**Capítulo 3 Una Extensión RIA para la metodología web MoWebA**

Se ha visto en la sección final del capítulo anterior, una breve introducción de los alcances de la metodología web MoWebA, presentando sus diferentes capas y fases de desarrollo y transformación. Se ha mencionado el hecho de que MoWebA resulta ser una metodología flexible para llevar a cabo extensiones que le permiten de cierto modo, mantenerse vigente con los nuevos avances que constantemente afectan a las aplicaciones web. También se ha tenido en cuenta el hecho que las RIA forman parte de esa evolución y que las metodologías web basadas en MDD/MDA necesitan tener en cuenta estos cambios.

Dentro de las diversas características que presentan las RIA, las presentaciones enriquecidas toman un papel preponderante, debido a que en esta característica es donde se encuentra presente el dinamismo e interactividad que diferencia en contrapartida a las aplicaciones de la web 1.0. Los widgets interactivos colaboran de manera notable a este enriquecimiento, y tanto es así que en la actualidad es difícil encontrar aplicaciones web, que carezcan de estos elementos para la interfaz de usuario.

Sin embargo, se ha visto que las diversas metodologías presentadas basadas en MDD/MDA ofrecen cobertura con respecto a los diversos tipos de widgets existentes, pero o bien, los mecanismos de extensión para la cobertura, son muy tediosos, con numerosas cadenas de transformaciones M2M y M2T; por ejemplo para el caso de OOH4RIA, u bien las herramientas para llevar a cabo el enriquecimiento son de uso propietario; por ejemplo WebML+ RUX o UWE+ RUX. También se ha notado que muchas de las transformaciones M2T no se llevan a cabo automáticamente sino de manera semiautomática o manual, como es el caso de UWE con patrones.

La capa de presentación de MoWebA en su definición original, contiene diversos elementos para la interfaz de usuario que son de uso común en la aplicaciones web 1.0. En este capítulo se presentarán nuevos elementos que forman parte de la extensión propuesta, precisamente los widgets comunes en las RIA que fueron presentados en la sección anterior. Los nombres de tales widgets (accordion, tabs, autocomplete, datePicker, tooltip y el fieldLiveValidation) serán presentados en MoWebA como richAccordion, richTabs, richAutoSuggest, richDatePicker, richToolTip y richFieldLiveValidation respectivamente.

Primeramente se dará a conocer la sintaxis abstracta de MoWebA para el modelado de los elementos de presentación, precisamente el metamodelo de presentación, que abarca a los metamodelos de contenido (content) y posicionamiento (layout) para luego continuar con los perfiles (profiles) UML para la representación de la sintaxis concreta de la presentación; tanto para el contenido como así también el posicionamiento. Se cerrará el capítulo con algunos ejemplos de algunos PIM obtenidos a partir de los perfiles de MoWebA y una breve discusión acerca del enfoque tomado para el modelado de widgets con MoWebA.

3.1 El enfoque utilizado con MoWebA para la generación de interfaces enriquecidas.

El proceso tenido en cuenta en este trabajo de fin de carrera para el modelado y generación de interfaces enriquecidas (también conocidos como los *front-ends* de las aplicaciones) se describe a continuación en la Figura 1. Primeramente se modelan los PIM que representan a una aplicación en particular, por medio de los distintos perfiles UML con las que cuenta MoWebA. Estos perfiles son extensiones a UML para agregar características específicas de MoWebA a los metamodelos, para que de esta forma sea posible representar la sintaxis concreta del *DSL*. Estos modelos PIM junto a los perfiles son llevados a cabo basados en el estándar MOF que forma parte del enfoque MDA, utilizando la herramienta MagicDraw[[1]](#footnote-1). Posteriormente tanto los PIM como los perfiles son exportados al formato xmi del EMF[[2]](#footnote-2) que está basado en el metamodelo llamado Ecore. Esto de por sí es llevado a cabo a fines de tener compatibilidad con la herramienta de transformación M2T Acceleo[[3]](#footnote-3), que toma como entrada modelos UML que están basados en el lenguaje de metamodelado Ecore.

