Una Propuesta Basada en Model Driven Architecture para el Soporte de Rich Internet Applications

Iván López, Magalí González, Nathalie Aquino y Luca Cernuzzi

Universidad Católica "Nuestra Señora de la Asunción", Asunción, Paraguay
ivalop81@gmail.com
{mgonzalez, nathalie.aquino, lcernuzz}@uca.edu.py

Abstract. Las Rich Internet Applications (RIA) han surgido para permitir aplicaciones en la Web con características similares a las de escritorio. Este trabajo se enfoca en las características RIA relacionadas a la validación local de datos y a la inclusión de widgets interactivos de amplia utilización para extender Model Oriented Web Approach (MoWebA). MoWebA es una aproximación de desarrollo de aplicaciones Web fundamentada en los principios de Model Driven Architecture (MDA). En ese contexto, este trabajo extiende MoWebA para dar cobertura a las Rich Internet Applications (RIA), dando lugar a MoWebA4RIA. Se han extendido los metamodelos y perfiles de MoWebA para dar soporte a las características mencionadas. De manera correspondiente, se adaptaron las reglas de transformación y el compilador de modelo a texto para generar interfaces enriquecidas usando jQuery UI y jQuery Validation Plugin. Finalmente, se ha realizado una ilustración de la propuesta, a fin de demostrar las nuevas capacidades de MoWebA4RIA.

Keywords: Model Driven Architecture (MDA), Model Oriented Web Approach (MoWebA), Rich Internet Applications (RIA)

1 Introducción

El término Rich Internet Application (RIA) fue introducido en el 2002 por la empresa Macromedia, que en ese entonces abordaba limitaciones en cuanto a la riqueza de las interfaces, medios y contenidos de las aplicaciones Web. Las RIA son aplicaciones Web que exhiben widgets, comportamientos y características que están presentes en las aplicaciones de escritorio. Poseen una mayor capacidad de respuesta, son más seguras y presentan una interfaz más avanzada con respecto a las aplicaciones del modelo Web tradicional [2][4].

Las RIA tienen cuatro características principales [3]: i) es posible almacenar datos en el lado cliente, o bien distribuir el almacenamiento entre el cliente y el servidor; ii) es factible distribuir la ejecución de la lógica de negocios entre el cliente y el servidor, o bien llevar a cabo operaciones complejas, específicas de dominio, o de validación de datos directamente en el lado cliente; iii) las interfaces de usuario ofrecen una mayor riqueza ya que permiten manejar eventos en el lado cliente, ofertan una amplia variedad de widgets interactivos, siguen un paradigma de página única, y dan soporte

a contenido multimedia; y iv) se crean mecanismos para reducir al mínimo la transferencia de datos entre el cliente y el servidor, dando soporte a comunicaciones asíncronas entre ellos.

Debido a los avances propuestos por las RIA, muchas metodologías de desarrollo basadas en modelos y enfocadas en las aplicaciones Web tradicionales (como OOHDM [12], OOH [15], WebML [14], UWE [8][13] , OOWS [1], entre otras) han tenido que evolucionar, agregando nuevos modelos o extendiendo los existentes, para dar cobertura a las características de las RIA.

Los trabajos de Wright y Dietrich [5], y el de Preciado et al. [6] exponen la necesidad de metodologías sistemáticas para el desarrollo de las RIA y llevan a cabo estudios presentando las diversas metodologías Web existentes para ese fin. Toffetti [3] presenta un estudio más exhaustivo y reciente enfocado en hacer comparativas entre las diversas metodologías. Busch y Koch [9] presentan además un estado del arte de las aplicaciones RIA.

En el contexto de este trabajo, se ha realizado un nuevo análisis comparativo entre las metodologías de desarrollo Web basadas en modelos para las RIA. Las metodologías consideradas en el análisis derivan de la evolución de enfoques dirigidos por modelos que fueron concebidos originalmente para el diseño y desarrollo de aplicaciones Web tradicionales y son: OOHDM-RIA [12], OOH4RIA [15], WebML-RIA [14], Patrones RIA con UWE [13], UWE-R [8] y UWE+RUX [7]. Cada una de estas metodologías es comparada con respecto a las cuatro características principales de las RIA.

Tabla 1. Comparación de metodologías de desarrollo Web basadas en modelos en relación a las características de las RIA

Características RIA / Metodologías de desarrollo Web basadas en modelos		**OOHDM-RIA**	**OOH4RIA**	**WebML-RIA**	**Patrones RIA con UWE**	**UWE-R**	**UWE+RUX**
Almacenamiento en lado cliente		-	-	sí	-	-	-
Lógica de negocios en lado cliente	Operaciones complejas	-	-	sí	-	-	-
	Operaciones específicas de dominio	-	-	-	-	sí	-
	Validación de datos local	sí	sí	sí	-	-	sí
Interfaces enriquecidas	Manejo de eventos en lado cliente	-	-	sí	sí	sí	sí
	Widgets interactivos	sí	sí	-	sí	sí	sí
	Paradigma de página única	sí	sí	sí	sí	-	sí
	Contenido multimedia	-	sí	-	-	sí	sí
Comunicación cliente-servidor	Sincronización de datos	-	-	sí	-	sí	sí
	Actualizaciones parciales de página	sí	sí	sí	sí	sí	sí
	Push y pull	-	-	sí	-	sí	-

Muchas de las metodologías citadas han logrado una notable evolución incorporando los avances propuestos por las RIA. Sin embargo, como se puede observar en la Tabla 1, ninguna de las metodologías ofrece cobertura completa a todas las características de las RIA, de allí la necesidad de proponer extensiones a tales metodologías, o bien, de promover nuevas.

