

VRU: Un método para validar requisitos y generar interfaces de usuario multiplataforma

Juan Sánchez¹, Jorge Belenguer¹, Pablo Belenguer², David Pascual²

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Valencia – España

¹{jsanchez, jorbefa}@dsic.upv.es;
²{pabbefa, dapasser}@inf.upv.es

Resumen. En el artículo se presenta un método de desarrollo de software que permite por una parte validar requisitos de usuario mediante prototipación automática y por otra generar interfaces de usuario. La generación de las interfaces utiliza como lenguaje intermedio el lenguaje de marcado UIML. Una interfaz en UIML puede ser traducida a diversas plataformas, lenguajes y sistemas operativos. El método es consistente con cualquier método de producción de software que utilice una capa de casos de uso y alguna variante de los diagramas de secuencia para mostrar las interacciones internas en el sistema. La ventaja de la propuesta radica en la utilización de una única definición de interfaz de usuario, que luego puede ser traducida y animada en diversos entornos. Las interfaces, con pequeñas modificaciones, pueden formar parte del producto software final.

1 Introducción

Con la proliferación de dispositivos móviles y con el incremento de las necesidades de acceso a la información desde diversos puntos (Internet, redes locales, redes locales inalámbricas, teléfonos móviles, etc.), una misma aplicación (o parte de su funcionalidad) puede ser invocada desde un ordenador de escritorio, un ordenador móvil, un asistente digital personal (PDA), o bien un teléfono con características WAP. Los usuarios exigen el acceso, desde cualquier ubicación, a la información y funcionalidad que poseen.

Los desarrolladores de las aplicaciones ahora también deben tener en cuenta qué funcionalidad ha de ejecutarse y visualizarse sobre los diversos dispositivos que pueden hacer uso de la aplicación. Si el proceso de construcción de la interfaz se aborda manualmente, lo habitual es que se empleen diversos lenguajes para programar la interfaz, siendo en el peor de los casos un lenguaje diferente por cada tipo de dispositivo.

Otra cuestión, que nosotros relacionaremos en este trabajo con las interfaces multiplataforma o multicanal, radica en el hecho de que la validación de los requisitos de usuario se realice de forma apropiada. Cuando se concibe un sistema informático,

una de las etapas cruciales, por el efecto que tiene en el resto del ciclo de desarrollo, reside en captar y mostrar de modo apropiado los requisitos de usuario. Esta actividad recibe el nombre de *Ingeniería de Requisitos*, habiendo sido reconocida como crucial ([15], [4]) dentro del proceso de desarrollo.

La etapa de captura de requisitos se puede abordar utilizando técnicas de escenarios. Un escenario se define como una descripción parcial del comportamiento de un sistema en una situación particular [5]. Un proceso de ingeniería de requisitos, basado en escenarios ([24]) posee dos tareas principales. En la primera, se genera una especificación, a partir de los escenarios, para describir el comportamiento del sistema. En la segunda, se validan éstos con el usuario, mediante simulación o prototipación. Ambas tareas son difíciles de manejar si no están asistidas por alguna herramienta automática o semiautomática.

Para abordar de un modo apropiado el proceso de captura y validación de requisitos, consideramos que es necesario definir una aproximación metodológica que permita derivar automáticamente interfaces a partir de requisitos de usuario. Por ello, estamos interesados en la validación de escenarios mediante la generación automática de prototipos de interfaces de usuario de la aplicación y la ejecución simbólica de los mismos.

En este artículo se presenta la propuesta metodológica y la herramienta que le da soporte, dentro del campo de la ingeniería de requisitos. La aproximación se basa en el Lenguaje Unificado de Modelado (UML [18]), extendido con la introducción de *Message Sequence Charts*¹ (MSCs [13]) enriquecidos con información referente a la interfaz de usuario. Dado que los MSCs se consideran una extensión de los diagramas de secuencia de UML, añadiendo los correspondientes estereotipos, la propuesta puede considerarse, desde el punto de vista notacional, consistente con UML.

El proceso es iterativo e incremental y permite derivar, como ya hemos comentado, los prototipos de la aplicación. Además, se genera una especificación formal del sistema que se representa mediante diagramas de transición entre estados. Estos diagramas describen el comportamiento de los objetos de interfaz y de control presentes en cada MSC. Una importante contribución del método es que genera de forma automática un modelo de navegación entre formularios, basado en las relaciones existentes, incluye y extiende, dentro del modelo de casos de uso. Esta característica de navegación permite la utilización de los prototipos generados dentro de entornos web.

