Capitulo 2

Marco teórico y trabajos relacionados

**2.1 - Las Rich Internet Applications (RIAs)**

Desde el lanzamiento oficial del primer sitio web en 1991 por Tim Berners Lee hasta hoy en día, las aplicaciones web que forman parte de la red de redes internet, han evolucionado de la web 1.0, en la que los usuarios obtenían información estática representada en documentos hipertextuales, a la web 2.0, en la cual la información de las páginas es generada de manera dinámica y en la que se combinan, no solamente información textual, sino también, características multimedia en las interfaces (audio, video streaming, widgets interactivos, entre otros). De igual forma, la evolución en la web también vino acompañada de cambios tecnológicos en los diferentes navegadores web y en los distintos protocolos de comunicación entre las aplicaciones cliente y servidor.

Muchos de estos avances se dieron, debido a las limitaciones de las aplicaciones web tradicionales en cuanto a la usabilidad e interactividad que ofrecen sus interfaces de usuario. Esto se debe en gran medida, a la comunicación síncrona existente entre el cliente y el servidor en la que, por cada acceso a un enlace, el cliente debe esperar la respuesta del servidor que una vez obtenida, debe recargar la página completamente, quedando ocioso la mayor parte del tiempo [<garrett>]. He ahí que surgen, como alternativa, las denominadas Aplicaciones de Internet Enriquecidas (*Rich Internet Applications* - RIAs) con la idea de mejorar las aplicaciones web tradicionales, agregando nuevas características que se encuentran presentes en las aplicaciones de escritorio. El término fue introducido en marzo de 2002 por la empresa Macromedia (actualmente Adobe) que en ese entonces abordaba las limitaciones en cuanto a la riqueza de las interfaces, medios y contenidos de las aplicaciones [].

Dado que las RIAs poseen numerosas características innovadoras, es difícil ofrecer una definición formal que englobe todos sus atributos. Diversos autores la han enmarcado en un contexto particular y todas las citadas aquí resultan valederas. A continuación se presentan algunas descripciones interesantes de algunos autores:

RIAs“Las RIAs, emulan características de las aplicaciones de escritorio, mejorando la experiencia de los usuarios con nuevos efectos visuales, dándose principal realce a las características multimedia. El intercambio de los datos puede llevarse a cabo por medio de una comunicación asíncrona, de tal forma que el cliente permanece receptivo a eventos, mientras que continuamente recalcula o actualiza partes de la interfaz de usuario. Las RIAs se caracterizan por poseer una variedad de controles interactivos de operación (widgets), y por dar la posibilidad de utilizar la aplicación con o sin conexión al servidor (uso offline de la aplicación), y también por ofrecer un uso transparente de las capacidades del cliente, del servidor y de la conexión de red” [].

RIAs“En las RIAs, las aplicaciones se cargan de manera completa en el cliente, desde el inicio, realizándose la comunicación con el servidor solamente en caso de que sea necesario actualizar los datos desde una base de datos o bien desde un archivo externo. La navegabilidad de las aplicaciones web mejora de manera substancial, debido a que se evitan las recargas innecesarias de toda la página, actualizando solamente las porciones de ésta que son relevantes. Con esto se minimiza la cantidad de información que se transmite por la red a la par de mejorar la performance de la aplicación[[1]](#footnote-1)”.

“Las RIAs son aplicaciones web que exhiben widgets, comportamientos y características que están presentes en las aplicaciones de escritorio. También, poseen una mayor capacidad de respuesta, son más seguras y presentan una interfaz más avanzada con respecto a las aplicaciones del modelo Web 1.0. Sus características principales incluyen, el paradigma de página única, un avanzado esquema de comunicación (con la inclusión de tecnologías push y comunicación asíncrona entre el cliente y el servidor y un manejo optimizado de los datos, reduciendo las solicitudes al servidor) y finalmente, la inclusión de un motor en el cliente (en la forma de máquina virtual o extensiones (plug-ins) en el navegador que administra la disposición gráfica de los elementos y la mayoría de las interacciones locales” [].

De las definiciones anteriores, puede notarse el hecho de que en las RIAs se busca que las aplicaciones tiendan a comportarse lo más similarmente a las aplicaciones de escritorio, evitando el refrescado excesivo de las páginas, más precisamente, permitiendo el refrescado parcial de ciertas zonas que son relevantes o que necesitan actualizarse dado un cambio de estado o debido a una actualización en la fuente de datos. También puede resaltarse que el refrescado parcial de las páginas es posible, debido a la comunicación asíncrona existente entre el cliente y el servidor, que mejora el intercambio de los datos que se transmiten por la red. Los *widgets* o elementos de interfaz interactivos son otra de las características que se encuentran presentes en las RIAs y que ofrecen una mayor riqueza a las interfaces de usuario, como así también los elementos multimedia como audio y video *streaming*. El lado del cliente en las aplicaciones RIAs, funciona de una manera más independiente del lado servidor y en ocasiones es posible utilizar las aplicaciones de manera *offline*. Finalmente, todos estos objetivos son alcanzados, agregando un motor en forma de plug-in en el cliente para la administración de las comunicaciones y para la gestión de las interacciones locales.

La definición propuesta por [<martinez-ruiz2010>], engloba la mayor cantidad de características que son comunes en las definiciones anteriormente presentadas y por ende, resulta ser la más completa. En este trabajo de fin de carrera, se tendrán en cuenta características RIAs de presentes en esta definición.

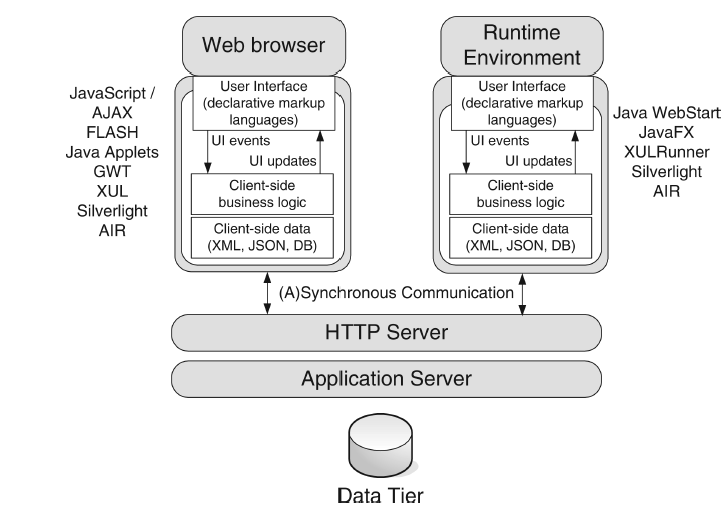
RIAs

Figura 1 Arquitectura RIA

En [<fraternali2010>] se describe la arquitectura RIA de manera general como se muestra en la Figura 1. El sistema está compuesto de una aplicación web servidor y un conjunto de aplicaciones de usuario corriendo en las máquinas clientes. Estas aplicaciones son implementadas o bien, en un navegador web utilizando una variedad de tecnologías como *Javascript*, animaciones Flash, código interpretado en plug-ins y Java applets, o fuera del navegador en términos de binarios descargados desde la web e interpretados en un ambiente especifico de ejecución, por ejemplo, utilizando tecnologías como Java Web Start[[2]](#footnote-2) y Adobe AIR[[3]](#footnote-3).