Recientemente se ha propuesto un nuevo enfoque, denominado Model Oriented Web Approach (MoWebA) [10] [11], para el desarrollo de aplicaciones Web, fundamentado en los principios propuestos por el Object Management Group (OMG - http://www.omg.org/). Este enfoque está basado en los estándares de Model Driven Architecture (MDA - http://www.omg.org/mda/) y ofrece un esquema de modelado en capas para la separación de conceptos.

En el marco de este trabajo, en lugar de extender alguna de las metodologías Web analizadas, se ha optado por extender MoWebA para que de soporte a las características RIA orientadas a datos en el lado cliente y widgets interactivos, en aplicaciones Web para PCs. Los motivos de elegir MoWebA son: i) está basado en estándares ampliamente aceptados, UML, mientras que entre las metodologías analizadas, solo OOH4RIA, Patrones RIA con UWE y UWE-R poseen un lenguaje de modelado cien por ciento basado en UML; ii) propone un enfoque de modelado orientado a la navegación dinámica centrada en la interacción del usuario y del contexto (y no del dominio), lo cual lo vuelve bastante atractivo para estos entornos y lo diferencia de las demás propuestas analizadas; iii) incorpora un modelo específico de arquitectura (ASM - Architectural Specific Model), no considerado por ninguna propuesta estudiada, que establece una clara separación entre los modelos independientes de la plataforma y los que incorporan detalles de la arquitectura; iv) finalmente, se tiene el interés de que MoWebA siga evolucionando e incorpore características RIA.

Para lograr el objetivo mencionado, los metamodelos y perfiles relacionados a la capa de presentación de MoWebA han sido mejorados y extendidos para dar soporte a la validación de datos en el lado cliente y a los widgtes interactivos. Además, se adaptaron las reglas de transformación del compilador de modelo a texto a fin de generar interfaces enriquecidas en código HTML y Javascript.

El resto del artículo se estructura como se explica a continuación. En la sección 2 se introduce el enfoque MoWebA. En la sección 3 se presentan las extensiones que se han realizado en los metamodelos de MoWebA para dar soporte a las RIA, así como, el proceso de transformación para generar las nuevas interfaces enriquecidas. En la sección 4 se presenta una ilustración de la propuesta. Finalmente, la sección 5 presenta un análisis de los resultados obtenidos y los posibles trabajos futuros.

2 La Aproximación MoWebA (Model Oriented Web Approach)

MoWebA [10] [11] adopta los principios de MDA y propone un enfoque que incluye un proceso de modelado y un proceso de transformación. En el proceso de modelado se especifican los diagramas necesarios para contemplar las fases CIM (Computational Independent Model), PIM (Platform Independent Model), ASM (Architectural Specific Model) y PSM (Platform Specific Model). Esta propuesta provee un modela-

do basado en la navegación como punto de referencia principal, en donde la navegación se encuentra orientada a funciones. Las fases del proceso de modelado se resumen a continuación.

- **Modelado del problema:** en esta fase se incluyen el CIM y el PIM. El CIM está orientado al modelado de los requisitos funcionales. El PIM está orientado al modelado del problema sin considerar aspectos de la arquitectura o plataforma. A partir de aquí es posible llevar a cabo transformaciones para obtener los modelos específicos de la arquitectura y de la plataforma de manera semi-automática.
- **Modelado de la solución:** en esta fase se incluyen el ASM y el PSM. Es en esta fase en la que todos los detalles de la arquitectura y plataforma destino se definen, permitiendo generar a partir de aquí, el código de la aplicación de manera automática. En MoWebA se plantea tener siempre el mismo PIM para una aplicación dada y a partir de este, se podrán generar diferentes ASM para diferentes arquitecturas destino. Finalmente, a partir de un ASM, se podrán generar diferentes PSM para diferentes plataformas destino. Como se puede ver, MoWebA separa aspectos arquitectónicos y de plataforma específica, y esto hace que la aproximación sea bastante prometedora para la implementación de las RIA, debido a que las RIA constituyen una arquitectura y existen numerosas plataformas destino para desplegarlas.
- **Código fuente:** incluye al ISM (Implementation Specific Model), que corresponde al código generado automáticamente y al código manual agregado, en caso de ser necesario, para obtener la aplicación final.

Cabe mencionar que todos los modelos de la aplicación pueden refinarse, dado que todas las fases son iterativas e incrementales.

La capa de presentación de MoWebA, para la cual ha sido definida la propuesta de extensión, tiene como fin principal facilitar la interacción de los usuarios con la aplicación y proveerles los elementos necesarios para que puedan realizar sus tareas (introducir datos, activar procesos, navegar, etc.) de la mejor manera posible. MoWebA propone dos diagramas para ella: el diagrama de contenido y el diagrama de estructura. El diagrama de contenido permite especificar los diferentes elementos que serán presentados en la interfaz. Se compone de un conjunto de páginas de presentación, las que, a su vez, se componen de uno o más elementos de interfaz. Los elementos de interfaz definen un conjunto de atributos clasificados de la siguiente manera: atributos estáticos, que son los atributos propios de la clase de presentación y sirven para representar información no asociada a la capa de datos, como por ejemplo, el titulo de la página web; y atributos de enlace, que representan la transición de un estado a otro, como un submit de un formulario.

