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](mailto:ivalop81@gmail.com)

{mgonzalez, nathalie.aquino, [lcernuzz}@uca.edu.py](mailto:lcernuzz%7d@uca.edu.py)

**Abstract.** Model Oriented Web Approach (MoWebA) es una aproximación de desarrollo de aplicaciones Web, basada en modelos y fundamentada en los principios de Model Driven Architecture (MDA). En los últimos años, MoWebA ha ido evolucionando para dar soporte a diversos aspectos de una aplicación Web. En ese contexto, este trabajo extiende MoWebA para dar cobertura a las Rich Internet Applications (RIA), dando lugar a MoWebA4RIA. Las RIA han surgido para paliar ciertos problemas existentes en la Web 1.0 y teniendo como una premisa importante la obtención de aplicaciones similares a las de escritorio, permiten explotar el lado del cliente. 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. 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 [].

Las RIA tienen cuatro características principales []: 1) es posible almacenar datos en el lado cliente, o bien distribuir el almacenamiento entre el cliente y el servidor; 2) 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; 3) 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 4) 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 [13], OOH [17], WebML [15], UWE [9], OOWS [2], entre otras) han tenido que evolucionar, agregando nuevos modelos o extendiendo los existentes, para dar cobertura a las características de las RIA. Muchas de las metodologías citadas han logrado una notable evolución incorporando los avances propuestos por las RIA, sin embargo, en la actualidad, ninguna de ellas ofrece un soporte total a las nuevas características RIA [6] [7] [10] [4].

Recientemente se ha propuesto un nuevo enfoque, denominado Model Oriented Web Approach (MoWebA) [11] [12], 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. Como parte de un objetivo mayor, que es el de extender MoWebA para que de soporte a todas las características RIA presentadas con anterioridad, este trabajo se enfoca en realizar extensiones para poder validar datos en el lado cliente y para dar soporte a widgets interactivos. Cabe mencionar que en paralelo a este trabajo, también se están realizando otros que se enfocan en las extensiones relacionadas a las demás características RIA.

En este trabajo, 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.

Lo que queda de este artículo se estructura como se explica a continuación. En la sección 2 se presenta el estado del arte de las metodologías de desarrollo Web basadas en modelos y que dan cobertura a características de las RIA. En la sección 3 se presenta la aproximación de desarrollo MoWebA. En la sección 4 se presenta el metamodelo de Contenido y Estructura de MoWebA y se presenta una propuesta de transformación de modelo a texto para las librerías jQueryUI y jQuery Validation Plugin, para cubrir algunas características RIA de las presentaciones enriquecidas y de la lógica de negocios en el lado cliente. En la sección 5 se evalúan los metamodelos de Contenido y Estructura extendidos con una ilustración. Finalmente, la sección 6 presenta un análisis de los resultados obtenidos y los posibles trabajos futuros.

1. Principales Enfoques de Desarrollo Web Basados en Modelos para las RIA

Los trabajos de Wright y Dietrich [6], y el de Preciado et al. [7] 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 [4] presenta un estudio más exhaustivo y reciente enfocado en hacer comparativas entre las diversas metodologías.

En el contexto de este trabajo, también se ha realizado un 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 [13], OOH4RIA [17], WebML-RIA [15], Patrones RIA con UWE [14], UWE-R [9] y UWE+RUX [8]. Cada una de estas metodologías es comparada con respecto a las cuatro características principales de las RIA, que han sido presentadas en la sección anterior. En las comparaciones en la Tabla 1, una casilla marcada con “si” implica que la metodología en cuestión da soporte a la característica RIA correspondiente.

**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** | | - | - | si | - | - | - |
| **Lógica de negocios en lado cliente** | Operaciones complejas | - | - | si | - | - | - |
| Operaciones específicas de dominio | - | - | - | - | si | - |
| Validación de datos local | si | si | si | - | - | si |
| **Interfaces enriquecidas** | Manejo de eventos en lado cliente | - | - | si | si | si | si |
| Widgets interactivos | si | si | - | si | si | si |
| Paradigma de página única | si | si | si | si | - | si |
| Contenido multimedia | - | si | - | - | si | si |
| **Comunicación cliente-servidor** | Sincronización de datos | - | - | si | - | si | si |
| Actualizaciones parciales de página | si | si | si | si | si | si |
| Push y pull | - | - | si | - | si | - |

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.

