Un enfoque de Desarrollo Dirigido por Modelos para Aplicaciones Web Enriquecidas

Guido Nuñez*, Magalí González†, Nathalie Aquino‡ & Luca Cernuzzi§
Departamento de Electrónica e Informática - DEI
Universidad Católica "Nuestra Señora de la Asunción"
Asunción, Paraguay
*Email: {*guido.nunez, †mgonzalez, ‡nathalie.aquino, §lcernuzz}@uc.edu.py*

***Abstract*—Diversas aplicaciones Web cuentan entre sus características la posibilidad de distribuir sus datos y su lógica de negocios entre el cliente y el servidor, permitiendo también una comunicación asíncrona entre los mismos. Estas características, originalmente asociadas al advenimiento de las aplicaciones enriquecidas de internet (RIA), siguen siendo particularmente relevantes y deseables. En el ámbito de las RIA, son escasas las propuestas que, a la vez, consideran estas características, adoptan MDD (Model-Driven Development), y utilizan tecnologías de implementación basadas en *scripting*. En este trabajo, partimos de MoWebA, un enfoque MDD para el desarrollo de aplicaciones web, y lo extendemos definiendo un modelo específico de arquitectura con funcionalidades RIA, dando soporte a las características previamente mencionadas. Hemos definido los metamodelos y perfiles UML necesarios, así como las reglas de transformación que permiten generar código basado en HTML5, Javascript, jQuery, jQuery Datatables y jQuery UI. La validación preliminar de la propuesta muestra evidencias positivas en relación a la eficacia, eficiencia y satisfacción de los usuarios con respecto a los procesos de modelado y de generación de código de la propuesta.**

***Keywords*—Desarrollo Dirigido por Modelos, Model Driven Development, MDD, Arquitectura Dirigida por Modelos, Model Driven Architecture, MDA, Aplicaciones Enriquecidas de Internet, Rich Internet Applications, RIA, Model Oriented Web Approach, MoWebA.**

1. Introducción

Anteriormente las aplicaciones Web estaban caracterizadas por un procesamiento realizado del lado del servidor, donde el cliente solo se encargaba de solicitar y desplegar contenido. Esto generaba interacciones muy limitadas con el usuario, necesidad de recargar toda la página para realizar la navegación, tiempos de respuesta muy largos y dificultad para desplegar animaciones y contenidos multimedia.

Para enfrentarse a estas limitaciones han aparecido las Aplicaciones Enriquecidas de Internet o RIA (Rich Internet Applications) [1]. Este tipo de aplicaciones ha permitido mejorar notablemente la experiencia del usuario, principalmente en cuanto a aspectos de presentación e interacción se refiere. Las RIA poseen cuatro características principales que son: la distribución de datos, la distribución de la lógica de negocios, la comunicación asíncrona entre cliente y servidor, y la mejora de la interfaz de usuario [1], [2]. Actualmente, independientemente de que siguan una arquitectura específica RIA o no, un porcentaje creciente de aplicaciones Web han adoptado entre sus características la posibilidad de distribuir sus datos y su lógica de negocios entre el cliente y el servidor, permitiendo también una comunicación asíncrona entre los mismos. Se han propuesto muchas tecnologías para facilitar el desarrollo de RIA, y por ende el soporte a las características mencionadas, y las mismas han logrado incrementar la productividad del desarrollador. Sin embargo, subsisten ciertas desventajas tales como la carencia de instrumentos de alto nivel que abstraigan detalles de implementación, la propensión a errores, la falta de consideración del ciclo de vida completo del software, entre otras más [3].

Los enfoques de Desarrollo Dirigido por Modelos o MDD (Model Driven Development) [4] ofrecen soluciones a dichas limitaciones, permitiendo crear aplicaciones a través de la especificación de modelos y generar código a partir de ellos. Específicamente, en el caso de que estos enfoques adopten un estándar como la Arquitectura Dirigida por Modelos o MDA (Model Driven Architecture) [5], se ve facilitada la tarea de dirigir el proceso de desarrollo y garantizar una mayor interoperabilidad y portabilidad entre sistemas. En este sentido, se encuentran en la literatura diversas propuestas MDD para la implementación de RIA [3] [2]. Sin embargo, muchas de éstas se enfocan en la capa de presentación y consideran escasamente las características de distribución de datos, distribución de la lógica de negocios y comunicación asíncrona entre cliente y servidor. Entre los enfoques que sí consideran estas últimas características, se puede notar que los mismos no han adoptado estándares o no han desarrollado una implementación con tecnologías basadas en *scripting*, las que constituyen la opción más común para las prácticas de desarrollo de RIA [3].