En este artículo presentamos una extensión a nuestra propuesta inicial de validación de requisitos de usuario mediante prototipación automática ([21], [22], [23]) que utiliza una única definición de interfaz en el lenguaje de marcado UIML (User Interface Markup Language [12]), a partir de la cual mediante un proceso de traducción, se pueden crear y animar interfaces en diversos lenguajes y plataformas. En el trabajo haremos mayor hincapié en los nuevos aspectos incluidos en la propuesta: la utilización de UIML y la traducción de la interfaz al lenguaje destino.

El artículo está estructurado de la siguiente forma: la sección 2 contiene un resumen de la propuesta de generación con el proceso que guía los diferentes pasos.

¹ Utilizaremos indistintamente MSC ó diagrama de secuenciación de mensajes.

VRU: Un Método para Validar Requisitos y Generar Interfaces de Usuario 1

Table 1. Flujos de trabajo, actividades y entregables de VRU.

Etapa del ciclo de vida	Flujo de Trabajo	Actividades de VRU	Entregables
Análisis preliminar	Análisis Externo	Interiorizar Proyecto	Diccionario
			Requisitos Informales
		Crear perfil de usuarios	Mapa de roles de usuario
		Elaborar ámbito sistema	Modelo de dominio
			Diagrama de contexto
Análisis detallado	Análisis Funcional	Crear modelo funcional	Actores
			Casos de Uso
			Servicios Funcionales
	Análisis Interno	Modelización de objetos	Diagrama de clases
		Síntesis de casos de uso	Diagramas MSC
		Etiquetar MSCs	Diagramas MSC etiquetados
	Generación	Generar interfaz	Modelo de presentación
		Generar dinámica	Diagramas de transición
Validación	Animación	Ejecución interfaz	

La fase de análisis de requisitos, o análisis preliminar de requisitos, supone el arranque del proyecto. En esta etapa participan dos tipos de trabajadores o están presentes dos roles. El primero, el experto en el dominio del problema, representa un profesional con experiencia en las prácticas y en la terminología utilizada por la organización para la cual se desarrollará el sistema. Se encarga de proporcionar la entrada inicial al proceso de desarrollo: el documento inicial de requisitos. El segundo, el ingeniero de requisitos, representa un profesional cuyo campo de experiencia es la construcción de modelos de requisitos de acuerdo a la propuesta descrita en este artículo. Debe ser capaz de analizar el documento informal de requisitos y construir los modelos de análisis (explicados después).

Esta etapa comprende tres actividades secuenciales: *interiorización del proyecto*, creación del *perfil de los usuarios*, y *elaboración del ámbito del sistema*. En la actividad de *interiorización* del proyecto un experto del dominio del problema se encarga de elaborar un documento textual, que contiene por una parte un *diccionario* con la terminología del dominio, y por otra una descripción de alto nivel de la funcionalidad o servicios que ofertará el futuro sistema (requisitos informales). Los artefactos utilizados se engloban en el *documento inicial de requisitos*. Tanto el diccionario como los requisitos informales deben estar elaborados en un lenguaje comprensible tanto por los clientes/usuarios como por los desarrolladores. Esta documentación textual no forma parte de ninguno de los modelos propuestos por UML.

VRU: Un Método para Validar Requisitos y Generar Interfaces de Usuario 1

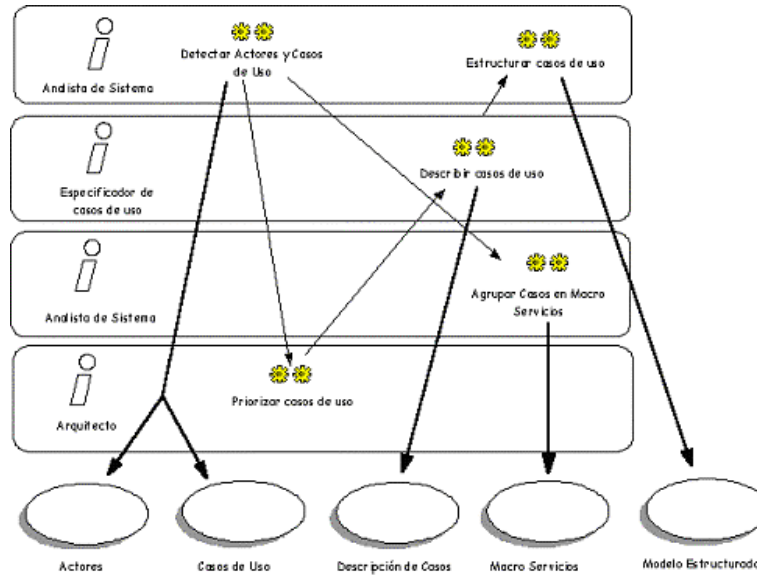


Fig. 1. Flujo de trabajo de VRU.