En este trabajo, en lugar de extender alguna de las metodologías Web ya existentes, se ha optado por extender MoWebA por los motivos que se exponen a continuación. Una característica deseable de una aproximación de desarrollo basada en modelos es que el lenguaje de modelado que se utilice sea estándar y moderado por una comunidad, como por ejemplo UML. Esta característica hace posible que los modelos puedan ser desplegados en diversas herramientas CASE de modelado (libres o licenciadas). Otra característica deseable es que sus modelos independientes de plataforma (PIM – Platform Independent Model) no contengan detalles de una arquitectura destino, como por ejemplo RIA. Entre las metodologías analizadas, solo OOH4RIA, Patrones RIA con UWE y UWE-R poseen un lenguaje de modelado cien por ciento basado en UML. Sin embargo, sus modelos PIM poseen detalles de la arquitectura destino RIA. Por su parte, MoWebA utiliza UML como lenguaje de modelado y además, como se verá en la próxima sección, incorpora un modelo específico de arquitectura (ASM - Architectural Specific Model), el cual es el indicado para incorporar características RIA. Además de todo esto, se tiene el interés de que MoWebA siga evolucionando e incorpore características RIA.

1. La Aproximación MoWebA (Model Oriented Web Approach)

MoWebA [11] [12] adopta los principios de MDA y propone un enfoque que incluye dos procesos complementarios: proceso de modelado y proceso de transformación.

Por una parte, 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 modelado 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 detino. 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.

Antes de pasar a los detalles de la extensión de MoWebA para dar soporte a las RIA, se presenta una breve introducción de la capa de presentación de MoWebA, ya que esta capa es la que ha sufrido principalmente las modificaciones. La capa de presentación de MoWebA 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 el modelo de presentación: el diagrama de contenido y el diagrama de estructura. Las clases que aparecen en color salmón y celeste en la Fig. 1 representan a los metamodelos originales de contenido (content) y estructura (layout) de MoWebA.

El diagrama de contenido permite especificar los diferentes elementos que serán presentados a los usuarios del sistema. Este diagrama se compone de un conjunto de páginas de presentación. A su vez, cada página de presentación se compone 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 algún tipo de información no representado en el diagrama de entidades, como por ejemplo, el titulo de la página web o un texto informativo; 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 de layout por aplicación.

Más detalles de la capa de presentación pueden ser conultados en [referencia MoWebA].

1. Una propuesta de extensión a MoWebA: MoWebA4RIA

El proceso de extensión de MoWebA para que de soporte a las RIA ha consistido de los siguientes pasos. Primero, se han mejorado y extendido los metamodelos de la capa de presentación de MoWebA. En el metamodelo de contenido, se han agregado nuevos elementos enriquecidos (widgets) y propiedades de validación local de campos de entrada. En el metamodelo de estructura, se han incorporado mejoras que permiten identificar el tipo de coordenadas (en pixeles o porcentajes) del posicionamiento de las diferentes regiones de interfaz definidas. 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 y jQuery Validation. jQuery UI y jQuery Validation fueron elegidos porque … Estas transformaciones 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/). Se ha utilizado Acceleo porque … Como resultado del proceso, MoWebA ha evolucionado a MoWebA4RIA. A continuación, se presentan más detalles del proceso descripto.

* 1. Extensiones a los Metamodelos y Perfiles de Contenido y Estructura

Las extensiones propuestas a los metamodelos de Contenido y Estructura de MoWebA se presentan en la Fig. 2. En ellos se despliegan los diversos elementos que permiten representar una interfaz de usuario enriquecida. Los diferentes elementos tanto del metamodelo de Contenido como del de Estructura, fueron catalogados en diferentes colores para diferenciarlos de su forma original, estableciendo el color salmón para las clases que no han sufrido ningún cambio con respecto a la versión original de MoWebA, color celeste para las clases originales de MoWebA que han sufrido modificaciones de agregado, modificación o eliminación de propiedades, y color verde para las clases y enumeraciones nuevas.