El diagrama de estructura permite representar las diferentes regiones de una pantalla. Cada región puede estar compuesta de subregiones. Con el diagrama de estructura se pueden definir diferentes estructuras por aplicación.

Cabe destacar que actualmente MoWebA no soporta la arquitectura RIA. Más detalles de la capa de presentación pueden ser consultados en [10] [11].

3 Una propuesta de extensión a MoWebA: MoWebA4RIA

Las RIA cubren una gran variedad de características que abarcan varias capas de una aplicación (presentación, lógica de negocio, persistencia y comunicación cliente-servidor). En este trabajo, se ha decidido enfocar la extensión hacia las presentaciones enriquecidas considerando a los widgets y en la lógica de la aplicación del lado cliente, dejando como trabajo futuro las demás características. Para identificar los widgets a ser considerados, se ha realizado primeramente una revisión de aquellos más utilizados a partir de un conjunto de sitios web muy demandados (como Facebook, gmail, amazon, entre otros). El proceso de extensión de MoWebA para que de soporte a las RIA (presentación y lógica del lado cliente) ha consistido de los siguientes pasos. Primero, se han mejorado y extendido los metamodelos de la capa de presentación de MoWebA. Las principales extensiones se han realizado en el metamodelo de contenido, donde se han agregado nuevos elementos enriquecidos (widgets) y propiedades de validación local de campos de entrada. Segundo, las mejoras y extensiones de los metamodelos de contenido y estructura han sido reflejadas en los perfiles (UML profiles) correspondientes de MoWebA. Estos perfiles son los que definen la sintaxis concreta de MoWebA. Tercero, se definieron las reglas de transformación para las plataformas jQuery UI (https://jqueryui.com/) y jQuery Validation (http://jqueryvalidation.org/). Si bien existen diferentes tecnologías para el desarrollo e implementación de las RIA, las basadas en Javascript o librerías Ajax son las más utilizadas debido a que utilizan tecnologías de uso abierto y estandarizado como Javascript, HTML y CSS. Además, según el sitio de encuestas sobre tecnologías para la Web, W3Tech, jQuery es utilizado por más del 67,9% de todos los sitios Web a nivel mundial, siendo el de mayor popularidad [16]. Las transformaciones definidas en este trabajo son de modelo a texto (M2T) y fueron definidas en las plantillas de transformación de contenido y estructura, respectivamente, que han sido implementadas utilizando la herramienta Acceleo (https://eclipse.org/acceleo/), ampliamente conocida y relacionada con el proyecto Eclipse. Como resultado del proceso, MoWebA ha evolucionado a MoWebA4RIA. A continuación, se presentan más detalles del proceso descripto.

3.1 Metamodelos y Perfiles de Contenido y Estructura de MoWebA4RIA

La Fig. 1 presenta los metamodelos de contenido (Content) y de estructura (Layout) de MoWebA4RIA. Estos metamodelos fueron obtenidos luego de aplicar mejoras y extensiones a sus versiones originales correspondientes de MoWebA. Los elementos de ambos metamodelos fueron catalogados en diferentes colores. Aquellos que ya existían en los metamodelos originales de MoWebA aparecen en color salmón. Los que ya existían y han sufrido modificaciones, aparecen en color celeste. Y los nuevos, que no existían en los metamodelos orginales de MoWebA, aparecen en color verde.

En relación al metamodelo de contenido, un diagrama de presentación (PresentationDiagram) se compone de páginas (PresentationPage), las que a su vez se componen de elementos de interfaz (PDElements). Los elementos de interfaz pueden ser compuestos (CompositeUIElement) o simples (UIElement). Un elemento compuesto de

interfaz puede contener a otros elementos que pueden ser compuestos o simples. Un elemento de interfaz puede acceder al modelo de datos y tener condiciones (UICondition) del tipo “order by” o “group by”. Además, un elemento de interfaz contiene las propiedades horizontalOrder y verticalOrder para indicar una prioridad en el orden de despliegue en pantalla horizontal o vertical de un elemento simple o compuesto que se encuentra definido en una misma región que otro elemento en una misma página. Por su parte, los Panels permiten aglomerar varios elementos compuestos de interfaz. Además, cada panel forma parte de un elemento enriquecido de interfaz, ya sea un Accordion o un Tabs.

Los elementos simples de interfaz, ya sean tradicionales o enriquecidos, se clasifican en elementos de entrada, de salida, y de control. Los elementos de entrada (InputElement) se encargan de obtener una entrada desde la interfaz de usuario (TextInput, List, Password, Email, DatePicker, AutoSuggest); los de salida (OutputElement), de desplegar información en la interfaz (Text, HtmlText, Multimedia, ToolTip); y los de control (ControlElement), de obtener órdenes de navegación o de cambio de página (ExternalLink, Button, Anchor).