Las RIAs hoy en día juegan un papel preponderante. Según un estudio de mercado patrocinado por la empresa Adobe en 2007, dada las mejoras con respecto a la interfaz de usuario y al comportamiento de las aplicaciones, las RIAs han conseguido incrementar la productividad y la satisfacción de los usuarios que llevan a cabo operaciones en internet, debido, en gran medida, a la nueva experiencia de interacción que ofrecen [<rogowskimarch122007>]. Un estudio similar [<kiewe2011>], presenta datos cuantitavos con referencia a como una aplicación con características de las RIAs, puede mejorar las utilidades y disminuir los costes de desarrollo en una compañía

**2.2 Características de las RIAs**

A continuación se presentan las características más distintivas de las RIAs con respecto a las aplicaciones Web tradicionales, que fueron presentadas en [] y []. Además, por cada una de estas características, se muestra un cuadro en donde se reflejan las ventajas y desventajas de cada característica.

**2.2.1 Almacenamiento de los datos**

En las RIAs, es posible almacenar datos en el lado cliente, con diferentes niveles de persistencia (temporalmente, mientras la aplicación está en ejecución, o persistentemente). También, los datos pueden distribuirse entre ambos pares, cliente y servidor.

En la Tabla 1 se presentan algunas ventajas y desventajas de llevar a cabo una distribución de datos entre el cliente y el servidor

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| * Es posible utilizar la aplicación sin necesidad de establecer una conexión con el servidor (uso offline). * Es factible la preparación y validación de los datos en el lado del cliente. | * Existe la posibilidad de replicación de datos en ambos pares. * Puede llegar a tornarse complejo establecer políticas para la asignación (distribución) de los datos. |

Tabla 1 Ventajas y desventajas de la distribución de los datos en el cliente

## 2 Lógica de negocio

En una aplicación web tradicional, la extracción de datos y la lógica de negocio se computan en el servidor. En las RIAs es posible llevar a cabo operaciones complejas directamente en el cliente (por ejemplo: efectuar navegaciones, filtrados y ordenamiento de los datos con múltiples criterios; operaciones de dominio específico para sistemas complejos; y validación local de datos). También es factible distribuir la lógica de negocios entre el cliente y el servidor para, por ejemplo, validar algunos campos de un formulario en el cliente y otros en el servidor. Por lo tanto, el diseño conceptual debe responder a la decisión de cómo asignar la computación tanto de las páginas como así también de los componentes de éstas [].

En la Tabla 2 se presentan algunas ventajas y desventajas de una computación distribuida de páginas entre el cliente y el servidor:

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| * Validación de datos en vivo. * La posibilidad de utilizar la aplicación sin necesidad de una conexión (uso offline de la aplicación). | * La lógica de la aplicación en conjunto se complica. * Puede ser confuso definir si una funcionalidad en particular debe computarse en el cliente o en el servidor. * El restablecimiento de la comunicación entre el cliente y el servidor luego de la utilización offline de la aplicación es una acción propensa a errores. |

Tabla 2 Ventajas y desventajas de una computación distribuida de páginas

## 2.2.3 Comunicación entre el cliente y el servidor

Con las RIAs se crean mecanismos para reducir al mínimo la transferencia de los datos migrando las capas de interacción y presentación del servidor al cliente. Las RIAs soportan comunicaciones asíncronas entre el cliente y el servidor para la distribución de objetos de dominio, datos y la computación.

Figura 2 Comunicación síncrona versus comunicación asíncrona.

En la Figura 2 se puede ver una comparativa con respecto a la comunicación entre los pares cliente y servidor, de las aplicaciones de la web 1.0 y las actuales basadas en RIAs. En las aplicaciones web tradicionales, los datos residen en el servidor, y los clientes a medida que necesitan alguna actualización de página, llevan a cabo la solicitud de actualización por medio de la activación de algún enlace navegacional (que puede ser algún hipervínculo, botón de solicitud de registro de usuario, etc.). Seguidamente, en respuesta a la solicitud del cliente, el servidor devuelve la página con la actualización correspondiente. La comunicación es llevaba a cabo de una manera síncrona, en donde un evento del usuario es necesariamente el elemento disparador de una solicitud al servidor. Con las RIAs, un motor instalado en el cliente es el encargado de gestionar las solicitudes de transferencia de los datos al servidor y de gestionar los cambios en la disposición de los distintos elementos en la interfaz del usuario. Las solicitudes al servidor al ser gestionadas asíncronamente por un motor (o plugin) instalado en el cliente, permite a la aplicación llevar a cabo diversas acciones en paralelo, como por ejemplo, actualizar distintas porciones de una misma página en un momento dado. En la Tabla 3 se muestran algunas ventajas y desventajas de una comunicación asíncrona entre el cliente y el servidor:

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| * Es posible llevar a cabo el refrescado parcial de las páginas, abarcando solamente las zonas de interés. * Se mejora la interacción del usuario con la aplicación. * *Server-push[[4]](#footnote-4)*. | * Se incrementan los esfuerzos de desarrollo de las aplicaciones. * El testing de las aplicaciones se vuelve más complejo. |

Tabla 3 Ventajas y desventajas de una comunicación asíncrona en entre el cliente y el servidor

## 2.2.4 Presentaciones enriquecidas

Las interfaces de usuario ofrecen una mayor riqueza con el manejo de eventos en el lado del cliente y los *widgets* interactivos que son micro programas empotrados dentro de las páginas web y administradas por un motor de *widgets* (que podría ser un plug-in instalado en el navegador). Los *widgets* presentan funciones bien específicas que por lo común resultan de utilidad a los usuarios tales como: presentar el estado del tiempo, la hora de diversos países, la cotización de las monedas extranjeras, calculadoras, entre otros. Los elementos multimedia dentro de las páginas como la intrusión de audio y video de alta calidad, a la par de animaciones también son características típicas de las RIAs, como así también, la capacidad de arrastrar y soltar elementos dentro de la interfaz, las auto-sugerencias de datos a medida que se va escribiendo un patrón en un campo y el refrescado automático de las páginas (o porciones de esta), son otras de las características interesantes que pueden encontrarse.