El concepto de macro servicio permite descomponer jerárquicamente el modelo de casos de uso. Para ello definimos un *servicio* del sistema, o *caso de uso elemental*, como la unidad básica de funcionalidad que puede invocar o utilizar un actor externo del sistema. Mientras que un *macro servicio* del sistema, o un *caso de uso de alto nivel*, contiene una agrupación jerárquica de servicios. Estos servicios pueden ser a su vez elementales o macro servicios. El modelo de casos de uso puede dividirse considerando los diversos tipos de usuarios y las necesidades de éstos con respecto al sistema. Podemos tener usuarios que utilicen frecuentemente las funciones ofertadas por el sistema; otros pueden utilizar el sistema ocasionalmente, mientras que una tercera clase de usuarios pueden encargarse de administrarlo. La *aproximación basada en grupos de usuarios* ([16]) utiliza los grupos de usuarios, representados en el modelo de actores, como criterio de agrupación y descomposición de los casos de uso. También puede dividirse utilizando el concepto de *situación de trabajo* ([17]). Ésta información será utilizada en el proceso de generación, para estructurar la interfaz de usuario de la aplicación.

El segundo flujo de trabajo, dentro de la fase análisis detallado, recibe el nombre de *análisis interno* y contiene las actividades llamadas modelización de objetos, síntesis de casos de uso y etiquetado de diagramas MSCs. En la Figura 2 se muestra el flujo de trabajo utilizando un diagrama de actividad de UML. Las dos actividades principales son la actividad de modelización de objetos y la actividad de síntesis de casos de uso. La primera crea un diagrama de clases del sistema, mientras que la segunda describe las interacciones internas asociadas a los casos de uso, utilizando diagramas MSCs. Estos diagramas se enriquecen con información referente a la interfaz de usuario.

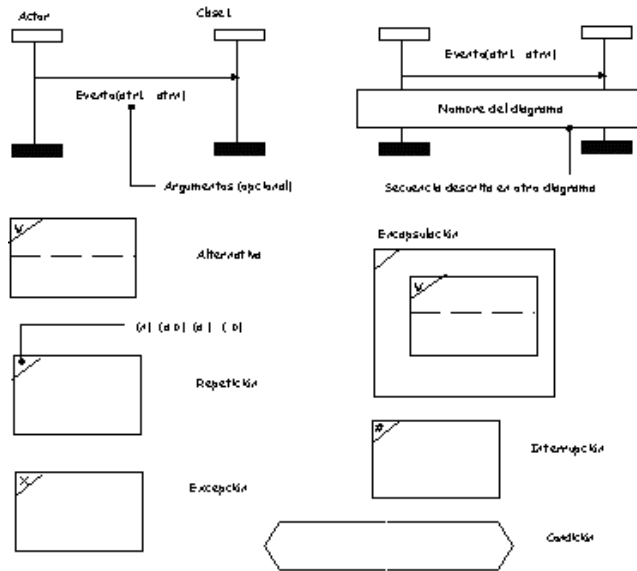


Fig. 3. Notación incluida en los diagramas MSCs.

3 Diagramas de secuenciación de mensajes (MSCs)

Los MSC se emplean ampliamente dentro del campo de las telecomunicaciones para describir el intercambio de información entre las instancias o procesos de un sistema. La notación incluida en nuestra herramienta aparece reflejada en la Figura 3. Las líneas verticales representan instancias de clases o actores del sistema, el paso de mensajes o el intercambio de información se refleja en la forma de líneas horizontales. Cada mensaje puede llevar asociado un conjunto de atributos. Éstos pueden ser tipos básicos (enteros, reales, etc.) o tipos clase. Existe una notación adicional que refleja: repetición de eventos, alternativa, excepciones, interrupciones y condiciones que pueden reflejar el estado en el que se encuentra el sistema. Los MSC se pueden descomponer jerárquicamente por niveles, de forma que en cada diagrama el analista puede utilizar el nivel de abstracción que desee. Los diagramas MSC se pueden enriquecer con información referente a la interfaz de usuario. Ésta información está formada por etiquetas textuales que aparecerán literalmente en la interfaz generada.