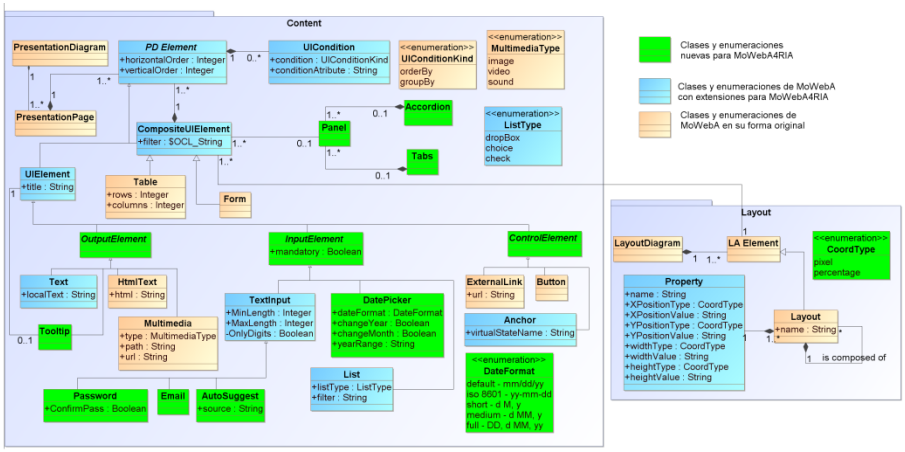

Fig. 1. Metamodelos de contenido (Content) y de estructura (Layout) de MoWebA4RIA

Luego de una explicación general del metamodelo de contenido, se presentan a continuación más de detalles de cada uno de sus nuevos elementos enriquecidos incorporados a la propuesta MoWebA: nuevos widgtes y opciones de validación local de datos.

- **AutoSuggest:** permite desplegar un listado de palabras sugeridas a partir de una fuente de datos definida por el atributo source (que puede ser introducida como listado, o bien a través de un archivo xml). También es posible obtener el listado de palabras desde el modelo de datos de MoWebA.
- **DatePicker**: contiene a los atributos dateFormat, changeYear, changeMonth y yearRange. El dateFormat corresponde a un tipo de dato enumerable que permite seleccionar formatos de fecha. El atributo changeYear es un valor booleano que

indica la ausencia o presencia de un rango de años desplegable en el DatePicker. El rango de años se especifica en el atributo yearRange. Por último, el atributo booleano changeMonth permite desplegar una lista con todos los meses del año.

- **ToolTip**: permite enriquecer con mensajes personalizados a cualquiera de los elementos de entrada, salida o control.
- **Live Validation**: corresponde a un conjunto de extensiones que permite llevar a cabo validaciones locales a elementos de interfaz del tipo TextInput, Email, Password y List. Para dar soporte a las validaciones se han agregado nuevos atributos en las clases correspondientes (por ejemplo, los atributos MinLength, MaxLength y OnlyDigits de la clase TextInput).
- **Accordion y Tabs**: permiten encapsular elementos de interfaz dentro de paneles colapsables para presentar información en una cantidad limitada de espacio.

En cuanto al metamodelo de estructura, un diagrama de estructura (LayoutDiagram) se compone de elementos de estructura (LAElement, Layout). A su vez, un elemento de estructura se puede componer de otros elementos de estructura. Además, cada elemento de estructura se asocia a varios elementos de propiedades (Property) que permiten definir valores para diversos atributos relacionados a las coordenadas de la posición de un elemento en la interfaz y a sus dimensiones.

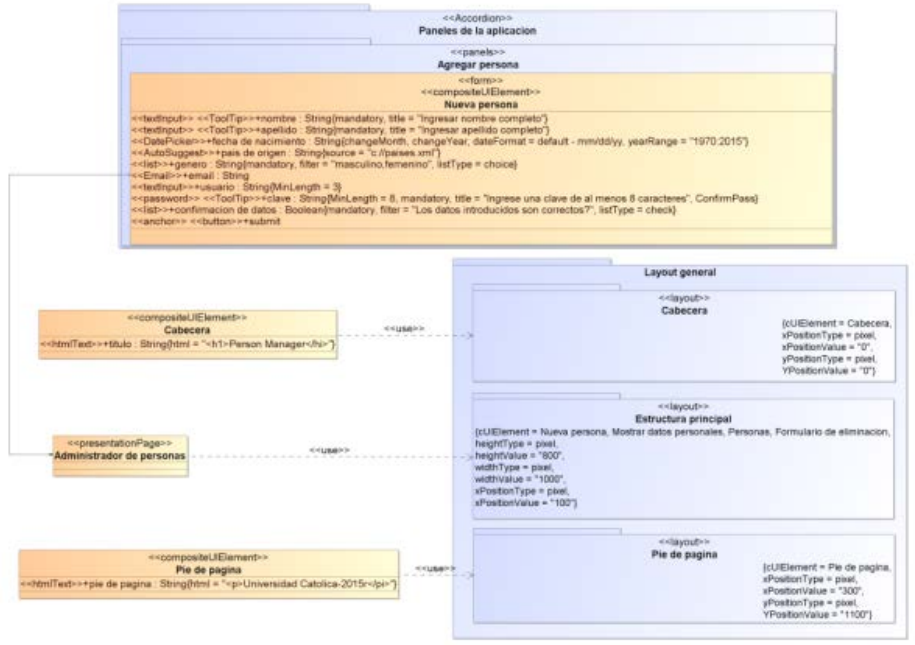

Fig. 2. Parte del PIM de capa de presentación modelado con MoWebA4RIA para una aplicación de administración de personas

A partir de los metamodelos de contenido y estructura se han derivado los perfiles UML correspondientes para MoWebA4RIA. Estos perfiles definen la sintaxis concreta de MoWebA4RIA. Con los metamodelos y perfiles definidos, es posible utilizar MoWebA4RIA para modelar los PIM de capa de presentación de una aplicación dada. La Fig. 2 presenta un ejemplo de PIM de capa de presentación modelado con MoWebA4RIA para una aplicación sencilla que permite administrar personas (listarlas, agregarlas o eliminarlas de un sistema). El ejemplo incluye varios elementos que forman parte de la extensión que se ha realizado en este trabajo, como el Accordion, paneles, etc.