En la Tabla 4 se presentan algunas ventajas y desventajas de un comportamiento más sofisticado en la interfaz de usuario:

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| * Funcionamiento como una aplicación de una sola página, evitando de esta forma perderse en la navegación del sitio web. * Se presenta al usuario una interfaz mucho más enriquecida y reactiva a eventos. | * Pueden presentarse problemas de rendimiento. * Es posible que se tengan incompatibilidades en el navegador web. |

Tabla 4 Ventajas y desventajas de un comportamiento más sofisticado en la interfaz de usuario

De todas las características subyacentes a las RIAs anteriormente descriptas, las presentaciones enriquecidas (que abarcan el manejo de eventos en el lado del cliente, los *widgets* interactivos, el paradigma de una sola página y el contenido multimedia) son las que representan el *look and feel* final.

## 2.3 Tecnologías para la implementación de las RIAs

Actualmente, las capacidades de las RIAs se pueden implementar en diferentes tecnologías cliente que pueden clasificarse en tres categorías principales, de acuerdo con el entorno de ejecución:

• **Basadas en *Javascript*:** la lógica del lado cliente está implementada en *Javascript* (el enfoque también es conocido como "Ajax", Asynchronous *Javascript* y XML [<garrett>] y las interfaces de usuario se basan en una combinación de HTML y CSS.

Figura 3 A la izquierda el Modelo de aplicación web clásico. A la derecha, el modelo de aplicación web Ajax.

La principal ventaja de este enfoque es que se basa en el *Javascript* incorporado en el navegador y soporta los estándares de W3C. En la Figura 3 se presenta el modelo de aplicación Ajax en comparación con el modelo de aplicación web clásico. Como puede apreciarse para el caso del modelo Ajax, el motor Ajax es el encargado de orquestar la disposición de los elementos en la interfaz de usuario en el lado del cliente por medio de HTML y CSS y la lógica de este por medio de *Javascript*. Por lo general la comunicación con el lado servidor es llevado a cabo por medio de solicitudes HTTP o HTTPS y la respuesta del lado servidor se da por medio de lenguajes de marcado como XML o JSON. Los principales inconvenientes son el soporte multimedia insuficiente, limitaciones en las cajas de arena (sandboxes) del navegador, por ejemplo, el acceso al sistema de archivos o almacenamiento persistente, y la inconsistencia en el comportamiento del navegador. Debido a este último aspecto, un gran número de bibliotecas se han propuesto para permitir a los desarrolladores abstraerse de las idiosincrasias del navegador.

• **Basadas en plug-ins:** en donde la representación avanzada y el procesamiento de eventos se encomienda a los plug-ins del navegador por medio de la interpretación de lenguajes específicos de scripting, XML o archivos multimedia. Una ventaja de los plug-ins es que generalmente soportan la interacción multimedia de forma nativa, permitiendo la persistencia en el lado del cliente y ofrecen un mejor desempeño que *Javascript* interpretado. Algunos plug-ins vienen ya instalados en los navegadores, pero otros requieren de la intervención del usuario administrativo. Sin embargo, en algunos casos no proveen el acceso a servicios del sistema operativo (por ejemplo, al sistema de archivos).

• **Basadas en ambientes en tiempos de ejecución:** las aplicaciones se descargan de la Web, pero se ejecutan fuera del navegador, utilizando un ambiente de escritorio en tiempo de ejecución. Estas soluciones ofrecen lo máximo en términos de capacidades de cliente y el uso off-line, con pleno acceso al sistema operativo subyacente. Sin embargo, se basan en un ambiente especializado en tiempo de ejecución, lo que obliga a los usuarios a que lo instalen (y podría no estar disponible en todas las plataformas, como por ejemplo en teléfonos móviles). Muchas de las tecnologías RIAs se pueden utilizar para desarrollar aplicaciones de este tipo. En la Tabla 5 puede apreciarse las capacidades y las limitaciones de cada una de las tecnologías con respecto a las características descritas en la sección 2.2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tecnología cliente vs**  **Características de las RIAs** | **Presentaciones enriquecidas** | **Almacenamiento de los datos en el cliente** | **Lógica de negocio en el cliente (o distribuida entre el cliente y el servidor)** | Comunicación entre el cliente y servidor |
| **Basados en JavaScript** | Limitada: sin multimedia | Limitada: no hay persistencia de datos | Si | Si |
| **Basados en Plug-ins** | Si | Si : con plug-ins adicionales | Si | Si |
| **Basados en ambientes en tiempo de ejecución** | Si | Si | Si | Si |

Tabla 5 Tecnología cliente vs Características de las RIAs

Entre las tecnologías citadas, la práctica más popular de desarrollo RIA ve la adopción de las basadas en *Javascript* (es decir, AJAX) como la opción más común. Las limitaciones actuales se suelen resolver utilizando extensiones de Flash para el procesamiento de vídeo (y Google Gears o Flash Shared Objects cuando en el lado del cliente el almacenamiento persistente es necesario). Las razones principales que explican esta tendencia son en el hecho de que: AJAX es asumido por muchos desarrolladores como el conjunto más abierto y estándar de tecnologías (y más cerca de la especificación HTML5) y que no requiere acciones administrativas (por ejemplo, la instalación de software) de los usuarios. También se pueden combinar fácilmente con plug-ins que se construyen para superar sus limitaciones.

**2.4 Herramientas para el desarrollo de las RIAs**

En la sección anterior se presentaron las diferentes tecnologías para el desarrollo e implementación de las RIAs. Para este trabajo de fin de carrera, se optó por herramientas y frameworks de desarrollo de uso abierto y que son de amplia utilización en la comunidad web. Se ha señalado el hecho de que las implementaciones basadas en *Javascript* o librerías Ajax son las más utilizadas en la actualidad, debido a que utiliza tecnologías de uso abierto estandarizado como lo son; *Javascript*, HTML y CSS. También esta forma de implementar las RIAs es el más cercano al estándar HTML5. RIAs

Son numerosas las librerías *Javascript* existentes en la actualidad. 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. En la Figura 4 se puede apreciar algunas librerías *Javascript* de uso extendido



Figura 4 Algunas librerías *Javascript* de uso común

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 RIAs 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. Los widgets son microprogramas que cumplen una función predeterminada y que a la vez sus propiedades pueden ser modificadas para expresar comportamientos personalizados por el usuario. Una vez modificada las propiedades del widget, éste es introducido dentro de la aplicación para cumplir una función en particular.

**2.4.1 Widgets más utilizados**

En estudios llevados a cabo en 2009[[5]](#footnote-5) y 2010[[6]](#footnote-6) , se presenta un análisis de los *widgets* más utilizados por las aplicaciones web. En la figura 5 se presentan de manera sumarizada tales *widgets*. De todos los *widgets* identificados, suena interesante determinar cuáles son los más utilizados hoy en día, por lo que se ha realizado un análisis de portales web populares (Facebook, Gmail, Youtube y Amazon) para determinar que *widgets* son comunes en estos sitios. El análisis determinó que los cuatro portales utilizan los siguientes *widgets*;

***Accordion:*** Muestra paneles de contenido plegable para presentar la información en una cantidad limitada de espacio.