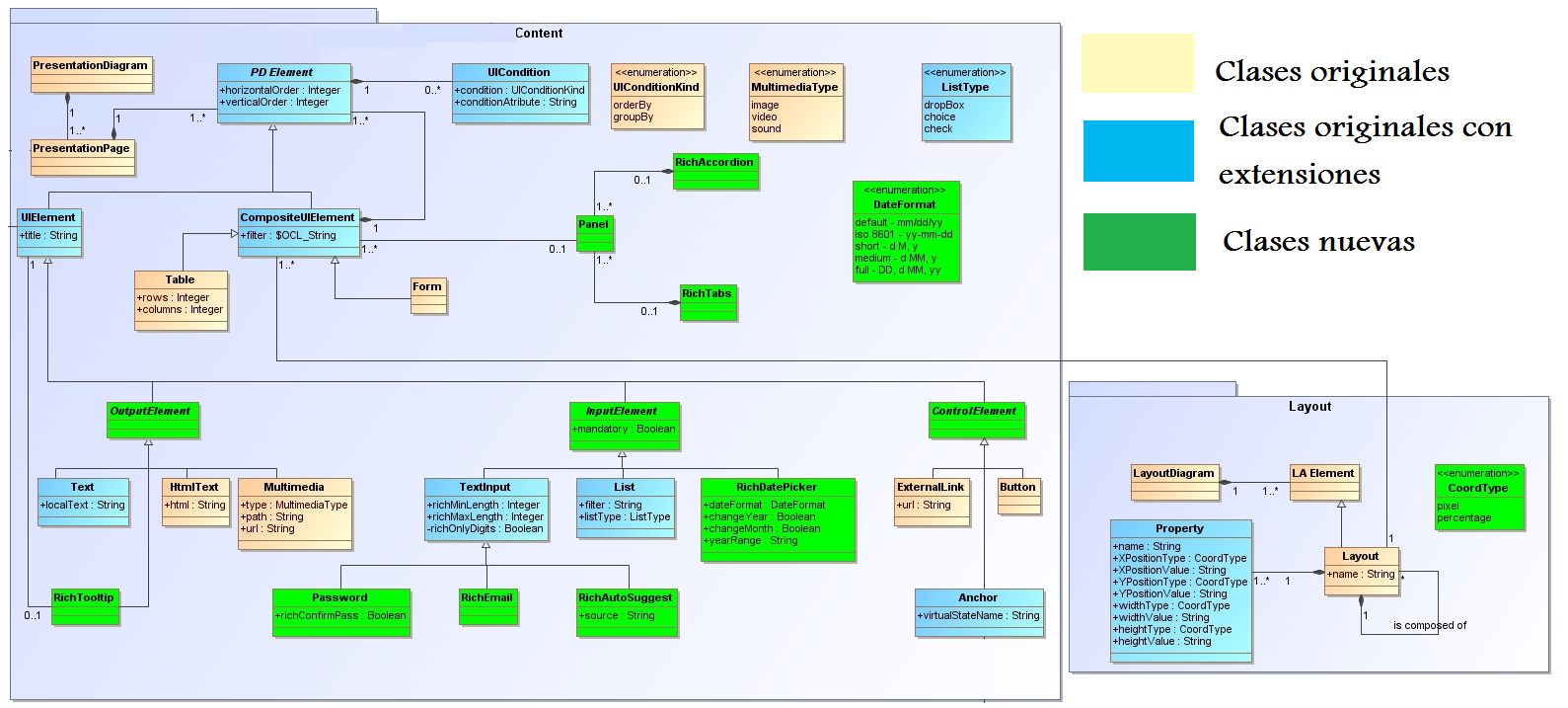


Fig. 1. Metamodelos de contenido (content) y de estructura (layout) de MoWebA4RIA

Diagrama o modelo de contenido - Presentation Page.

PDElement, patrón de composición, CompositeUIElement, UIElement. UICondition.

Se estableció una jerarquía entre los elementos compuestos (CompositeUIElement) y los elementos simples o elementos hoja (UIElements), aplicando el patrón de diseño composite. El patrón composite permite crear una jerarquía de elementos anidados unos dentro de otros. Cada elemento permite alojar una colección de elementos del mismo tipo, hasta llegar a los elementos “reales” que se corresponden con los nodos “hoja” del árbol [1]. Para el caso de MoWebA, cada CompositeUIElement puede contener uno o más elementos PD Element que a la vez pueden ser compuestos (CompositeUIElemet) o simples u hojas (UIElement).

El PD Element, que corresponde a una clase padre abstracta, 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. El PD Element puede acceder al modelo de datos y para ese caso, pueden establecerse cero o más condiciones sobre estos elementos, del tipo order by y group by, que forman parte de la clase UICondition.

Formando parte también de la extensión, los CompositeUIElement pueden, o no, tener Panels asociados. A su vez, los Panels pueden estar asociados a uno o muchos CompositeUIElement. Los Panels pueden formar parte de un RichAccordion o un RichTabs, y tanto el RichAccordion como el RichTabs pueden contener uno o muchos Panels. Cada uno de estos Panels, permite aglomerar a uno o muchos elementos de interfaz CompositeUIElement. Cada Panels puede formar parte de un RichAccordion o un RichTabs. De manera inversa, un RichAccordion o un RichTabs se compone de uno o muchos Panels.

Como un nuevo aporte al metamodelo de Contenido de MoWebA, se propone la clasificación de los diferentes elementos simples de interfaz (UIElement), en elementos de entrada, salida y control respectivamente. Esto fue necesario para clasificar mejor a los elementos de interfaz y para una mayor claridad dentro del metamodelo de Contenido. Los distintos UIElements se clasifican entonces de la siguiente forma:

* ***Elementos de salida* (*OutputElement*):** Comprende a los elementos de interfaz enriquecidos y tradicionales encargados de desplegar o mostrar información en las páginas de presentación. En esta categoría se engloba a los elementos text, htmlText, multimedia y richToolTip.
* **Elementos de entrada (*InputElement*):** Comprende a los elementos de interfaz enriquecidos y tradicionales encargadas de obtener una entrada desde la interfaz de usuario. En esta categoría se engloba a los elementos textInput, list, richAutoSuggest, richDatePicker, password y richEmail.
* ***Elementos de control* (*ControlElement*)*:*** Comprende a los elementos de interfaz tradicionales encargados de obtener una orden de navegación o cambio de página. En esta categoría se engloba a los elementos externalLink, anchor y button.