3.2 Transformación para la Generación de Interfaces Enriquecidas

El proceso que se debe realizar para generar aplicaciones con interfaces enriquecidas con MoWebA4RIA se representa en la Fig. 3.

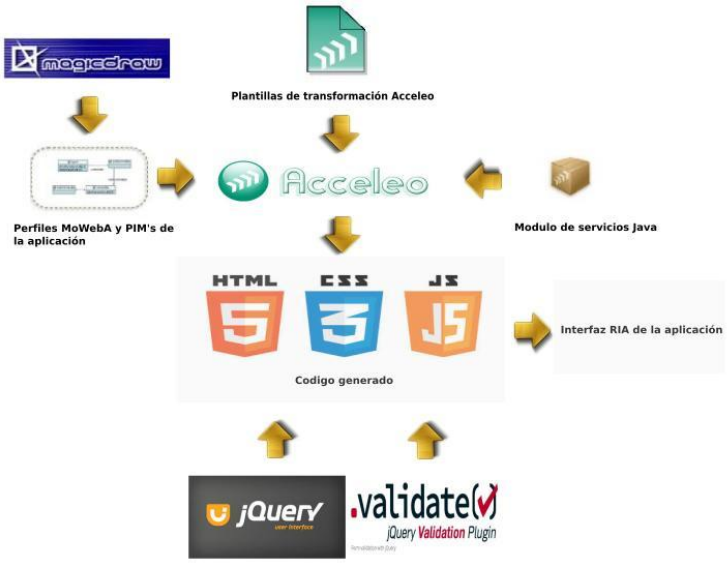

Fig. 3. Proceso de desarrollo para generar interfaces enriquecidas con MoWebA4RIA

Primeramente se modelan los PIM que representan a una aplicación particular utilizando los distintos perfiles UML de MoWebA4RIA (no sólo los de presentación, sino también los correspondientes a la lógica de negocios, navegación, etc.). Posteriormente, tanto los PIM como los perfiles son exportados al formato XMI del Eclipse Modeling Framework (EMF - https://www.eclipse.org/modeling/emf). Esto se lleva a cabo a fin de tener compatibilidad con la herramienta de transformación Acceleo, que toma como entrada modelos UML basados en el metamodelo Ecore1. Por medio de plantillas de transformación y módulos de servicio Java (Java Service Wrappers), que

[1] Ecore es un metamodelo nativo del Eclipse Modeling Framework para describir los modelos.

forman parte de Acceleo, es posible llevar a cabo las transformaciones necesarias sobre los modelos de entrada para obtener los archivos fuentes (.html, .css y .js) que representan a la aplicación en sí. Como parte de este trabajo, se han definido las plantillas de transformación necesarias para ejecutar las transformaciones. Cabe destacar que estas plantillas pueden ser utilizadas para generar otras aplicaciones, siempre que las mismas hayan utilizado la extensión MoWebA4RIA para modelarlas.

Para los widgets Accordion, Tabs, DatePicker, ToolTip y AutoSuggest se genera código HTML y Javascript correspondiente a la librería JQuery UI. Para los diversos tipos de validación local se genera código Javascript correspondiente al jQuery Validation plugin. Se genera también código CSS para estructurar cada uno de los elementos de interfaz (enriquecidos o no).

Finalmente, las librerías Javascript jQuery UI y jQuery Validation Plugin se invocan desde el código fuente generado para desplegar y utilizar las funcionalidades enriquecidas de la aplicación en uso.

4 Ilustración de la Propuesta

Con la intensión de ilustrar de una manera práctica las nuevas capacidades ofrecidas por MoWebA4RIA, y a la vez, de realizar ciertas comparaciones entre MoWebA y MoWebA4RIA, que sirvan como validación preliminar, los autores han implementado, con ambos enfoques, un sistema de administración de personas (Person Manager). Los requisitos del Person Manager, se refieren básicamente a poder agregar, editar, eliminar y listar a las personas de un sistema.

Se han definido cinco preguntas de investigación (PI) para guiar el análisis de las comparaciones entre la implementación del Person Manager con MoWebA y la implementación con MoWebA4RIA. A continuación se presentan las cinco preguntas de investigación junto con los datos que se han recabado para contestarlas y los análisis realizados a partir de dichos datos.

- **PI1:** ¿Consume una mayor cantidad de tiempo modelar la aplicación aplicando MoWebA4RIA que MoWebA?

  La Tabla 2 presenta el tiempo, en minutos, que se ha tardado para modelar la aplicación del PersonManager con MoWebA y con MoWebA4RIA.