***Tabs:*** Ofrece una sola área de contenido con múltiples paneles, cada uno asociado con una cabecera en una lista.

***Autocomplete:*** Permite a los usuarios a seleccionar y encontrar rápidamente de una lista previamente poblada de valores a medida que se escribe, mejorando la búsqueda y el filtrado.

***Tooltip:*** Ofrece mensajes personalizados de sugerencia sobre los elementos de interfaz, reemplazando los mensajes nativos.

Figura 5 Elementos de interfaz de usuario enriquecidos (*widgets*) más utilizados.

***Datepicker:*** Permite seleccione una fecha de un calendario emergente o *inline*.

***Live validation:*** Ofrece validaciones en vivo de los campos en los formularios.

**2.4.2 Las librerías *Javascript* *jQuery*UI[[7]](#footnote-7) y *jQuery Validation Plugin*[[8]](#footnote-8)**

Estudios de mercado recientes han presentado a *jQuery*[[9]](#footnote-9) como la librería *Javascript* más utilizada a nivel global[[10]](#footnote-10). *jQuery* es de uso abierto y ha tenido un crecimiento notable en términos de evolución hasta hoy en día desde su aparición en el año 2005. Esta librería a la par de presentar una manera robusta y confiable para desarrollar código *Javascript*, en su versión *jQueryUI* se encuentran numerosos *widgets* interactivos idóneos para las diversas interfaces de usuario, como para aplicaciones web y móviles.

De todas características citadas en la sección anterior, solamente el *Live validation* no es soportado por *jQueryUI* de manera nativa. Sin embargo con *jQuery Validation Plugin*, que es una extensión basada en *jQuery*, es factible llevar a cabo validaciones enriquecidas sobre los campos de un formulario de una manera bastante intuitiva.

Con *jQueryUI* y *JQuery Validation Plugin*, es posible dar cobertura a todos los widgets que serán tenidos en cuenta en este trabajo de fin de carrera.

**2.5 Model driven software engineering (MDSE)**

Los modelos son de suma importancia para entender y compartir conocimiento acerca de un software complejo. MDSE es concebida como una herramienta para convertir este hecho, en una manera concreta de trabajar y pensar, transformando los modelos en elementos fundamentales para todo el ciclo de desarrollo en la ingeniería de software []. En MDSE, los conceptos principales son los modelos y las transformaciones (esto es, manipulaciones y/o operaciones sobre los modelos).

MDSE , tiene como objetivo llevar a cabo el desarrollo de artefactos de software utilizando a los modelos y a las transformaciones sobre estos, como piezas clave para el logro de tal objetivo. Hoy en día se ha dado un valor extra a los modelos, debido a que no solamente sirven para mantener una mejor comunicación entre los desarrolladores y las partes interesadas en un sistema en particular (*stakeholders*) o bien para mantener los sistemas debidamente documentados, sino también, estos modelos pueden contener la suficiente expresividad y riqueza como para representar información que posteriormente puede transformase y obtener así, el software deseado.

Un concepto clave en el contexto MDSE es el de metamodelo. Con el metamodelo es posible definir la sintaxis abstracta de un lenguaje de modelado. Análogamente a las gramáticas que sirven para definir a un lenguaje de programación, el metamodelo permite representar a todos los modelos posibles que forman parte del lenguaje de modelado.

**2.5.1 *Model Driven Development* (MDD) *y Model Driven Architecture* (MDA)**

En MDSE, es posible adoptar un enfoque MDDpara el ciclo de desarrollo de una aplicaciónMDA MDD es un paradigma de desarrollo que utiliza a los modelos como artefactos primarios en el proceso de desarrollo. Usualmente en MDD la implementación es generada de manera automática o semiautomática a partir de los modelos. Por otra parte, MDA es un estándar impulsado por el consorcio OMG (*Object Management Group*) que contiene en si misma a varios estándares de facto, tales como UML *(Unified Modeling Language*) , OCL (*Object Constraint Language*), MOF(*Meta Object Facility*), *QVT*(*Query View Transformation*), entre otros; con la meta presente de promover el desarrollo de software para diversos dominios de aplicación, como los son las aplicaciones para el ámbito de las finanzas , las telecomunicaciones , las aplicaciones aeroespaciales, las embedidas, etc. *MDA* es un subconjunto de MDD, que propone estándares para cada paso en el proceso de desarrollo de las aplicaciones. Utiliza un esquema de arquitectura dividida en capas como puede apreciarse en la Figura 6. Los meta-metamodelos (M3) se expresan por medio MOF o ECORE para el *Eclipse Modelling Framework* (*EMF*) y los metamodelos (M2) de la aplicación se expresan por medio de un *General Purpose Modelling Lenguage (GPML)* que por lo general es UML, que cuenta con diversos modelos para representar los comportamientos (estáticos y dinámicos) de una aplicación en particular. La capa M2 describe los conceptos utilizados en M1 para la definición de los modelos. Finalmente el objeto del mundo real, en este caso un video, se representa en M0.



Figura 6 Objetos del mundo real (M0), modelos (M1), metamodelos (M2) y meta-metamodelos (M3)

Las fases de desarrollo con el enfoque MDA se presentan en la Figura 7. La fase *Computation independent Model* (CIM) corresponde a los documentos, modelos o diagramas utilizados para la toma de requerimientos en una aplicación en particular, independientemente de cómo han sido implementados. Representan al punto de vista del negocio de la solución.



Figura 7 Cadena de transformaciones en MDA

Los CIM son los puntos de entrada de los *Platform independent model* (PIM). La transformación CIM a PIM se da por lo general por medio de un mapeo manual. La fase PIM contempla la representación del sistema por medio de modelos que son independientes de la tecnología de implementación. Los PIM pueden ser transformados a un *Platform specific model*(PSM) a través de una transformación modelo a modelo (M2M), y en muchos casos soportados por un lenguajes de transformación como QVT o ATL. Los PSM son modelos enriquecidos con detalles de una plataforma destino en particular. Finalmente estos PSM pueden ser transformados a código fuente por medio de una transformación de modelo a texto (M2T) apoyándose por herramientas de trasnformación M2T como MOFScript , Acceleo u JET (*Java Emmitter Template*).