El tipo de los atributos de los mensajes determina durante el proceso de generación, el componente gráfico (*widget*) que le corresponde en una interfaz de usuario. Si el argumento es un tipo enumerado cuya talla es menor que 4, se le asocia un conjunto de botones de selección. Por el contrario, si la talla es mayor que 4 le corresponde una lista de selección. Los detalles de la equivalencia pueden obtenerse en [22].

clases de piezas y de propiedades de clases, y puede comprender desde una única plataforma hasta una familia completa plataformas. En el momento de escribir este artículo, la lista de glosarios estándar disponibles se podía encontrar en [27].

Un vocabulario UIML genérico ([8]) puede describir cualquier interfaz de usuario en cualquier plataforma de su familia. Sin embargo, un glosario genérico no es útil sin el correspondiente conjunto de transformaciones que permitan convertir la interfaz de usuario desde el plano abstracto a una implementación particular.

En la Figura 4 puede observarse un ejemplo de documento UIML en el que se ha hecho uso de un vocabulario genérico. En la ilustración también puede verse la interfaz de usuario que resultaría tras su proyección en Windows. El proceso de generación de interfaces de usuario multiplataforma mediante UIML, así como una descripción del marco de trabajo puede consultarse en [9].



Fig. 4. Ejemplo de documento UIML genérico.

Todo lo expuesto hasta el momento hace que UIML se profile como un lenguaje idóneo para crear interfaces de usuario dinámicas, multiplataforma, multimodales y multilingües, siempre y cuando se utilice un vocabulario lo suficientemente universal.

5 Uso de UIML en el proceso de generación de interfaces de usuario

El objetivo de este trabajo ha sido poder definir una interfaz que sea independiente del dispositivo, de la plataforma y del sistema operativo en la cual será visualizada, construida o proyectada. Para ello se ha hecho uso del lenguaje UIML, permitiendo así la definición genérica y abstracta de las interfaces. Éste lenguaje ha sido integrado dentro de nuestra propuesta, de modo que el proceso de generación de interfaces de usuario producirá, en un primer estadio, un conjunto de especificaciones UIML

Hay que indicar que todas las actividades contenidas en la figura anterior, desde la 1 hasta la 3, son actividades automáticas. El analista únicamente debe seleccionar el MSC que describe formalmente un caso de uso y el lenguaje destino para la generación.

5.1.1 Cohesión semántica entre controles

Un documento UIML contiene la jerarquía de elementos que participan en la interfaz, donde para cada uno de éstos elementos se puede describir una serie de propiedades básicas como puede ser el texto informativo, la visibilidad o la habilitación/inhabilitación de los mismos. Sin embargo, al margen de lo que es meramente la interacción con el usuario, hay cierta información semántica que escapa del ámbito de UIML.

La información de la que hablamos es la *cohesión semántica* entre controles, o dependencia semántica entre controles, que tiene como objetivo la agrupación de los controles en unidades funcionales. De este modo, catalogaremos una cohesión semántica de acuerdo a la siguiente categorización: *fuerte*, *intermedia* y *débil*.

Una *cohesión fuerte* indica la necesidad de que dos o más elementos de la interfaz queden fuertemente ligados, de forma que sea cual fuere la traducción de dichos elementos, no se les separe en distintas unidades. La cohesión fuerte debe emplearse para exigir que pares de elementos del tipo etiqueta-valor permanezcan juntos en una interfaz (v.g. una caja de texto puede requerir de una etiqueta informativa para informar al usuario sobre el valor solicitado).

Por el contrario, una *cohesión intermedia* indica que un conjunto de elementos presenta un contexto semántico común, como puede ser el caso de varios grupos de pares etiqueta-valor que solicitan nombre, DNI y dirección para crear una instancia de trabajador. En ese caso, el traductor intentará que dichos elementos pertenezcan a la misma unidad funcional, pero en caso de no ser posible por restricciones de espacio, se separarían en unidades distintas. Este tipo de cohesión se asume como propiedad intrínseca a los elementos de agrupación (como pueden ser los marcos o grupos de controles).