Tabla 2. Tiempo de modelado y cantidad de generaciones de código hasta obtener la interfaz final al implementar el Person Manager con MoWebA y con MoWebA4RIA

	Tiempo de modelado (minutos)		**Cantidad de generaciones de código hasta obtener la interfaz final**	
	MoWebA	**MoWebA4RIA**	**MoWebA**	**MoWebA4RIA**
Agregar persona	50	56	3	4
Listar persona	27	28	1	1
Eliminar persona	29	30	2	3
Totales	106	114	6	8

Con MoWebA4RIA se tardó 8 minutos más en el proceso de modelado que con MoWebA. Esto se debe principalmente a que con MoWebA4RIA es necesario de-

finir un mayor número de propiedades en cada uno de los elementos que forman parte de la extensión, a diferencia de MoWebA. Esto concuerda con la intuición en el sentido de que cuanto mayor es el nivel de detalle en el modelo, mayor tiempo de modelado se requiere. Sin embargo, se puede considerar que este hecho no constituye una limitante demasiado grave, teniendo en cuenta que ese tiempo extra de modelado permite a la interfaz de la aplicación Person Manager enriquecerse notablemente.

- **PI2:** ¿Para cuál de los enfoques es necesaria una mayor cantidad de generaciones de código para obtener la interfaz de usuario final?
  La Tabla 2 presenta la cantidad de generaciones de código que se han realizado hasta obtener la interfaz final del PersonManager con MoWebA y con MoWebA4RIA. Una vez que los modelos de la aplicación se encontraban listos, el siguiente paso era generar código fuente a partir de ellos. Cuando a alguno de los modelos le faltaba definir alguna propiedad para alguno de sus elementos, entonces el código fuente generado en primera instancia, no reflejaba el resultado esperado. En estos casos, iterativamente se volvían a hacer cambios al modelo y luego se generaba de vuelta la aplicación. Un leve incremento en el número de generaciones para el enfoque MoWebA4RIA se pudo apreciar a partir de los datos recabados, con respecto a MoWebA. Debido a que los requerimientos RIA requieren un mayor nivel de detalle en los modelos de MoWebA4RIA, existe una mayor posibilidad de cometer errores u omisiones y, por ende, será necesaria una mayor cantidad de generaciones de código para ir depurando la aplicación.
- **PI3:** Desde el punto de vista de las presentaciones enriquecidas, ¿qué ventajas aportan las características RIA presentes en la aplicación implementada con MoWebA4RIA con respecto a MoWebA?
  El enfoque MoWebA4RIA ofrece numerosas ventajas con respecto a las presentaciones enriquecidas, evitando recargas innecesarias de las páginas por medio del widget Accordion y presentando widgets interactivos como los hDatePicker, AutoSuggest y ToolTip. El enfoque MoWebA no contempla tales elementos enriquecidos y navegar por cada una de sus páginas implica recargar completamente cada una de ellas. En la Fig. 4 se pueden apreciar los elementos enriquecidos presentes en la versión de MoWebA4RIA, en contrapartida a MoWebA, para el Person Manager.
- **PI4:** Desde el punto de vista de la lógica de negocios en el lado del cliente, ¿qué ventajas aportan las características RIA presentes en la aplicación implementada con MoWebA4RIA con respecto a MoWebA?
  El enfoque MoWebA4RIA permite llevar a cabo diversas validaciones en los campos de entrada de la aplicación, como campos que deben ser obligatorios, longitudes mínima y máxima de caracteres en un campo, validaciones de claves y formato de email. En contraparte el enfoque MoWebA, no soporta ningún tipo de validación. En la Fig. 5 se pueden apreciar los diversos tipos de validaciones locales presentes en la versión de MoWebA4RIA, en contrapartida a MoWebA, para el Person Manager.

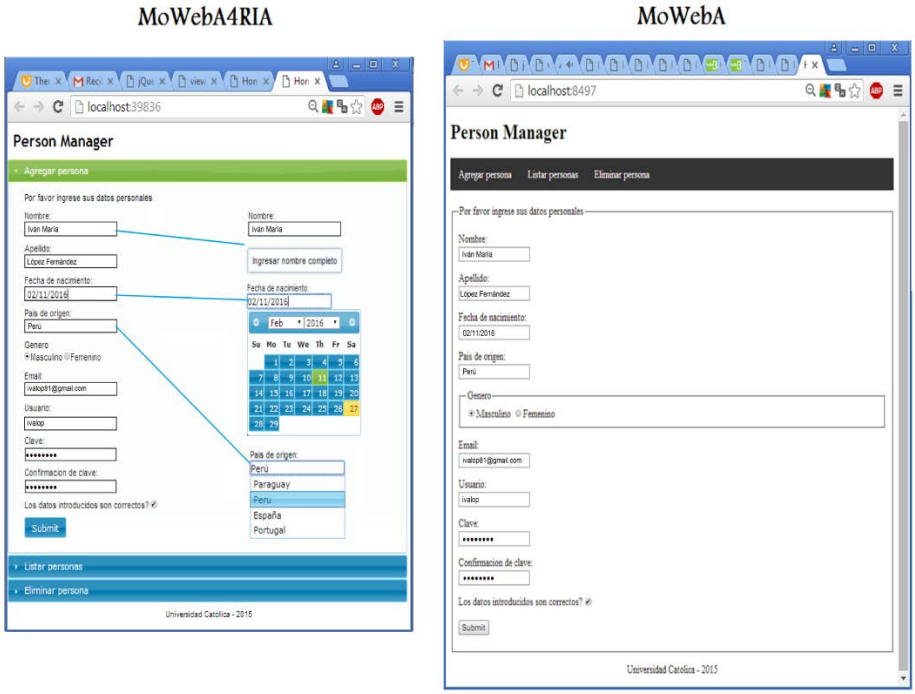

Fig. 4. Person Manager implementado con MoWebA4RIA y con MoWebA, desde el punto de vista de las presentaciónes enriquecidas