Una vez exportados los modelos (PIM y profile) al Acceleo, por medio de las plantillas de transformación y los módulos de servicio en 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 (.js, .html y .php) que representan a la aplicación en sí. Finalmente la libreía javascript jQueryUI se agrega al código fuente generado para tener todas las funcionalidades enriquecidas de la aplicación.

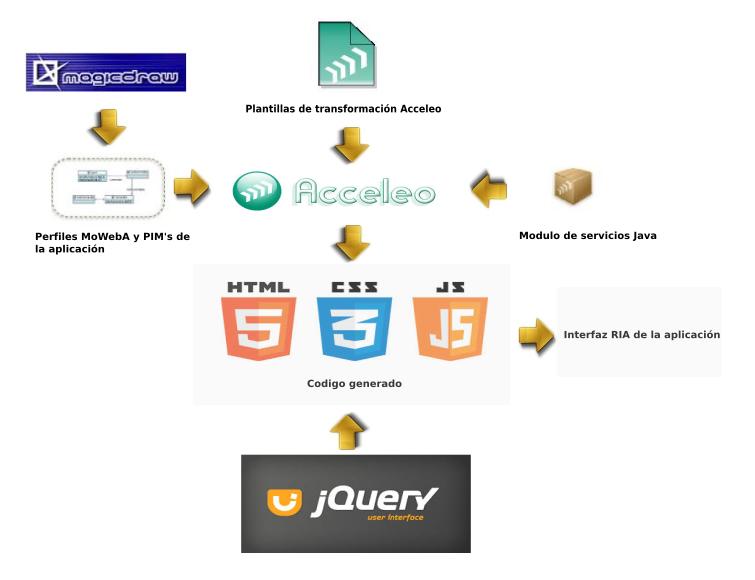


Figura 1 Fases de desarrollo con MoWebA extendido

**3.1 – Extensiones a la capa de presentación de MoWebA.**

La capa de presentación de MoWebA, comprende a los metamodelos de contenido y posicionamiento. El metamodelo de contenido ofrece los distintos elementos para la interfaz de usuario web y que originalmente contienen elementos comunes para la representación de interfaces para la web 1.0. Entre estos elementos tenemos, entradas de texto, entradas de selección (choices y dropboxes) , elementos multimedia como audio y video, entre otros. En el metamodelo de posicionamiento, los distintos elementos del metamodelo de contenido, pueden ser parametrizados con características de posicionamiento que definen las regiones en la cuales las distintas porciones de página serán presentadas al cliente.

El objetivo de este trabajo de tesis es agregar nuevos elementos al metamodelo de contenido, precisamente elementos de la web 2.0, que son los widgets interactivos y la validación en el lado cliente de formularios. Estos nuevos elementos serán modelados en primera instancia y luego traducidos a código por medio de una transformación M2T. Las extensiones se llevarán a cabo en el metamodelo de contenido para obtener la nueva representación de la sintaxis abstracta como así también a su correspondiente perfil que permitirá el modelado de la sintaxis concreta, precisamente los diversos modelos PIM representados con diagramas UML.

**3.1.1 - El metamodelo de contenido de MoWebA.**

El metamodelo de contenido de MoWeba está representado en la Figura 2. En él se presentan los diversos elementos para la interfaz de usuario. En color rojo, se marcan los nuevos elementos propuestos a MoWebA, con la cual es posible desplegar elementos enriquecidos comunes en las RIAS *(widgets)* que fueron presentados en la sección anterior. Se han clasificado primeramente a los distintos elementos de interfaz de usuario simples en, elementos de salida ( text, htmlText y multimedia), elementos de entrada (textImputs, list, richAutoSuggest, richDatePicker y richFieldLiveValidation) y elementos de control (externalLink, anchor y button). Una particularidad ocurre con el elemento simple richToolTip con respecto a la clasificación anterior, ya que este elemento tiene como objetivo enriquecer con mensajes personalizados a cualquiera de los elementos que forman parte de los elementos de entrada, salida y control, por lo tanto se encuentra definido afuera de estas clasificaciones con su respectiva relación de inclusión a cada una de ellas.