A continuación se describen cada uno de los elementos RIA que forman parte de la extensión al metamodelo de Contenido de MoWebA con sus respectivas propiedades.A continuación, de presentan los elementos introducidos en la extensión:

### 1. RichAutoSuggest: 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). Por otro lado, es posible obtener el listado de palabras desde el modelo de datos de MoWebA, estableciendo una relación entre el elemento RichAutoSuggest y un Value Object que contiene la información necesaria de una entidad en particular.

### 2. RichDatePicker: 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 una lista que formará parte del richDatePicker. También es posible asignar al valor etiquetado yearRange un rango de años para el richDatePicker. El atributo yearRange permite la selección de fechas pasadas, como el año de nacimiento o fechas históricas. Por último, el valor etiquetado booleano changeMonth permite desplegar una lista con todos los meses del año para una rápida selección.

### 3. RichToolTip: Este elemento de salida, tiene como objetivo enriquecer con mensajes personalizados a cualquiera de los elementos que forman parte de la clasificación de elementos de entrada, salida y control.

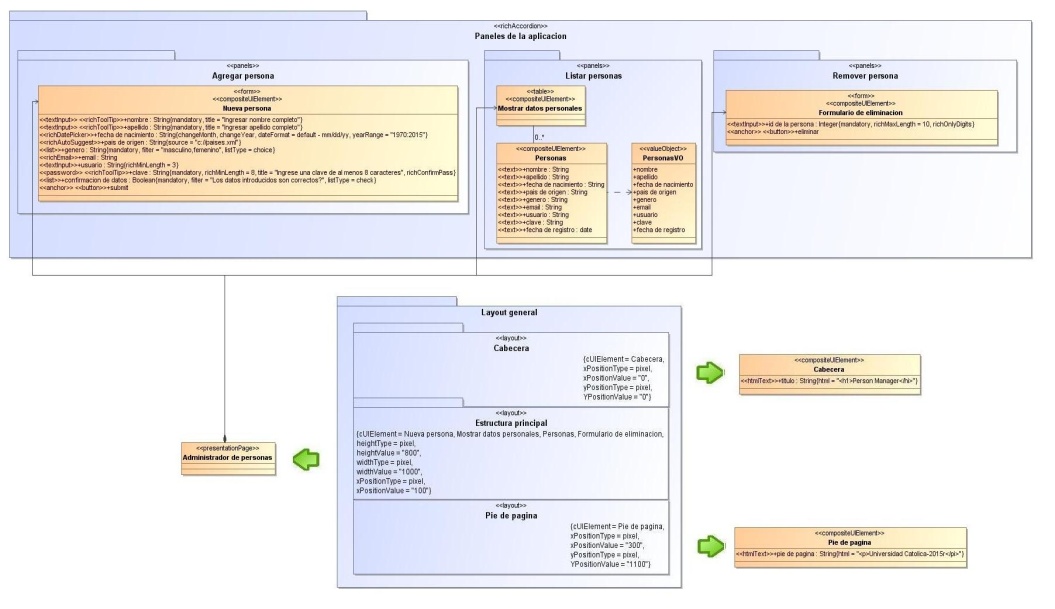
### 4. Live Validation: corresponde a un conjunto de extensiones que permite llevar a cabo validaciones locales a diversos elementos pertenecientes a un formulario. Estas validaciones pueden llevarse a cabo sobre diversos elementos de entrada, como los del tipo TextInput, RichEmail y Password, y a los elementos del tipo List, choice y check.

### 5. RichAccordion y RichTabs: permiten encapsular a varios elementos de interfaz de MoWebA dentro de paneles colapsables para presentar información en una cantidad limitada de espacio. Entre los elementos que pueden ser desplegados en los paneles se encuentran los UIElement, en cualquiera de sus extensiones InputElement, OutputElement o ControElement, como así también los CompositeUIElements, Table y los Form.

El metamodelo de Estructura no ha sufrido muchos cambios con respecto a su versión original. Entre las adaptaciones que se han realizado a este metamodelo se cuentan los cambios llevados a cabo a los atributos de la clase Properties, XPosition, YPosition, width y height. Cada uno de estos atributos se divide en dos para distinguir su tipo y valor. Por lo tanto, los atributos quedan como XPositionType y XPositionValue, YPositionType y YPositionValue, widthType y widhValue y finalmente heightType y heightValue. Los tipos de coordenadas que forman parte de la enumeración CoordType son pixel y percentage. Cualquiera de estas coordenadas puede establecerse para configurar la posición de cada uno de los CompositeUIElement definidos en el metamodelo de Contenido.

### De los metamodelos de Contenido y Estructura presentados, se derivan los perfiles UML, para agregar las características propias de MoWebA y por ende hacer posible la representación de la sintaxis concreta de MoWebA. Por motivos de espacio estos perfiles no se presentan en este resumen.

