible MultiModal Annotation Language [27]. Es una especificación para establecer cómo los objetos de entrada o salida se conectan al gestor de interacciones, un mecanismo de unificación o discriminación entre las distintas modalidades de entrada para por ejemplo resolver casos de entradas contradictorias, o selección entre valores de distintas modalidades en base a determinadas reglas o prioridades pero que a diferencia de nuestro lenguaje XMMVR no especifica comportamiento.
XMMVR tiene similitudes con proyectos como MIML y MPML que hemos descrito anteriormente. Este último es un lenguaje de

marcas diseñado para presentaciones multimodales que originalmente utilizaba un agente para animar presentaciones de tipo Powerpoint. La necesidad de dar más realismo a las presentaciones hizo plantearse la posibilidad de presentar espacios tridimensionales en lugar de simples fotografías en dos dimensiones, así se evolucionó a MPML-VR que es una extensión de MPML que usa VRML 2.0 para poder presentar espacios tridimensionales a través de un agente antropomórfico o avatar de aspecto humano. Se basa en el control de los estados de una escena es decir, controlar las transiciones de los estados de una escena basándose en elementos "evento" tales como un clic de ratón o un comando de voz o "speech input". Es en este control de eventos en lo que se asemeja a nuestra propuesta pero en su objetivo concreto se distancia de ella. NICE [28] es un proyecto para el desarrollo de juegos interactivos donde se presenta un escenario con personajes con los que el usuario puede interactuar para conseguir el éxito en el juego. Cada personaje tiene un domain model que corresponde a su visión del mundo, ontología que puede describirse de forma arborescente y que se puede representar en formato XML. El "mundo" se define por escenas que tienen unas fases en base a unos "dialogue events" o eventos de diálogo que generan clases java. El proyecto CONTIGRA [29] ha definido Behavior3D del que antes hablamos y que es un XML Schema automáticamente generado que integra todos los comportamientos disponibles para widgets 3D dando un repertorio de definiciones de comportamiento y que ha sido parte inspiradora de nuestra propuesta aunque sus objetivos se reducen a definir comportamiento de widgets 3D.

## 6. Conclusiones y trabajo futuro

Con la propuesta de lenguaje XMMVR presentada [30], queremos demostrar que es necesario definir un meta-guión para especificar cualquier mundo virtual que permita interacción multimodal y que éste aporta modularidad obteniendo con ello claridad y aumentando las posibilidades de reutilización y estandarización. Con el fin de comprobar la efectividad del lenguaje propuesto, hemos implementado la arquitectura descrita con una pequeña aplicación de ejemplo. En ésta sólo hemos definido un actor y un escenario por lo que habría que hacer más complejo el número de actores y escenarios en futuros desarrollos. Por otro lado, esta propuesta sólo considera la metáfora del proxy o delegado para la interacción vocal por lo que queda pendiente su extensión para poder dar solución a cada una de las metáforas de interacción vocal presentadas o a todas globalmente. Puesto que hemos utilizado el lenguaje VRML para especificar los elementos y la interacción gráfica basándonos en un navegador VRML ya obsoleto, Cortona, otra de nuestras tareas será redefinir nuestra arquitectura para poder trabajar con el lenguaje de especificación gráfica X3D utilizando un navegador adecuado lo que requerirá el desarrollo de la arquitectura basándonos en la interfaz SAI [23] en lugar de la EAI. Por último, queremos que nuestra plataforma se base totalmente en software de código abierto para que la comunidad de trabajo que desee adherirse a nuestra propuesta pueda ampliarse sin problemas, por ello estamos trabajando con otros navegadores como FreeWRL [26] o XJ3D [27] que pueden ejecutarse sobre plataformas Debian Linux entre otras. Respecto a los navegadores vocales, también nos planteamos la búsqueda de soluciones de código abierto pero esto a día de hoy será más difícil por lo que de momento seguiremos con la plataforma actual sin dejar de evaluar otras posibles candidatas. Tras la definición de una nueva plataforma basada en herramientas de código abierto, desarrollaremos una fase de evaluación con usuarios reales de las capacidades de interacción definidas para así corregir y mejorar nuestro diseño intentando dar respuesta a todas las limitaciones presentadas.