Con respecto a los elementos de interfaz compuestos (estos son elementos que pueden contener a uno o más elementos de interfaz simples) tenemos a los form y table. También se encuentran los Panels (richAccordion y el richTabs) que forman parte de las extensiones propuestas a MoWebA

Del metamodelo de contenido presentado, se deriva el perfil UML de contenido para la representación de la sintaxis concreta de MoWebA que se presenta en a continuación en la siguiente sección.

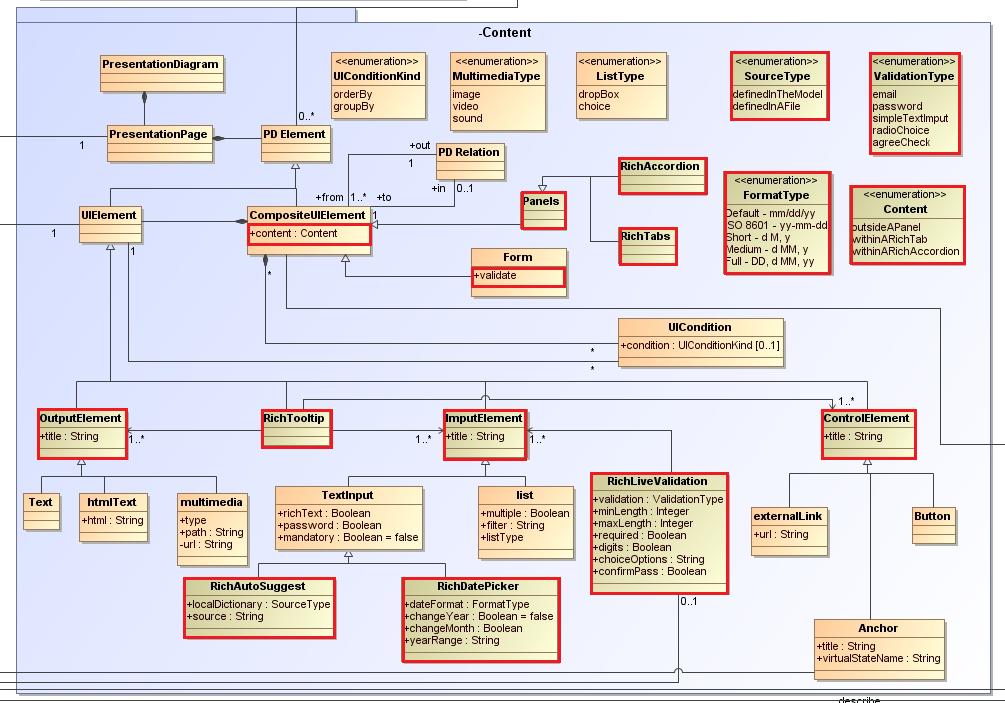
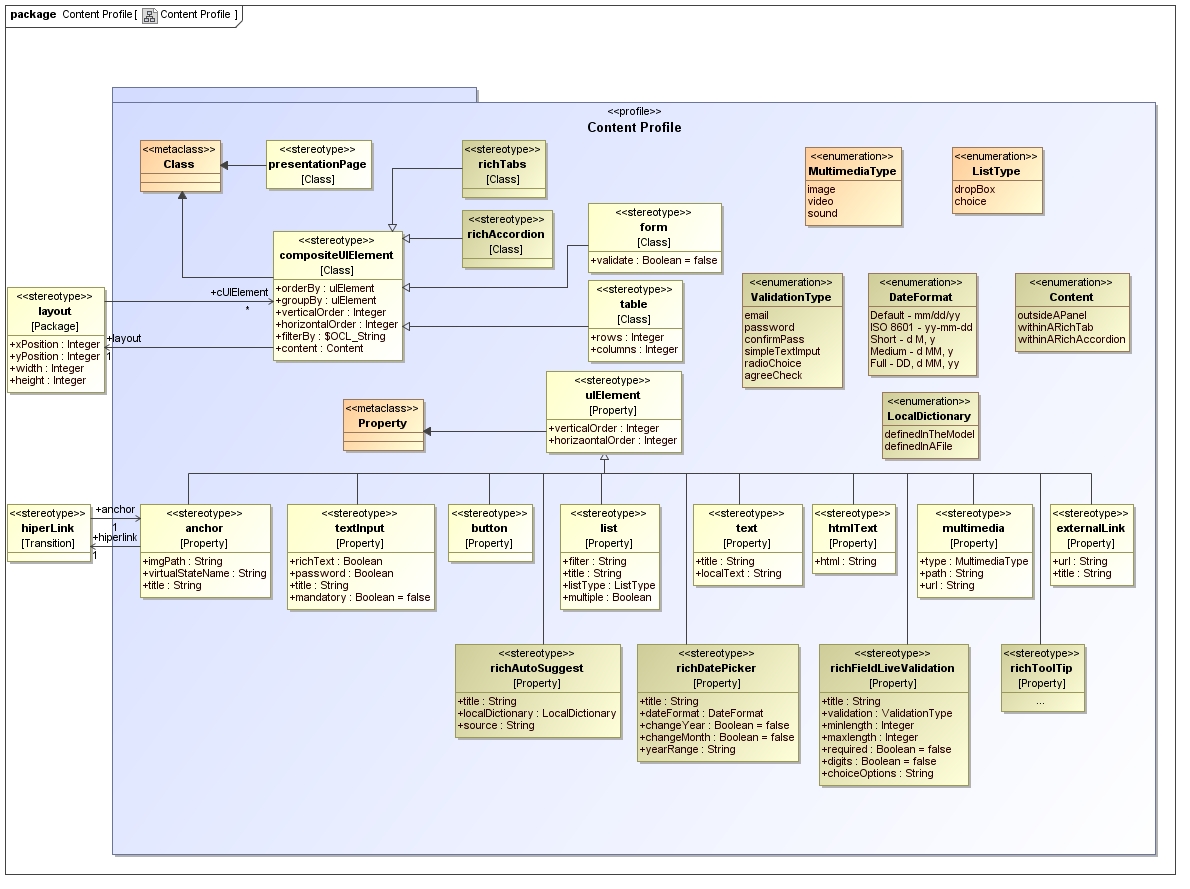