En la Fig. 3 se presenta un ejemplo de PIM con MoWebA4RIA en la que se contemplan varios elementos que forman parte de la extensión.



**Fig. 3.** PIM de ejemplo con MoWebA4RIA

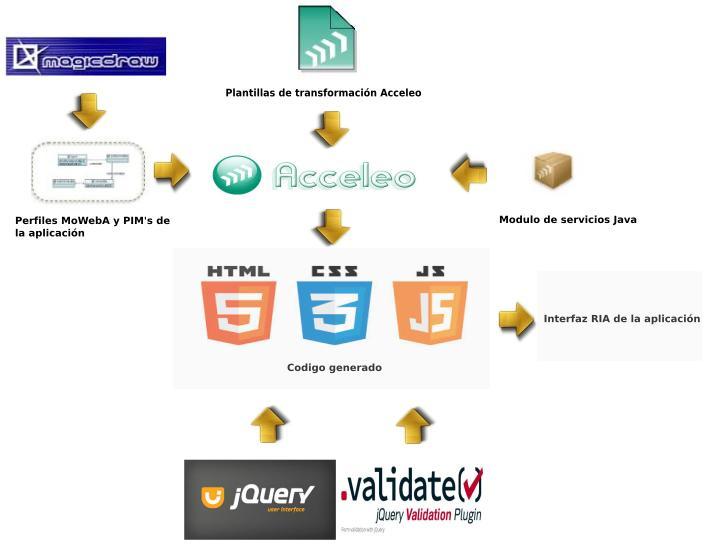
* 1. Transformación para la Generación de Interfaces Enriquecidas

Para realizar la transformación a código, primeramente se modelan los PIM que representan a una aplicación en particular utilizando distintos perfiles UML de MoWebA. Posteriormente tanto los PIM como los perfiles de MoWebA son exportados al formato XMI del EMF[[1]](#footnote-2). Esto de por sí es llevado a cabo a fines de tener compatibilidad con la herramienta de transformación M2T Acceleo[[2]](#footnote-3), que toma como entrada modelos UML basados en el metamodelo Ecore[[3]](#footnote-4).

Por medio de las plantillas de transformación y los módulos de servicio Java (Java Service Wrappers), que 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. En la Fig. 4 se presenta el proceso de modelado y generación de interfaces enriquecidas. Para el caso de MoWebA con RIA, se genera código HTML y Javascript para la librería jQueryUI, específicamente el código para los widgets RichAccordion, RichTabs, RichDatePicker, RichTooltip, y RichAutoSuggest, y jQuery Validation plug-in para los diversos tipos de validación local. De igual manera que en su forma original, es posible generar el código CSS para estructurar cada uno de los elementos de interfaz enriquecidos (o no). Finalmente las librerías Javascript jQueryUI y jQuery Validation Plugin se invocan desde el código fuente generado para tener todas las funcionalidades enriquecidas de la aplicación a partir del código generado.

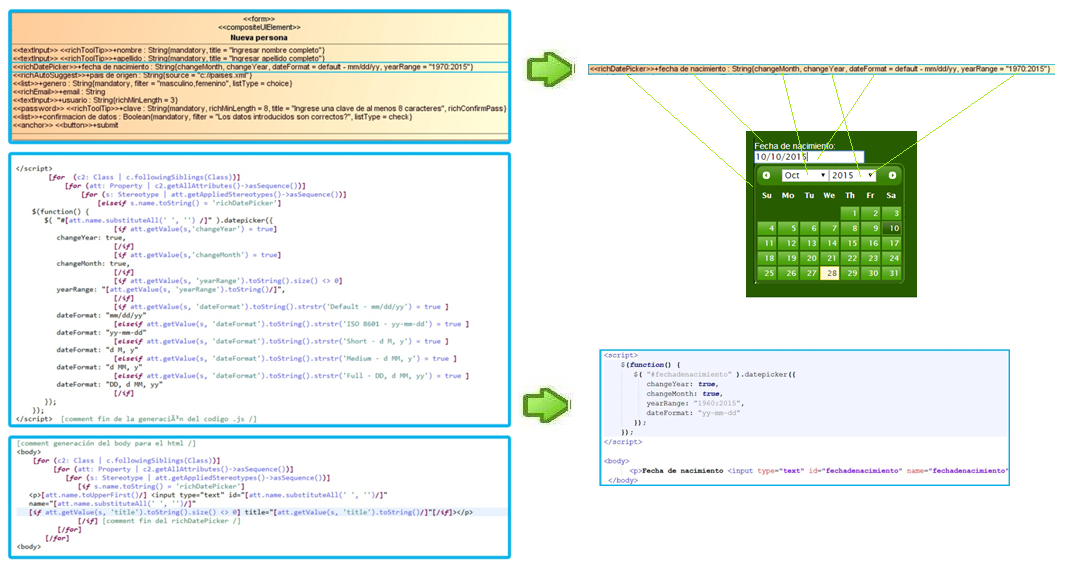
Existen diferentes tecnologías para el desarrollo e implementación de las RIA. Las implementaciones basadas en Javascript o librerías Ajax son las más utilizadas en la actualidad, debido a que utilizan tecnologías de uso abierto y estandarizado como Javascript, HTML y CSS. Estas librerías tienen como objetivo abstraer a los desarrolladores de tener que lidiar directamente con el DOM (Document Object Model) para la disposición de los elementos en las páginas Web, ofreciendo capas de software amigable, reduciendo notablemente los tiempos de desarrollo y mejorando la productividad. Entre los principales frameworks de desarrollo para las RIA[[4]](#footnote-5) se pueden citar a jQueryUI, JQuery Validation, Prototype, MooTools, YUI Library, Dojo Toolkit, Angular.JS, entre otros.