Una práctica interesante para desarrollar RIA consiste en dotar a metodologías MDD orientadas al desarrollo Web con nuevas funcionalidades, ya sea introduciendo nuevos elementos a nivel del modelo independiente de plataforma o PIM (Platform Independent Model), o a través de un

nivel de abstracción menor que permita introducir información acerca de una arquitectura determinada. En esta línea, MoWebA [6] constituye un enfoque MDD para el desarrollo de aplicaciones Web, adopta el estándar MDA y, además, permite ser extendido a través de la definición de un Modelo Específico de Arquitectura o ASM (Architecture Specific Model), utilizado para complementar a los modelos PIM con información acerca de una arquitectura específica. MoWebA ya ha sido previamente extendido para dar soporte a las RIA [7], pero dicha extensión se ha limitado a aspectos de presentación enfocándose exclusivamente en la característica de mejora de interfaz de usuario, por lo que vemos conveniente realizar una extensión complementaria a fin de dar soporte a las demás características deseables de aplicaciones Web enriquecidas.

En base a lo expuesto, este trabajo define una extensión de MoWebA centrada en el desarrollo de RIA, y, más en general, aplicaciones Web enriquecidas con características de distribución de datos, distribución de lógica de negocios y comunicación asíncrona entre cliente y servidor.

El resto del trabajo se encuentra organizado como sigue: la sección 2 presenta el estado del arte de los enfoques MDD para el desarrollo de RIA existentes; la sección 3 describe la extensión de MoWebA presentando los metamodelos, perfiles y reglas de transformación correspondientes a esta propuesta; la sección 4 describe la experiencia de validación realizada, y la sección 5 finaliza el documento especificando la conclusión del trabajo.

2. Trabajos Relacionados

Se ha realizado un estudio de la literatura acerca de los trabajos existentes relacionados a enfoques MDD para el desarrollo de RIA que contemplen las características de distribución de datos, distribución de lógica de negocios y comunicación asíncrona entre cliente y servidor, que son las características de interés en este trabajo. Los principales resultados se muestran en la tabla 1.

En esta tabla podemos observar los enfoques identificados junto con sus referencias, ordenados por año de menor a mayor. También se presenta información sobre las características RIA soportadas, la adopción de MDA como estándar para dirigir el proceso de desarrollo, las tecnologías de implementación y los tipos que constituyen estas tecnologías.

Notamos que los trabajos comprendidos entre los años 2006 y 2010 llegaron a considerar las características de distribución de datos, distribución de lógica de negocios y comunicación asíncrona entre cliente y servidor, mientras que los trabajos más contemporáneos que van entre los años 2013 y 2016, han considerado escasamente estas características, enfocándose mayormente en la característica de interfaz de usuario enriquecida. Si bien somos conscientes de la importancia de la capa de presentación para asegurar una experiencia positiva al usuario, desde el punto de vista funcional y de prestaciones, las otras características siguen jugando un rol muy importante y llama la atención como han sido escasamente consideradas en los últimos años. Por esto, nuestra propuesta actual se enfoca a recuperar la importancia del modelado y soporte al desarrollo de dichas características.

Entre los trabajos que abarcan las características de interés en forma simultánea, solo Meliá et al. [8] adopta MDA para guíar su proceso. La adopción de MDA es conveniente para estandarizar y mejorar el proceso de desarrollo de software estableciendo una mejor interoperabilidad y portabilidad entre sistemas.

Además, este mismo trabajo es el único que genera una implementación haciendo uso de tecnologías basadas en *scripting*, aquellas tecnologías mayormente utilizadas en la comunidad de desarrolladores para la implementacion de RIA [3], que en particular, favorecen el desarrollo de las características de distribución de datos y de lógica de negocios, así como la comunicación asíncrona entre cliente y servidor. Otras tecnologías basadas en *scripting* como HTML5, jQuery y GWT no son tenidas en cuenta para la implementación de las características de interés.

En virtud de lo señalado, percibimos la necesidad de proveer una solución al desarrollo de RIA, y más en general de aplicaciones Web enriquecidas, mediante un enfoque MDD que contemple las características de distribución de datos, distribución de lógica de negocios, y comunicación asíncrona entre cliente y servidor, adopte estándares (como por ejemplo MDA) y genere una implementación con tecnologías basadas en *scripting*.

3. Una Extensión RIA de MoWebA

En este trabajo, tomamos como punto de partida a MoWebA [6] proponiendo una extensión para funcionalidades propias de RIA, y en general, de aplicaciones Web enriquecidas. Optamos por MoWebA debido a que presenta ciertas ventajas en comparación a otras propuestas, entre ellas la adopción de estándares, la provisión de un ASM para extender al enfoque con funcionalidades relativas a una arquitectura específica y la concepción de un modelado basado en una navegación orientada a funciones. Esta navegación orientada a funciones provee una mayor cohesión y menor acoplamiento con respecto a propuestas orientadas a datos y potencia un desarrollo *top-down*, que a fines arquitecturales provee una estructura más clara y comprensible, reflejando de mejor manera las necesidades del usuario.