**2.6 Principales enfoques de desarrollo web basado en modelos para las RIAs**

En [] y [], se identifica la necesidad de metodologías sistemáticas para el desarrollo de las RIAs y se llevan a cabo estudios presentando las diversas metodologías web existentes para ese fin. El estudio más exhaustivo y reciente de comparativas se presenta en [] en donde se clasifican las metodologías en las siguientes categorías:

1. Contribución a la investigación proveniente de la comunidad de ingeniería web, derivada de la evolución de los enfoques dirigidos por modelos concebidos para el diseño y desarrollo de aplicaciones web tradicionales en las que se incluyen a WebML-RIA[], OOHDM -RIA[], OOH4RIA[] , (UWE-R[]Patrones con UWE[] y UWE+RUX []).
2. Enfoques de desarrollo sistemáticos provenientes de la comunidad *Human Computer Interactión (HCI),* en donde el diseño RIA es el foco de la metodología RUX[<trigueros2007>] y puede ser logrado de igual forma con el enfoque más general UsiXML[] [].
3. Enfoques que combinan HCI y técnicas de ingeniería web: espacios interactivos con UML presentado en [] y OOWS for RIA [].
4. Propuestas recientes de los vendedores de herramientas comerciales que adoptan MDD entre ellos WebRatio, Mendix, Novulo, RUX-Tool y Thinkwise.

Con respecto al contexto en el cual se analizan las metodologías web anteriores, una de las consideraciones a tener en cuenta, es que las mismas adopten estándares (por ejemplo, UML). También se busca que las metodologías en cuestión, sean de uso abierto para la comunidad de desarrolladores y no propietarias. He ahí que a continuación se describirán brevemente las metodologías basadas en UML que son OOH4RIA, UWE-R, Patrones con UWE, UWE combinado con la herramienta RUX y los patrones de interacción con OOWS. El resto de las metodologías se presentarán en el cuadro comparativo con sus respectivos alcances para las RIAs. La categoría d) del estudio mencionado no se considera debido a que son propuestas cerradas basadas en herramientas comerciales.

**a1) Extensión a OO-H (OOH4RIA)**

OO-H (*Object Oriented Hypermedia*) [] es una metodología orientada a objetos para la web tradicional, basada parcialmente en estándares (XML, UML y OCL). Se propone a esta metodología en su forma original, un enfoque MDD para especificar una aplicación RIA, por medio de una extensión, agregando nuevos modelos para la presentación. La Figura 8 muestra una representación del proceso MDD con las definiciones de modelos y transformaciones que permiten obtener la implementación correspondiente a las RIAs, como así también, los actores que participan en el ciclo de desarrollo.

OOH-RIA, propone un metamodelo de presentación definido con abstracciones de bajo nivel, donde los elementos principales, son representados por los *widgets* proveídos por una plataforma específica,en este caso *Google Web Toolkit* (GWT[[11]](#footnote-11): este metamodelo permite la especificación de los aspectos estructurales de las RIAs. Los *widgets* pueden ser combinados, extendidos, adaptados y enlazados a otros modelos. Se genera el código de la aplicación tanto para el lado cliente como par el lado servidor.

Figura 8 Representación del proceso MDD para OOH-RIA

**a2) Extensiónes RIA a UWE (UWE-R)**

El enfoque UWE (*UML-based Web Engineering)* [][<koch2001>] es una metodología orientada a objetos que tiene la particularidad distintiva de que está basada totalmente en UML. Está definida en la forma de perfil y de por sí, es una extensión al metamodelo UML. UWE-R es una ligera extensión de UWE para RIA, que abarca las capas de navegación, proceso y presentación. Por lo tanto, los nuevos elementos de modelado están definidos heredando la estructura definida y el comportamiento de los elementos UWE.

Con respecto a las extensiones a la navegación, se extienden las metaclases *Nodo* y *Enlace*. Como puede verse en la Figura 9, la metaclase *Nodo* es extendida agregando la metaclase *RichNavigationClass*, que a diferencia de UWE clásico que se basa en la navegación hipertextual principalmente, esta nueva metaclase, podría estar contenida dentro de un objeto *Flash* o un *Java Applet*. La metaclase *Enlace* se extiende agregando la metaclase *RichNavigationLInk*, que tiene como finalidad modelar la interacción entre la aplicación cliente y servidor; especificando si se trata de una comunicación síncrona o asíncrona. En el caso de ser asíncrona, la respuesta es un *callback*.



Figura 9 Extensiones a las metaclases Nodo y Enlace.

A nivel de presentación, se agregan algunas metaclases para expresar la riqueza de las RIAs con respecto al aspecto de la interfaz de usuario, como puede verse en la Figura 5.



Figura 10 Extensiones al metamodelo de Presentación en UWE-R.

Por último, se llevan a cabo extensiones con respecto al proceso (o la lógica de la aplicación), en la cual, se puede modelar los procesos que pueden realizarse en los lados servidor y cliente respectivamente. También por medio de la metaclase *Autonomous Action,* se pueden modelar las acciones que deben llevarse a cabo automáticamente en la aplicación, sin la interacción del usuario; por ejemplo en el caso que expire un temporizador; se dispara automáticamente alguna acción.

**a3) UWE combinada con la herramienta RUX**

La metodología UWE también puede combinarse con el método RUX []. La metodología RUX es un enfoque dirigido por modelos para el enriquecimiento de las interfaces de usuario. Puede ser utilizada en el tope de muchas metodologías de modelado web. En este enfoque, UWE es utilizado para especificar el contenido, navegación y proceso de negocio de una aplicación Web y la metodología RUX se emplea sobre estos modelos para adicionar capacidades enriquecidas a la interfaz de usuario. La propuesta presentada por busca construir el puente entre ambos enfoques, definiendo reglas de transformación entre sus respectivos metamodelos [13]. En otros términos, se extiende las reglas de generación de UWE de manera a obtener la conexión con la metodología RUX automáticamente.

La metodología RUX presenta 3 niveles de interfaces, proveyendo de esta forma una cadena de refinamientos. La interfaz abstracta provee de una representación común a todos los dispositivos y plataformas de desarrollo RIAs, sin ningún tipo de dependencia espacial, de estética ni de comportamiento. La interfaz concreta es independiente de la plataforma, pero específica para un dispositivo o grupo de dispositivos. Está dividida en 3 niveles de presentación, espacial, temporal y presentación interactiva. En la presentación espacial; los modeladores simplemente necesitan refinar esta agrupación, especificar el arreglo espacial de los componentes y definir sus dimensiones y la estética. La presentación temporal permite la especificación del comportamiento lo cual requiere una sincronización temporal. (Por ejemplo Animaciones). La presentación interactiva permite la especificación del comportamiento del usuario con la interfaz de usuario RIA.

La interfaz final contiene la información final para la generación de código de la interfaz de usuario, lo cual es específica para un dispositivo o un grupo de dispositivos y para una plataforma de desarrollo RIA tal como Flex, Ajax o OpenLaszlo[[12]](#footnote-12).