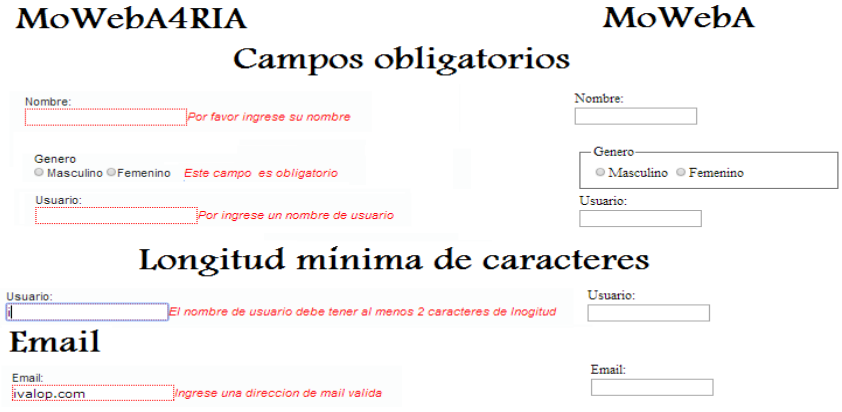

Fig. 5. Person Manager implementado con MoWebA4RIA y MoWebA, desde el punto de vista de la lógica de negocios en el lado del cliente

- **PI5**: Para cada una de las vistas del Person Manager, ¿qué cantidad de líneas de código para la interfaz de usuario se pudieron generar de manera automática a partir de los modelos, en cada uno de los enfoques implementados?
  La Tabla 3 presenta la cantidad de líneas de código resultantes de implementar el Person Manager con MoWebA y con MoWeba4RIA. Analizando primeramente el tamaño total del Person Manager para ambos enfoques, se puede apreciar que la versión de MoWebA posee 123 líneas de código menos que la versión de MoWebA4RIA (equivalente a un 32% menos). Esto se debe a que con MoWebA no se genera código Javascript en la interfaz de usuario, ya que su interfaz no posee elementos enriquecidos interactivos. El código que fue implementado manualmente en la aplicación, 53% (para el enfoque MoWebA) y 43% (para MoWebA4RIA) respectivamente, corresponde a código para refinar la aplicación final y código para el acceso a la capa lógica y de dominio de la aplicación. De ahí se puede concluir que el 47% del código del Person Manager fue generado de manera automática para el enfoque MoWebA y el 57% para el enfoque MoWebA4RIA.

Tabla 3. Cantidad de líneas de código al implementar el PersonManager con MoWebA y con MoWebA4RIA

	MoWebA			**MoWebA4RIA**		
	Líneas de código automáticas	**Líneas de código manuales**	**Totales**	**Líneas de código automáticas**	**Líneas de código manuales**	**Totales**
Agregar persona	51	56	107	135	56	191
Mostrar persona	1	45	46	3	45	48
Eliminar persona	7	27	34	31	27	58
Estructura y código común para todas las vistas (cabecera, estructura y pie de página)	67	10	77	52	38	90
Totales	126	138	264	221	166	387

Como se puede notar, si bien MoWebA4RIA implica un mínimo mayor esfuerzo de tiempo e iteraciones, al mismo tiempo ofrece la posibilidad de generar aplicaciones más enriquecidas conforme las RIA. Esto también se refleja en la automatización de la generación de código.

5 Conclusiones y Trabajos Futuros

En este trabajo se presenta MoWebA4RIA, una extención de MoWebA para incorporar las principales características y tecnologías de las RIA. La extensión abarca el

metamodelo de contenido, agregando una nueva reestructuración y clasificación de los elementos de interfaz, separando a los distintos componentes de interfaz en elementos de entrada, salida y control. Además, para enriquecer la implementación, algunos widgets interactivos comunes en las aplicaciones RIA se han agregado, precisamente Accordion, Tabs, AutoSuggest, DatePicker, ToolTip y LiveValidation. La extensión ha incorporado también mejoras en el metamodelo de estructura. Para la definición de la sintaxis concreta de la presentación, se han incoporado las extensiones con sus respectivos widgets en los perfiles de contenido y estructura.

Con la herramienta de transformación Acceleo, se implementaron las plantillas de transformación para la presentación. La transformación genera código para cada uno de los elementos definidos en el perfil de contenido, a partir de los PIM de entrada, en donde, para los widgets se genera código para la plataforma destino jQueryUI y jQuery validation plug-in. A partir del perfil de estructura, se genera código CSS con las posiciones de los elementos de interfaz según fueron establecidas en el PIM de entrada. Finalmente, una ilustración de la propuesta se llevó a cabo para comparar los aportes de MoWebA4RIA con respecto a MoWebA.