Nuestra propuesta a diferencia de las descritas, tiene la intención de ser una plataforma genérica para desarrollo de aplicaciones RV con interacción multimodal que permita experimentar con las metáforas de interacción propias de cada interacción, vocal y/o gráfica. De esta forma conseguiremos un laboratorio para "jugar" con las posibilidades que el uso de la multimodalidad nos permita. Esto hará posible desarrollar nuevas aplicaciones destinadas a ámbitos específicos donde la multimodalidad sería un gran apoyo para facilitar la interacción. Ejemplo de esto sería el desarrollo de juegos educativos basados en RV para usuarios con problemas de accesibilidad.

## Agradecimientos

Este trabajo ha sido financiado parcialmente por la Consejería de Educación de la Junta de Castilla y León, en el marco del proyecto VA053A05 ARACNOS: MARCOS PARA EL DESARROLLO DE INTERFACES WEB 3D QUE INCORPOREN INTERACCION VISUAL Y HABLADA CON EL USUARIO.

## Referencias

[1] W. R. Sherman, A. Craig. *Understanding Virtual Reality: Interface, Application, and Design*, The Morgan Kaufmann Series in Computer Graphics, 2002

[2] D.Dahl. *Practical Spoken Dialog Systems (Text, Speech and Language Technology)*, Springer, 2004

[3] C. González-Ferreras, A. González Escribano, D. Escudero Mancebo y V. Cardeñoso Payo. *Incorporación de interacción vocal en mundos virtuales usando VoiceXML*, CEIG, 2004

[4] VoiceXML Forum. *"Voice eXtensible Markup Language"*: http://www.voicexml.org

[5] Extensible 3D (X3D): http//www.web3d.org/x3d.html

[6] Jed Hartman, Josie Wernecke. *The VRML 2.0 Handbook*, Silicon Graphics, 1994

[7] R. Dachselt. *Action Spaces - A metaphorical concept to support navigation and interaction in 3D interfaces*; User Guidance in Virtual Environments, Workshop "Usability Centred Design and Evaluation of Virtual 3D Environments", 2000

[8] S. McGlashan, T. Axling. *Talking to Agents in Virtual Worlds*, UK VR-SIG Conf., 1996

[9] SALT Technical White Paper: http://www.saltforum.org/whitepapers/whitepapers.asp

[10] XHTML+Voice Profile 1.2: http://www.voicexml.org/specs/multimodal/x+v/12/spec.html

[11] R. Dachselt. *BEHAVIOR3D: An XML-Based Framework for 3D Graphics Behavior*; ACM Web3D, 2003

[12] VHML Standard: http://www.vhml.org

[13] Latoschik, M.E. *Designing transition networks for multimodal VR-interactions using a markup language*, ICMI, 2002

[14] Naoaki Okazaki y otros. *An Extension of the Multimodal Presentation Markup Language (MPML) to a Three-Dimensional VRML Space*, Wiley-Interscience 2005

[15] M.P Carretero y otros. *Animación Facial y Corporal de Avatares 3D a partir de la edición e interpretación de lenguajes de marcas*, CEIG, 2004

[16] DTD de XMMVR: http://www.infor.uva.es/~holmedo/xmmvr/xmmvr.dtd

[17] ATLAS de IBERVOX http://www.verbio.com

[18] Rolando Quintero Téllez. *Asignación de Comportamiento Complejo a Mundos Virtuales VRML Utilizando C++*, http://www.revista.unam.mx/vol.2/num2/art2/index.html

[19] Ignacio Fernandez-Divar Escacho, Alejandro Alcántara Zarzuela, *Herramientas para manipulación dinámica de mundos virtuales,* PFC-ECASIMM (9/2004)

[20] Samuel García Blanco, *Modelo e implementación de un gestor de acciones en mundos virtuales,* PFC-ECASIMM (9/2005)

[21] Multimodal Architecture and Interfaces http://www.w3.org/TR/2006/WD-mmi-arch-20061211/

[22] Cortona http://www.parallelgraphics.com/products/cortona/

[23] SAI, Scene Access Interface http://www.xj3d.org/tutorials/general_sai.html

[24] Andrew M Phelps, Rochester Institute of Technology, Department of Information Technology, *Introduction to the External Authoring Interface, EAI.* http://andysgi.rit.edu/andyworld10/gallery/archives/vrml/media/eaiclass.doc

[25] FreeWRL http://freewrl.sourceforge.net/

[26] XJ3D http://www.xj3d.org/

[27] EMMA http://www.w3.org/TR/emma/

[28] NICE http://www.niceproject.com/

[29] CONTIGRA http://www-mmt.inf.tu-dresden.de/Forschung/Projekte/CONTIGRA/index_en.xhtml

[30] XMMVR http://www.xmmvr.info