**a4) UWE con patrones**

UWE puede extenderse por medio de patrones[].Los patrones RIAs, describen la interacción, la operación y la presentación de un *widget*, en donde la interacción es el disparador del patrón RIA (por ejemplo, el movimiento del mouse, presionar una tecla o un evento temporal). Como resultado de la interacción, una variedad de operaciones pueden ser llevadas a cabo, tales como validaciones, búsquedas y refrescados de página. Finalmente el resultado implica una actualización en la interfaz de usuario.

Este enfoque consiste en el uso de modelos que representan *widgets* RIAs, y la inclusión de estos modelos dentro de metodologías de desarrollo web existentes. Cada *widget* es modelado por medio de máquinas de estados que representan la característica RIA deseada. En las figuras 11 y 12 se muestran los patrones definidos en propuesta sugerida a la metodología UWE.

Figura 11 Patron en UWE *Periodic Refresh.*

Figura 12 Patron en UWE Autocomplete

**a5) Patrones en OOWS**

La principal contribución de este trabajo es un modelo de interacción para especificar la nueva semántica para hacer frente al desarrollo basado en modelos RIAs.[Valverde y pastor] El modelo se compone de patrones de interacción que describen, desde el punto de vista conceptual, una solución genérica para la interacción común de un usuario con un sistema siguiendo los principios de la *Human Computer Interaction (*HCI*)[[13]](#footnote-13)*, este modelo se basa en los siguientes aspectos: 1) una vista abstracta, que consta de patrones de interacción abstractos, que describen la interacción sin tener en cuenta los detalles tecnológicos y 2) una vista concreta formada por patrones de interacción RIA que especifican la nueva interacción y los requerimientos para la interfaz. Con estas dos premisas se implementan *widgets* para el autocompletado y la expansión/colapso de texto. En la Figura 13 se puede analizar el proceso de desarrollo para esta propuesta



Figura 13 Un resumen del enfoque MDD con patrones para OOWS de Valverde y Pastor

La Tabla 7 que se muestra a continuación presenta un resumen de las propuestas, indicando si las características RIAs contempladas por las mismas.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Características versus metodologías | | OOHDM-RIA | OOH4RIA | WebML - | Patrones con UWE | Patrones OOWS | UsiXML | UWE-R | Espacios interactivos con UML | UWE + RUX |
| Almacenamiento en el lado del cliente | | - |  | si | - | - | - | - | - | - |
| Lógica de negocio en el lado del cliente | **Operaciones complejas** | - | - | si | - | - | - | - | - | - |
| **Operaciones específicas del dominio** | - | - | - | - | - | - | si | - | - |
| **Validación local** | Si | si | si | - | - | - | - | - | Si |
| Presentaciones enriquecidas | **Manejo de eventos en el lado cliente** | - | - | si | si | si | - | si | si | Si |
| **Widgets** | Si | si | - | si | si | si | si | si | Si |
| **Paradigma de página única** | Si | si | si | si | - | si | - | - | Si |
| **Contenido multimedia** | - | si | - | - | - | si | si | - | Si |
| Comunicación cliente servidor | **Sincronización de datos** | - | - | si | - | - | - | si | - | Si |
| **Obtención de actualizaciones parciales de página** | si | si | si | si | si | - | si | si | Si |
| **Push y Pull** | - | - | si | - | - | - | si | - | - |

Tabla 6 Metodologías web y sus alcances para RIA

En el análisis de la Tabla 6, notamos que la metodología que más características de las RIAs abarca es WebML con la salvedad que utiliza herramientas propietarias para su modelado[[14]](#footnote-14), se basa en un DSL gráfico propio, no utiliza UML y no cubre widgets. Con respecto a la característica de presentaciones enriquecidas que es la que concierne a este trabajo de tesis, la metodología RUX y la combinación de UWE+RUX son las que ofrecen cobertura completa a diferencia de las otras metodologías. Sin embargo, RUX, no es precisamente una metodología, sino más bien, una herramienta propietaria que sirve para enriquecer con características de las RIAs a las metodologías web UsiXML, ofrece una metodología estándar bastante completa que utiliza una serie iterativa de transformaciones XSLT *(Extensible Stylesheet Language Transformations)* para obtener la interfaz de usuario final para una plataforma destino a partir de una interfaz abstracta, definida previamente, pero está abocada específicamente al desarrollo de interfaces y no es una metodología que abarque todo el ciclo de vida de una aplicación web.

Dado el comportamiento dinámico y reactivo de los *widgets* es necesario representarlos con diagramas que logren captar su dinamismo. He allí que las metodologías más influyentes en este trabajo son UWE-R, UWE con patrones, los espacios interactivos con UML, OOHDM, OOWS y OOH-4RIA que proponen diagramas interactivos (de estado y de secuencia) para la representación de los elementos interactivos, necesarios en las presentaciones de web 2.0.

**2.7 La Aproximación MoWebA (Model Oriented Web Approach)**

MoWebA[][<gonzalez2011>] es una propuesta creada en el DEI (Departamento de Electrónica e Informática) que adopta los principios de MDA. En la figura 12 se muestran las dimensiones de MoWebA. Como puede observarse, consta de fases, niveles y aspectos, que se van describiendo a continuación.

Las fases se refieren a los procesos de modelado y transformación. Estas se encuentran claramente diferenciadas e incluyen a su vez una serie de modelos entre los que se citan:

1.  **Modelado del problema:** en el que se incluyen al CIM (*Computation Independent Model*), orientado al modelado de los requisitos funcionales*,* y al PIM (*Platform Independent Model*) orientado al modelado del problema sin considerar aspectos de la arquitectura o plataforma. De aquí es posible llevar a cabo transformaciones para obtener los modelos específicos de la plataforma de manera semi-automática por medio de reglas.

2.   **Modelado de la solución**: en donde forman parte el ASM (*Architectural Specific Model*) y el PSM (*Platform Specific Mode*l). Es en esta fase, en donde 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 independiza esta fase, y esto hace que sea bastante prometedora para la implementación de las RIAs, debido a que existen numerosas plataformas destino para desplegarlas. En las aproximaciones estudiadas, por lo general las extensiones RIAs son definidas en el marco de los modelos conceptuales (PIMs), haciendo que los modelos que deberían ser independientes de la solución, adquieran elementos que ya son propios de una arquitectura específica.



Figura 14 Niveles y fases en el desarrollo de MOWEBA.

3. **Código fuente:** incluye al ISM (*Implementation specific model*) que corresponde al código generado y el código manual a ser agregado (en caso de ser necesario) para generar la aplicación final. La aplicación puede refinarse, dado que todas las fases son iterativas e incrementales.