Entre los trabajos futuros, como primer paso nos interesa migrar a un entorno totalmente abierto, con el uso de Papyrus como herramienta de modelado. Por otro lado, consideramos que podrían incluirse otras características RIA a MoWebA, no sólo a nivel de la presentación, sino también en la comunicación cliente-servidor, en la lógica de negocios (no limitadas a las validaciones locales) y ofrecer cobertura de persistencia de datos en el lado del cliente, incluyendo también a los ambientes de dispositivos móviles. Otro trabajo futuro interesante sería realizar transformaciones para otros frameworks o plataformas destino RIA. Se prevé además la realización de validaciones más rigurosas, como casos de estudios o experimento, de la extensión propuesta, que incluya también comparaciones con otras propuestas que extienden a RIA (como OOH4RIA, WebML-RIA, etc.). Finalmente, resultaría de mucho interés explorar la posibilidad de no enriquecer directamente el PIM sino definir un ASM específico para RIA, siguiendo así una de las originalidades de MoWebA, no contemplada por otras propuestas. Con este aporte, se logrará incorporar conceptos y propiedades de arquitecturas en un nuevo modelo (ASM), manteniendo siempre la independencia de la plataforma (en el PIM) y disponer de un mecanismo de adaptación a la evolución dinámica de aplicaciones.

Agradecimientos. Este trabajo ha sido desarrollado con el apoyo financiero del Consejo Nacional de Ciencia y Tecnología (CONACYT, Paraguay) en el marco del proyecto MejorMDD (14-INV-056).

Referencias

1. Valverde F and Pastor O. Applying interaction patterns. In Towards a Model-Driven Approach for Rich Internet Applications Development.Proc. 7th Int. Workshop on Web-Oriented Software technologies, IWWOST 2008, 2008.

2. Martínez-Ruiz F J. *A Development Method for User Interfaces of Rich Internet Applications*. PhD thesis, Université catholique de Louvain, Belgium, August 2010.
3. Toffetti G, Comai S, Preciado J C, and Linaje M. State-of-the art and trends in the systematic development of rich internet applications. *J. Web Eng.*, 10(1):70–86, March 2011.
4. Allaire J. Requirements for rich internet applications. http://download.macromedia.com/pub/flash/whitepapers/richclient.pdf, March 2002.
5. Wright J and Dietrich J. Survey of existing languages to model interactive web applications. In *Proceedings of the fifth Asia-Pacific conference on Conceptual Modelling - Volume 79*, APCCM '08, pages 113–123, Darlinghurst, Australia, 2008. Australian Computer Society, Inc.
6. Preciado J C, Linaje M, Sanchez F, and Comai S. Necessity of methodologies to model rich internet applications. In *Proceedings of the Seventh IEEE International Symposium on Web Site Evolution*, WSE '05, pages 7–13, Washington, DC, USA, 2005. IEEE Computer Society.
7. Preciado J C, Linaje M, Morales-Chaparro R, Sanchez-Figueroa F, Zhang G, Kroiβ C, and Koch N. Designing rich internet applications combining *UWE* and rux-method. In *Proceedings of the 2008 Eighth International Conference on Web Engineering*, ICWE '08, pages 148–154, Washington, DC, USA, 2008. IEEE Computer Society.
8. Machado L, Filho O, and Ribeiro J. *UWE*-r: an extension to a web engineering methodology for rich internet applications. *WSEAS Trans. Info. Sci. and App.*, 6(4):601–610, April 2009.
9. Busch M and Koch N. Rich internet applications state-of-the-art. Technical report 0902, Programming and Software Engineering Unit (PST), Institute for Informatics, Ludwig-Maximilians-Universität München, Germany, December 2009.
10. González M, Casariego J, Bareiro J, Cernuzzi L, and Pastor O. Una propuesta *MDA* para las perspectivas navegacional y de usuarios. In *XXXVI Conferencia Latinoamericana de Informática (CLEI) - ISBN 978-99967-612-0-1*, page 58, Asunción, Paraguay, 2010.
11. González M, Cernuzzi L, and Pastor O. Una aproximación para aplicaciones web: *MoWebA*. In *XIV Congreso Iberoamericano en Software Engineering – CibSE*, Río de Janeiro, Brasil, 2011.
12. Urbieta M, Rossi G, Ginzburg J, and D. Schwabe. Designing the interface of rich internet applications. In *Proceedings of the 2007 Latin American Web Conference*, LA-WEB '07, pages 144–153, Washington, DC, USA, 2007. IEEE Computer Society.
13. Koch N, Pigerl M, Zhang G, and Morozova T. Patterns for the model-based development of Rias. In *Proceedings of the 9th International Conference on Web Engineering*, ICWE '9, pages 283–291, Berlin, Heidelberg, 2009. Springer-Verlag.
14. Fraternali P, Comai S, Bozzon A, and Carughi G T. Engineering rich internet applications with a model-driven approach. *ACM Trans. Web*, 4(2):7:1–7:47, April 2010.
15. Meliá S, Gómez J, Pérez S, and Díaz O. A model-driven development for gwt-based rich internet applications with ooh4ria. In *Proceedings of the 2008 Eighth International Conference on Web Engineering*, ICWE '08, pages 13–23, Washington, DC, USA, 2008. IEEE Computer Society.
16. Usage of Javascript Libraries for Websites, http://w3techs.com/technologies/overview/javascript_library/all, December 2015.