Figura 2 Metamodelo de contenido de MoWebA

**3.3.3 - El perfil para el modelado de contenido en MoWebA (Content profile).**

En la figura 3 se muestra el perfil de contenido para el modelado de los PiM’s de contenido de la aplicación con MoWebA. Como puede apreciare, los elementos resaltados forman parte de la extensión. Los elementos compuestos del perfil (*compositeUIElements*) son representados por medio de Metaclases y estereotipos. Los elementos simples para la representación de contenido son extendidos por medio de Propiedades estereotipadas. A continuación se presentan los nuevos elementos simples y compuestos propuestos a MoWebA.

**Elementos simples**

**RichaAutoSuggest**: este elemento simple, contiene los valores etiquetados *localDictionary* y *source*. El primero corresponde a una enumeración que permite definir al diccionario de sugerencias en un archivo en formato .*xml* o bien definir el listado de sugerencias en el mismo diagrama de clases en un formato de cadenas de sugerencias separado por comas. El valor etiquetado *source* será completado de acorde a lo definido en la enumeración anterior, en el caso de estar definido con el valor *definedInTheModel*, entonces *source* que es del tipo *string*, contendrá un conjunto de palabras correspondiente a las sugerencias separadas entre comas. Si en la enumeración está definido el valor *definedInAFile*, entonces el diccionario de palabras está definido en un archivo xml y en valor etiquetado *source* se colocará la ruta en la cual se encuentra el archivo de palabras.

**RichDatePicker**: este elemento simple contiene a los valores etiquetados *dateFormat*, *changeYear*, *changeMonth* y *yearRange*. El *dateFormat* corresponde a una enumeración que contiene cinco formatos de fecha distintos que son *Default - mm/dd/yy*, *ISO 8601 - yy-mm-dd*, *Short - d M, y*, *Medium - d MM, y* y el *Full - DD, d MM, yy*. El valor etiquetado *changeYear* es un valor booleano que indica la ausencia o presencia de un rango de años desplegable en una lista será parte del *datePicker.* Por omisión, si *changeYear* está configurado en verdadero, se mostrará en el *datePIcker*, una lista desplegable presentando los diez años anteriores a partir de la fecha actual. También es posible asignar al valor etiquetado *yearRange* un rango de años en el formato yyyy:yyyy, por ejemplo 1970:2015, que resulta ideal para la selección de fechas pasadas, como el año de nacimiento. Por último, el valor etiquetado booleano *changeMonth*  permite desplegar una lista con todos los meses del año para una rápida selección.