Estas librerías también buscan explotar el lado del cliente en las aplicaciones y minimizar las interacciones con el lado servidor, para que de esta forma se obtenga un mejor rendimiento. A la par de permitir a los desarrolladores implementar aplicaciones a un alto nivel de abstracción, las librerías ofrecen una gran variedad de widgets interactivos que son de uso común en las aplicaciones Web. Los widgets para las RIA representan elementos enriquecidos para la interfaz de usuario, que tienen como objetivo ofrecer una mayor interactividad, dada sus características dinámicas y un comportamiento general, similar a los patrones de comportamiento.



**Fig. 4.** Fases de desarrollo para la propuesta de extensión a MoWebA

En la Fig. 5 se presenta un ejemplo de como se establecen los valores etiquetados para un element en particular enriquecido, precisamente el richDatePicker y como estas propiedades son utilizadas por las reglas de transformción definidas en Acceleo para la generación automática de código.



**Fig. 5.** PIM y regla de transformación para el elemento richDatePicker en MoWebA4RIA

1. Ilustración de la Propuesta

Con la intensión de ilustrar de una manera práctica las extensiones llevadas a cabo a la aproximación Web MoWebA, se ha tomado la decisión de implementar un sistema que refleje tales extensiones. Con este trabajo, se ha logrado recabar datos que permiten intuir que la propuesta de extensión presentada ofrece cobertura a algunas de las diversas características que contemplan las RIA que han sido analizadas. El objetivo de esta ilustración, es analizar estas características por medio de la resolución de un toy problem denominado Person Manager. El Person Manager es una aplicación Web que contiene en sus especificaciones funcionales características de las RIA y resulta lo suficientemente expresiva para ilustrar la propuesta de extensión.

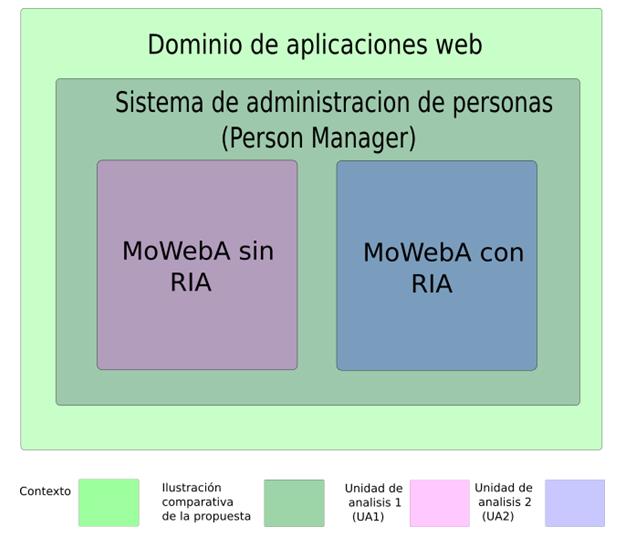
Si bien en una primera instancia, se ha deseado llevar a cabo un caso de estudio para validar las extensiones RIA hechas a la aproximación MoWebA y de esa forma, ofrecer una mayor formalidad a los resultados obtenidos, no fue posible implementar esa idea. Esto se debió a la principal limitante de no contar con la población debidamente instruida en la aproximación MoWebA para validar la extensión en el tiempo pre-estimado de desarrollo del caso de estudio. De allí, que el autor del trabajo tuvo que abocarse a la tarea de diseñar el caso, preparar la colección de datos, colectar los datos, analizar los datos colectados y reportar los resultados. Debido a que existían muchas amenazas a la validez de los datos reportados que no podían atenuarse, se optó por el método de ilustración. No obstante, con el objetivo de seguir una secuencia estructurada de pasos se ha optado por seguir las recomendaciones de Runeson [16] con respecto a cómo realizar casos de estudio.