El tercer tipo de cohesión, la *cohesión débil*, se emplea para relacionar las diversas partes de un mismo caso de uso, de manera que todos los elementos incluidos en el mismo participan en fases de la misma acción. Sin embargo, este tipo de enlace no se explicita sino que se presupone en cada interfaz dado que la utilización de una interfaz conlleva la funcionalidad de la cohesión débil.

Nuestra cohesión semántica actúa como un contenedor de elementos de interfaz. Pero no sólo puede contener elementos de interfaz sino también otras cohesiones, siempre y cuando se cumplan las siguientes reglas de anidamiento: una cohesión fuerte sólo puede contener elementos de interfaz; una cohesión intermedia puede contener tanto elementos de interfaz como otras cohesiones, siempre y cuando éstas sean fuertes o intermedias; y una cohesión débil puede contener tanto elementos de interfaz como otras cohesiones, excepción hecha de cohesiones débiles.

Cada documento UIML tendrá asociado un fichero, en formato XML y con una gramática elaborada *ex profeso* para ello, que contendrá la cohesión semántica

alto nivel y en ellos estarán definidas todas las propiedades de cada uno de los *widgets* (desde la ubicación hasta el tamaño, incluyendo el color, etc.).

5.1.4 Etapa 3: Generación del código fuente del lenguaje de programación de alto nivel destino

En la última etapa de la generación, los documentos UIML específicos son enviados al módulo *render* correspondiente, quién se encargará de la traducción a código fuente del lenguaje de programación de alto nivel, pudiéndose compilar y ejecutar a continuación. Hay que indicar que a pesar de poder utilizar renders de terceras partes, en la herramienta se ha creado un vocabulario genérico propio (adaptado a los diagramas MSCs) y se han programado cada uno de ellos.

6 Un escenario de ejemplo: Gestión de Congresos

El caso de estudio seleccionado es una versión más general del problema conocido como “Conference Review System” distribuido en la conferencia OOPSLA de 1991. Una solución propuesta por J. Rumbaugh puede consultarse en la columna de análisis y diseño del *Journal of Object Oriented Programming* [20]. El propósito del sistema es dar soporte a los procesos de envío, evaluación y selección de artículos, vía Web para una conferencia o congreso. La interacción de los usuarios con el sistema se podrá llevar a cabo mediante un navegador web o bien mediante un dispositivo PDA conectado a un servidor central.

Por cuestiones de espacio únicamente trataremos parte de la funcionalidad asociada al actor autor. Los autores pueden registrarse en la conferencia para enviar un artículo, enviar un artículo, modificarlo, mandar la versión definitiva e inscribirse en la conferencia. La Figura 7 contiene, por una parte, los tres macro servicios que puede activar el autor y, por otra, los casos de uso incluidos en gestión de envíos.

La descomposición de macro servicios o la agrupación de casos de uso, permite obtener automáticamente una estructura de menú para cada actor del sistema (modelo de vistas). A cada macro servicio le corresponde un botón de activación o una entrada de menú en un formulario, en el lenguaje destino.

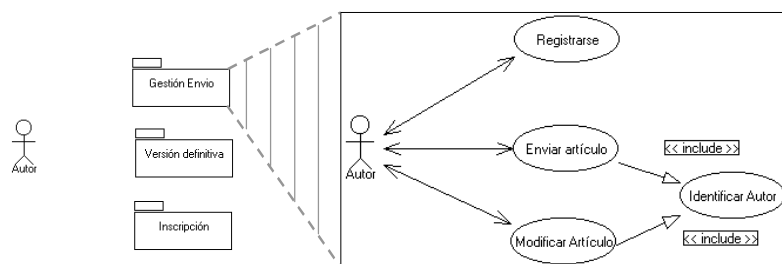


Fig. 7. Vista del actor sobre el modelo de Gestión de Congresos.

al actor primario del caso de uso (en este caso el autor), en cada MSC situamos un objeto de interfaz (I.U.) y un objeto de control (O.C), la cuarta línea vertical representa una instancia de Autor_Registrado. El diagrama contiene la etiqueta “Identificación Autor” que luego se utilizará en la interfaz generada. El MSC contiene los dos escenarios posibles incluidos en el caso de uso, representados por un rectángulo con la etiqueta “V”, que el autor esté registrado o que no esté. Para este caso de uso se genera automáticamente, a partir de la información contenida en el MSC, una definición de interfaz en UIML. Seleccionando como lenguaje destino HTML y eMbedded Visual Basic, el proceso de *renderización* genera los formularios de la Figura 10.