MoWebA también presenta distintos niveles de construcción separados en capas para representar a una aplicación web. Se contemplan niveles para el contenido, la lógica del negocio, la navegación, la presentación, y los usuarios. Los aspectos están relacionados con la estructura y el comportamiento de la aplicación. Cada modelo es visto desde dos puntos de vista (estructura y comportamiento) por lo que existe una propuesta notacional para definirlos.

Definir una propuesta RIA para MoWebA resulta interesante ya que sería posible realizar un análisis para diferenciar el PIM del ASM, no contemplados en otras metodologías. Esto hace que al definir los modelos propios de las RIAs, si hubiera necesidad de llevar a cabo una migración a otra arquitectura destino, probablemente deberán realizar muchos cambios sobre el modelo mismo. En MoWebA se plantea tener siempre el mismo PIM, y a partir de este adoptar la arquitectura correspondiente.

**2.7.1 La capa de presentación de MoWebA**

La capa de presentación de MoWeba abarca a los metamodelos de contenido y estructura. En el metamodelo de contenido (*Content*), se tienen los diversos elementos de interfaz (*UIElements*) correspondientes a la web 1.0, en la que se tienen a los *textImput*, a los enlaces que podrían corresponder a navegaciones internas de la aplicación (*anchor*) o bien a navegaciones externas (*externalLink*), a los *button,* a los elementos del tipo selección que corresponde a los *choice*  y a los *dropBox*, a los *text* para texto plano en las páginas, al *htmlText* para el despliegue de cualquier texto HTML, y al elemento del tipo *multimedia* para audio y video. Cada uno de estos elementos cuenta con sus respectivos *atributos* para identificar a sus propiedades intrínsecas como puede apreciarse en la Figura 15. El *compositeUIElement* contiene a los distintos *UIElements* y en él, pueden definirse condiciones de tipo *orderBy* y *groupBy,* en caso que sea necesario obtener datos del modelo de dominio. El elemento de interfaz *form*, extiende al *compositeUIElement*, permitiendo definir a los distintos *UIElements* dentro de un formulario de entrada.

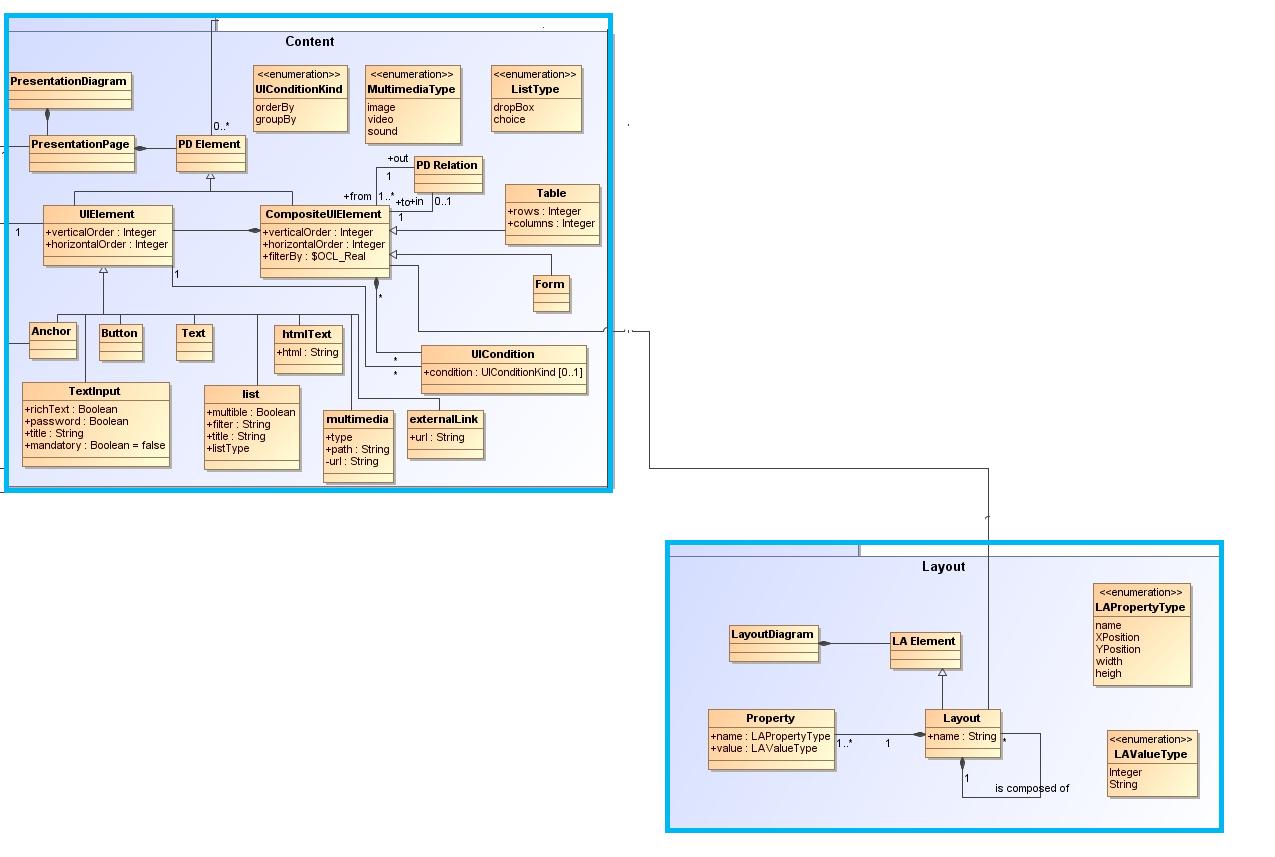


Figura 15 Metamodelo de Contenido y Estructura de MoWebA

El metomodelo de estructura (*Layout*), permite establecer a cada uno de los *compositeUIElement* definidos en el metamodelo de contenido,una posición específica dentro de las páginas. Un *Layout* está compuesto de uno o muchos *Layout*, y cada uno de ellos a la vez puede tener una o varias propiedades definidas, que corresponden a sus coordenadas posicionales.

Los metamodelos *Content* y *Layout* definen la sintaxis abstracta de la capa de presentación de MoWebA, por medio del estándar MOF (*Meta Object Facility*). La sintaxis concreta de MoWebA, es llevada a cabo por medio de UML, utilizando la técnica de perfil(*profiling*), que permite agregar a UML, los diversos estereotipos *(stereotypes)* y valores etiquetados (*tagged values*) propios de MoWebA. Los perfiles *Content y Layout*, permiten definir los PIM de presentación de una aplicación modelada con MoWebA. Los perfiles de presentación se presentan en la Figura 16.