* 1. El Caso y las Unidades de Análisis

El caso ilustrativo consistió en un sistema de administración de personas (Person Manager) en el dominio de las aplicaciones Web, que fue elegido entre varias otras opciones debido a que sus requerimientos funcionales ofrecen la posibilidad de representar a todas las características RIA que han sido agregadas al enfoque MoWebA, de una manera clara y sencilla.

El caso fue analizado desde dos unidades de análisis como se puede apreciar en la Fig. 6. La primera unidad de análisis se refiere a la implementación de la capa de presentación del Person Manager con MoWebA sin extensiones RIA. La segunda unidad de análisis se refiere a la implementación de la misma capa de presentación del caso estudiado con la nueva propuesta de extensión RIA a MoWebA. El Person Manager está basado en el trabajo de Gharavi [18].

La secuencia de acciones llevadas a cabo en el transcurso de este proceso ilustrativo fue la siguiente:



**Fig. 6.** Ilustración del sistema Person Manager implementado con MoWebA desde dos enfoques distintos

1. Se diseñó el toy problem Person Manager y se separó el problema en 2 unidades de análisis.
2. Se elaboraron las preguntas de investigación de interés y se identificaron las variables de medición para la colección de los datos.
3. Se colectaron los datos en base a las mediciones hechas.
4. Se analizaron los datos colectados y se reportaron los resultados.
   1. Preguntas de Investigación

Las siguientes cinco preguntas de investigación (PI) fueron propuestas para el análisis:

* PI1: ¿Consume una mayor cantidad de tiempo modelar la aplicación aplicando MoWebA con RIA que MoWebA sin RIA?
* PI2: ¿Para cuál de los enfoques es necesaria una mayor cantidad de generaciones de código para obtener la interfaz de usuario final?
* PI3: Desde el punto de vista de las presentaciones enriquecidas, ¿qué ventajas aportan las características RIA presentes en la aplicación implementada con MoWebA con RIA con respecto a MoWebA sin RIA?
* 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 MoWebA con RIA con respecto a MoWebA sin RIA?
* 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?
  1. Colección de los Datos

Para las PI1, PI2 y PI5 los datos correspondientes a las variables de medición que fueron previamente definidas, fueron recabados y almacenados en tablas diseñadas para el efecto. Para las PI3 y PI4, las capturas de pantalla de cada uno de los enfoques implementados, fueron la fuente de datos para concluir los resultados.

* 1. Análisis e Interpretación de los Resultados

En base a los datos obtenidos en cada una de las mediciones, se respondió cada una de las preguntas de investigación anteriormente presentadas y se obtuvieron ciertas conclusiones:

* **Con respecto a PI1**: En la Tabla 2, se presentan los tiempos de modelado para cada uno de los enfoques tomados. El enfoque MoWebA con RIA tardó 8 minutos más en el proceso de modelado que MoWebA sin RIA. Esto se debe principalmente a que para establecer características RIA con MoWebA, es necesario definir un mayor número de propiedades (valores etiquetados) intrínsecas en cada uno de los elementos que forman parte de la extensión, a diferencia de MoWebA sin RIA. En otras palabras, 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, 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.

**Tabla** 2**.** Tiempo de modelado para cada uno de los enfoques

|  |  |  |
| --- | --- | --- |
| **Tiempos en minutos de modelado y generación de código para cada una de las vistas del Person Manager** | **MoWebA sin RIA** | **MoWebA con RIA** |
| **Agregar persona** | 50 minutos | 56 minutos |
| **Listar persona** | 27 minutos | 28 minutos |
| **Remover persona** | 29 minutos | 30 minutos |
| **Totales** | 106 minutos | 114 minutos |

* **Con respecto a PI2**: En la Tabla 3 se presenta el número de generaciones de código para cada uno de lo enfoques. 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 MoWebA con RIA se pudo apreciar a partir de los datos recabados, con respecto a MoWebA sin RIA. Las vistas de agregar persona y de borrar persona son las que incurrieron en la mayor cantidad de generaciones de código. Debido a que los requerimientos RIA requieren un mayor nivel de detalle en los modelos para el caso de MoWebA con RIA con respecto a MoWebA sin RIA, existe una mayor posibilidad de cometer errores en los modelos y por ende será necesaria una mayor cantidad de generaciones de código para ir depurando la aplicación.

**Tabla** 3**.** Cantidad de generaciones de código para cada uno de los enfoques para la obtención de la interfaz final

|  |  |  |
| --- | --- | --- |
| **Número de generaciones de código para cada una de las vistas del Person Manager** | **MoWebA sin RIA** | **MoWebA con RIA** |
| **Agregar persona** | 3 | 4 |
| **Listar persona** | 1 | 1 |
| **Remover persona** | 2 | 3 |
| **Totales** | 6 | 8 |

* **Con respecto a PI3**: El enfoque MoWebA con RIA ofrece numerosas ventajas con respecto a las presentaciones enriquecidas, evitando recargas innecesarias de las páginas por medio del widget richAccordión y presentando widgets interactivos como los richDatePicker, richAutoSuggest y richToolTip. El enfoque MoWebA sin RIA no contempla tales elementos enriquecidos y navegar por cada una de sus páginas implica recargar completamente cada una de ellas. ­En la Fig. 7 se pueden apreciar los elementos enriquecidos presentes en MoWebA con RIA en contrapartida a MoWebA sin RIA para el Person Manager.