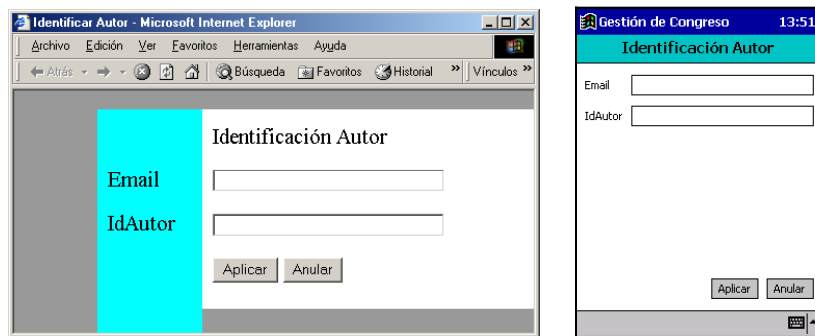


Fig. 10. Apariencia en HTML y eVB de la pantalla de identificación del autor.

El último componente del modelo de presentación, es el modelo de navegación, que habilita la navegación desde un formulario a otro de la aplicación. Para el macro servicio gestión de envíos la única navegación posible se da entre los casos de uso que utilizan a “identificación”. Esto se simula en tiempo de ejecución con un botón que permite pasar de un formulario a otro.

7 Conclusiones

En el artículo se ha presentado una extensión multiplataforma a un proceso de validación de requisitos de usuario, mediante la prototipación automática de interfaces de usuario. La utilización de un lenguaje declarativo como UIML para almacenar las definiciones de interfaz de usuario, permite que las mismas sean independientes de la plataforma destino. Además de crear un vocabulario genérico, que cubre aquellos aspectos propios de la información de interfaz que puede incluirse en un diagrama MSC, se han utilizado ficheros de configuración en XML para ubicar los controles sobre la plataforma destino. La utilización de los distintos tipos de cohesión entre los componentes de la interfaz permite una ubicación conveniente de los controles. El proceso de proyección se encuentra parametrizado por estos archivos. Cambiando los archivos de configuración se puede modificar la apariencia de las interfaces

VRU: Un M todo para Validar Requisitos y Generar Interfaces de Usuario 1

19. Angel Puerta, Jacob Eisenstein; "XIML: A Universal Language for User Interfaces", 2002 Conference on Intelligent User Interfaces – IUI 2002 (San Francisco, California, Enero 2002).
20. James Rumbaugh; "Onward to OOPSLA"; Journal of Object Oriented Programming, 5(4): 20-24, (Julio/Agosto 1992).
21. Juan Sánchez Díaz; "Validación de requisitos de usuario mediante prototipación y técnicas de transformación de modelos"; Tesis doctoral, Universidad Politécnica de Valencia, Departamento de Sistemas Informáticos y Computación (Noviembre 2002).
22. Juan Sánchez; Juan J. Fons; Óscar Pastor; "From user requirements to user interfaces: a methodological approach". CAISE 2001. (Interlaken, Suiza Junio 2001).
23. Juan Sánchez, Óscar Pastor, Jorge Belenguer; "Generación automática de prototipos de interfaz de usuario a partir de modelos de requisitos"; VI Jornadas de Ingeniería del Software y Bases de Datos (Almagro, España, Noviembre 2001).
24. Stéphane S. Somé, Rachida Dssouli, Jean Vaucher; "Toward an Automation of Requirements Engineering using Scenarios"; Journal of Computing and Information, vol. 2,1, (1996), pp 1110-1132.
25. "eXtensible Markup Language (XML) 1.0"; World Wide Web Consortium Recommendation (Agosto 1998), <http://www.w3.org/XML/>
26. "HTML 4.01 Specification"; World Wide Web Consortium Recommendation (Diciembre 1999), <http://www.w3.org/TR/html401/>
27. User Interface Markup Language (UIML) Web, <http://www.uiml.org>.
28. WAP Forum; "Wireless Markup Language (WML) version 2 Specification"; (Septiembre 2001), <http://www1.wapforum.org/tech/documents/WAP-238-WML-20010911-a.pdf>