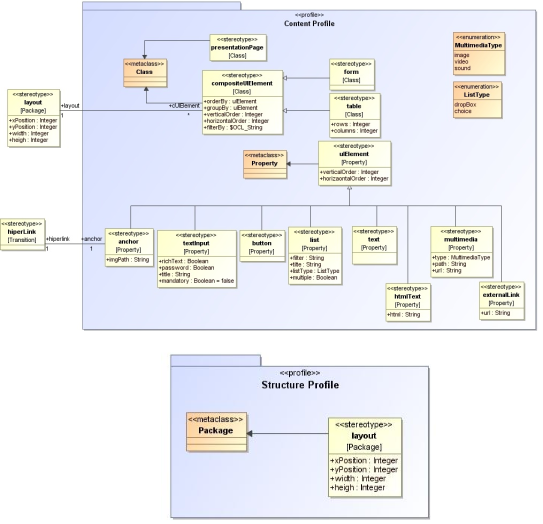


Figura 16: Perfiles de Contenido y Estructura de MoWeba

A modo de ejemplo se presenta un PIM correspondiente a la presentación de una aplicación con MoWebA

2.8 Síntesis del Capítulo

En este capítulo se han visto las diversas características que ofrecen las RIAs, como así también los enfoques tecnológicos para explotar el lado del cliente en este tipo de aplicaciones. Estos enfoques son: las implementaciones basadas en librerías *Javascript,* , las basadas en la instalación de *plug-ins* o las que se enfocan en ambientes en tiempo de ejecución. La primera de ellas es la más utilizada en la comunidad web, debido a que la aplicación web se implementa por medio de un compendio de estándares abiertos trabajando conjuntamente como lo son HTML y CSS *(para la representación de los elementos y el posicionamiento)*, *Javascript (para la lógica de la aplicación en el lado cliente)* y *XML* o JSON*(para la comunicación entre el cliente y el servidor)*. También resulta importante resaltar el hecho que este enfoque es el más cercano al último estándar HTML5. Para este trabajo de fin de carrera, la plataforma destino en el dominio web, se basa en el conjunto de estándares abiertos citados anteriormente, por lo tanto, el enfoque elegido es el basado en librerías *Javascript*.

Son varias las librerías *Javascript* existentes en la actualidad[[15]](#footnote-15). Algunas de ellas permiten la representación de ciertos elementos de interfaz interactivos (*widgets*) que son comunes en las interfaces enriquecidas actuales y a la vez ofrecen la posibilidad de agregar cierta lógica en el lado cliente, como validaciones locales de campos de entrada en los formularios. Según el análisis llevado a cabo a principales portales web, dentro de los *widgets* más utilizados se ven a los *tooltips*, *tabs, accordion, datepicker y autocomplete,* como así también diversas validaciones locales en los campos de entrada como validaciones de tipo numérico, email, password etc. *jQuery* a la par de ser la librería *Javascript* más popular actualmente, ofrece en sus versiones *jQueryUI* y *JQuery Validation Plugin,*cobertura a todas estas características enriquecidas.

Los modelos hoy en día ocupan un lugar importante en proceso de desarrollo de software, comúnmente para la comunicación entre los desarrolladores y las personas sin conocimientos técnicos u bien entre los mismos desarrolladores. Las metodologías de desarrollos enmarcadas en el contexto MDD y MDA toman estos modelos y por medio de transformaciones sobre los mismos (aplicando técnicas *M2M* o *M2T)*, pueden obtener el código fuente de una aplicación para una plataforma destino en particular. Son varias las metodologías web existentes en la actualidad en donde los modelos y las transformaciones sobre estos son los elementos principales del proceso. Algunos ejemplos son:OOHDM-RIA, OOH4RIA, WebML-RIA, Patrones con UWE, Patrones OOWS, UsiXML, UWE-R, Espacios interactivos con UML y UWE + RUX. Se ha visto que muchas de ellas cubren características de las RIAs en ciertos aspectos, pero ninguna ofrece cobertura total a todas las características. Es por ese motivo, que resulta necesario extender alguna de las metodologías web existentes o bien crear nuevas metodologías web para satisfacer esta necesidad.

La metodología web MoWebA, resulta prometedora para llevar a cabo extensiones y de esta forma dar cobertura a características de las RIAs, debido principalmente a la forma en la que está estructurada la metodología, en donde existe una separación adecuada de conceptos y capas para el modelado de una aplicación a la par de contemplar todo el ciclo de desarrollo de una aplicación. Este trabajo de fin de carrera se aboca a la extensión de elementos enriquecidos en la capa de presentación de MoWebA, en donde se contemplan los diversos elementos para la interfaz del usuario y la estructuración de estos dentro de las páginas. Estos elementos enriquecidos de interfaz (widgets) son precisamente los denominados, *richToolTip*, *richAccordion*, *richAutoSuggest*, *richDatePicker*, *richTabs* y el elemento de validación para los campos de entrada *richFieldLiveValidation.*

1. Wikipedia- *Rich internet applications*: <http://en.wikipedia.org/wiki/Rich_Internet_application> 2015 [↑](#footnote-ref-1)
2. **Oracle:** <http://www.oracle.com/technetwork/java/javase/javawebstart/index.html> 2015 [↑](#footnote-ref-2)
3. **Adobe AIR:** <http://www.adobe.com/products/air.html> 2015 [↑](#footnote-ref-3)
4. Describe un estilo de comunicacion sobre [Internet](http://es.wikipedia.org/wiki/Internet) donde la petición de una transacción se origina en el [servidor](http://es.wikipedia.org/wiki/Servidor). [↑](#footnote-ref-4)
5. **Designing Web Interfaces:** <http://designingwebinterfaces.com/essential_controls> 2009 [↑](#footnote-ref-5)
6. **UX BOOTH:** <http://www.uxbooth.com/articles/essential-controls-for-web-applications/> 2010 [↑](#footnote-ref-6)
7. **jQuery user interface:** <http://jqueryui.com/> 2015 [↑](#footnote-ref-7)
8. **jQuery Validation Plugin:** <http://jqueryvalidation.org/> 2015 [↑](#footnote-ref-8)
9. **jQuery:** <http://jquery.com/> 2015 [↑](#footnote-ref-9)
10. **Usage of JavaScript libraries for websites** <http://w3techs.com/technologies/overview/javascript_library/all> 2015 [↑](#footnote-ref-10)
11. **Google Web Toolkit:** <http://www.gwtproject.org/> 2015 [↑](#footnote-ref-11)
12. **OpenLaszlo:** <http://www.openlaszlo.org/> 2015 [↑](#footnote-ref-12)
13. **INTERACTION DESIGN FOUNDATION:**<http://www.interaction-design.org/encyclopedia/human_computer_interaction_hci.html> 2015 [↑](#footnote-ref-13)
14. **WebRatio:** <http://www.webratio.com/site/content/es/home> 2015 [↑](#footnote-ref-14)
15. **List of Ajax frameworks:** <http://en.wikipedia.org/wiki/List_of_Ajax_frameworks> 2015 [↑](#footnote-ref-15)