**Fig.** 7**.** MoWebA con RIA y MoWebA sin RIA desde el punto de vista de las presentaciónes enriquecidas

* **Con respecto a PI4**: El enfoque MoWebA con RIA 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 sin RIA, no contiene ningún tipo de validación. ­En la Fig. 8 se pueden apreciar los diversos tipos de validaciones locales presentes en MoWebA con RIA en contrapartida a MoWebA sin RIA para el Person Manager.

**Fig.** 8**.** MoWebA con RIA y MoWebA sin RIA desde el punto de vista de la lógica de negocios en el lado del cliente

* **Con respecto a PI5:** Analizando primeramente el tamaño total del Person Manager para ambos enfoques, se puede apreciar que el enfoque sin extensiones RIA posee 123 líneas de código menos que el enfoque con extensiones RIA (equivalente a un 32% menos). Esto se debe a que en el enfoque sin RIA no se genera código Javascript en la interfaz de usuario, ya que su interfaz no posee elementos enriquecidos interactivos. Teniendo en cuenta que el objetivo de este artículo está enmarcado en los front-ends de las interfaces de usuario Web, el código que fue implementado manualmente en la aplicación, 53% (para el enfoque MoWebA sin RIA) y 43% (para MoWebA con RIA) 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 sin RIA y el 57% para el enfoque MoWebA con RIA.

**Tabla** 4**.** Líneas de código para ambos enfoques del Person Manager

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Líeas de código/ Vistas de la aplicación** | **MoWebA sin RIA** | | | **MoWebA con RIA** | | |
| **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 |
| **Remover 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 |

1. Conclusiones y Trabajos Futuros

En este trabajo se ha extendido MoWebA para incoprorar las principales características y tecnologías de las RIA. La extensión abarca el metamodelo de Contenido de MoWebA, 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, utilizando el patrón de diseño general composite. Además, para enriquecer la implementación, algunos widgets interactivos comunes en las aplicaciones RIA se han agregado, precisamente richAccordion, richTabs, richAutoSuggest, richDatePicker y richToolTip y el live Validation. El metamodelo de Estructura también se extendió para permitir definir cada una de las coordenadas posicionales de las páginas en pixeles o en porcentajes. Para la definición de la sintaxis concreta de la presentación, se agregaron los nuevos widgets al perfil de Contenido de MoWebA, y las nuevas coordenadas al perfil de Estructura.

Con la herramienta de transformación (modelo a texto) M2T 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 de MoWebA, 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 evaluativa de la propuesta se llevó a cabo para presentar los aportes realizados a la capa de presentación de MoWebA.

Entre los trabajos futuros, 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, a diferencia de las validaciones locales, y ofrecer cobertura de persistencia de datos en el lado del cliente. Otro trabajo futuro interesante sería realizar transformaciones para otros frameworks o plataformas destino RIA.

Referencias

1. Freeman E, Robson E, Sierra K, and Bates B. *Head first Design Patterns*, ISBN 978-0-5960-07126. O’ Reilly Media, 2014.
2. 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.
3. Martínez-Ruiz F J. *A Development Method for User Interfaces of Rich Internet Applications*. PhD thesis, Université catholique de Louvain, Belgium, August 2010.
4. 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.
5. Allaire J. Requirements for rich internet applications. http://download.macromedia.com/pub/flash/whitepapers/richclient.pdf, March 2002.
6. 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. 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.
14. 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.
15. 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.
16. Runeson P, Höst M, Rainer A, and Regnell B. *CASE STUDY RESEARCH IN SOFTWARE ENGENEERING. Guidelines and Examples*, ISBN 978-1118104354. Jhon Wiley & Sons,Inc, Hoboken, New Jersey, 2012.
17. Meliá S, Gómez J, Pérez S, and Dáz 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.
18. Vahid Gharavi S V. Model-driven development of *Ajax* web applications. Master’s thesis, Faculty EEMCS, Delft University of Technology, September 2008.

1. **Eclipse Modelling Framwwork:** <https://www.eclipse.org/modeling/emf> , 2015 [↑](#footnote-ref-2)
2. **Acceleo:** <https://eclipse.org/acceleo> , 2015 [↑](#footnote-ref-3)
3. **Ecore:** Metamodelo nativo que forma parte del core del EMF para describir a los modelos [↑](#footnote-ref-4)
4. **List of Ajax frameworks :** <https://en.wikipedia.org/wiki/List_of_Ajax_frameworks> , 2015 [↑](#footnote-ref-5)