**RichToolTip**: este elemento simple no cuenta con valores etiquetados. Al definirse este estereotipo en conjunción con algunos de los elementos simples de entrada, salida o de control, implica que un mensaje emergente será desplegado cuando el puntero del mouse se posicione sobre el elemento. Cada uno de los elementos de entrada, salida y control posee el valor etiquetado *title*, que corresponde al mensaje que será desplegado.

**RichFieldLiveValidation:** este elemento simple permite llevar a cabo validaciones locales a diversos elementos pertenecientes a en un formulario. El valor etiquetado *validation* corresponde a una enumeración que contiene todas las validaciones que son posibles de llevar a cabo. Estas validaciones pueden ser del tipo email, *password*, *simpeTextImput*, *radioChoice* y *agreeCheck*. Para las validaciones del tipo *email*, *password* y *simpleTextImput* es posible establecer la cantidad mínima de caracteres en un campo de entrada estableciendo el valor etiquetado *minLength*. Similarmente el valor etiquetado *maxLength* permite establecer la cantidad máxima de caracteres que es posible ingresar en un campo de entrada. Para el *simpleTextImput,*  el valor etiquetado *digits*, establece que el campo de entrada debe tener estrictamente valores del cero al nueve. Las validaciones email, password y simpleTextImput, pueden configurarse como mandatorios, lo que implica que no pueden quedar vacios, estableciendo el valor etiquetado booleano *required* como verdadero.

Al seleccionar el tipo de validación *password*, es posible establecer el valor booleano del valor etiquetado *confirmPass* en verdadero, para el caso en el que se necesite crear otro campo de entrada para la confirmación de contraseña. Por otro lado, la validación del tipo *radioChoice*, permite establecer de un listado de opciones de selección, un valor mandatorio, que es establecido en el valor etiquetado *choiceOptions* , como cadenas separadas por comas. De todas las opciones existentes en *choiceOptions*, una de ellas se establece con la palabra *Required*, separada con un espacio en blanco para indicar que esa opción es mandatoria y debe seleccionarse de manera obligatoria. Finalmente, el tipo de validación *agreeCheck*, una vez seleccionado, despliega un cuadro de selección del tipo *radio*, el cual es mandatorio y debe ser seleccionado en un formulario para que este pueda ser validado. Este campo es idóneo para acuerdos de conformidad.

**Elementos compuestos**

**Form:** este elemento compuesto que es una especialización de un *compositeUIElement*, ha sido extendido con el valor etiquetado booleano *valídate*,que indica si el formulario en cuestión tendrá algún tipo de validación de campo. Por lo tanto *valídate*, debe estar establecido en verdadero para que el RichFieldValidation pueda llevarse a cabo.

**RichAccordion:** este elemento compuesto, permite encapsular a varios elementos simples y compuestos dentro de paneles colapsibles. Dentro de los elementos compuestos que pueden ser desplegados en cada una de los paneles que lo componen, se encuentran los compositeUIElements, *table* y los *form*. Para establecer que un *form* se encuentra dentro de un panel del *richAccordion*, se debe seleccionar de la enumeración *content* que se encuentra dentro de un *compositeUIElement*, el valor *withinARichAccordion*. El valor por omisión establecido para la enumeración *content* es *independent* que especifica que el contenido del panel no es un *form*  ni una *table*.

3.3.\* El metamodelo de posicionamiento de contenido en MoWeba.

3.3.\* El pérfil de posicionamiento de contenido en MoWebA (Layout profile)

3\* Modelado de interfaces de usuario con MoWebA.

3\* Conclusiones

1. **No Magic:** <http://www.nomagic.com/products/magicdraw.html> 2015 [↑](#footnote-ref-1)
2. **Eclipse Modelling Framwwork:** <https://www.eclipse.org/modeling/emf> 2015 [↑](#footnote-ref-2)
3. **Acceleo:** <https://eclipse.org/acceleo> 2015 [↑](#footnote-ref-3)