3.1. ASM RIA para MoWebA

A fin de facilitar el modelado de funcionalidades RIA, implementamos extensiones al metamodelo y perfiles originales de MoWebA. Estas extensiones corresponden a funcionalidades que no se incluían en la versión inicial de MoWebA y que permiten facilitar el posterior modelado de características RIA. Aplicamos las extensiones a los diagramas lógico y de contenido. Mayores detalles acerca del metamodelo y perfiles originales de MoWebA pueden consultarse en [6].

A partir de las extensiones realizadas, desarrollamos el metamodelo de la Figura 1, utilizado para la definición de

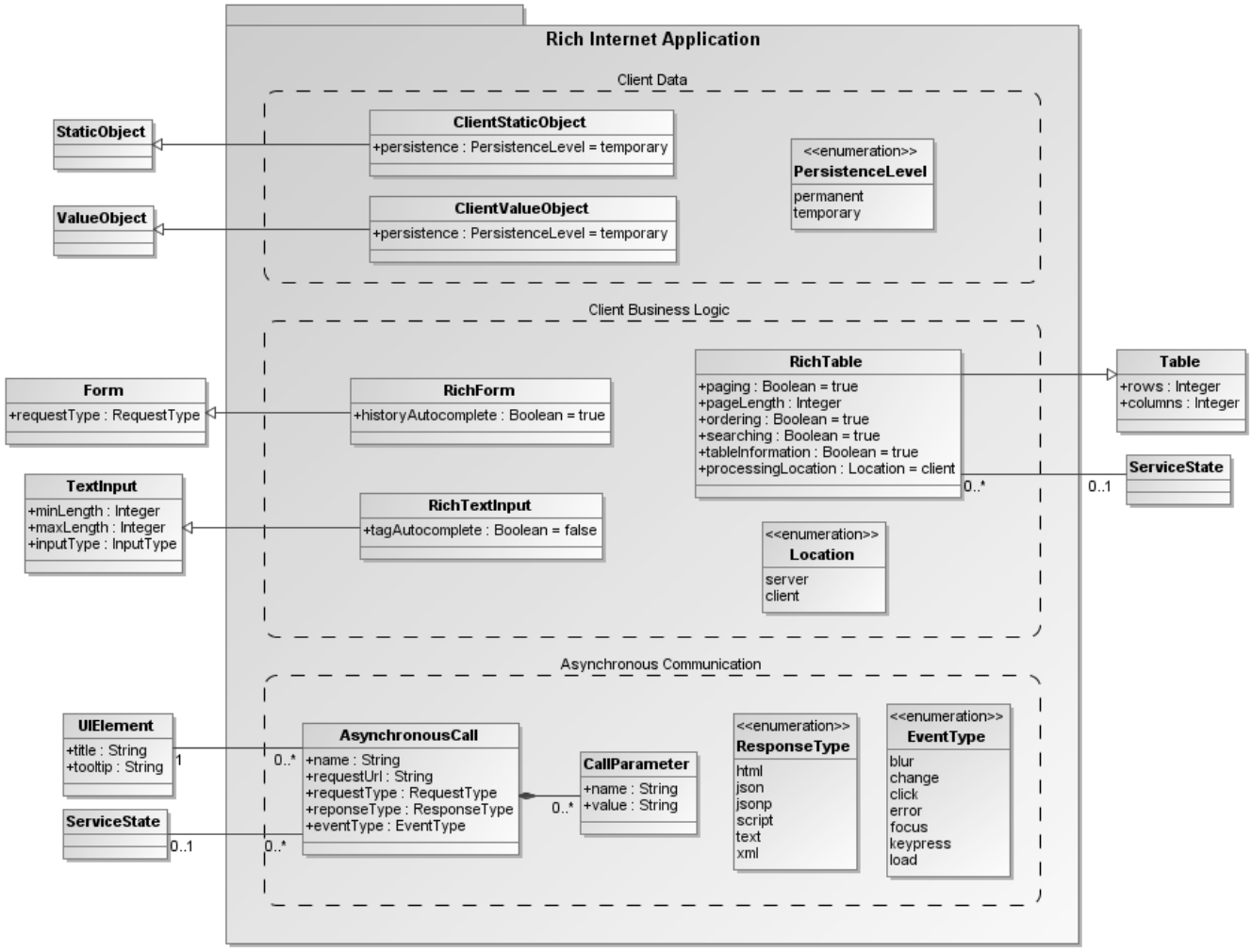
Table 1. ENFOQUES MDD PARA EL DESARROLLO DE RIA.

Criterios / Enfoques	Año	Distribución de Datos	Distribución de Lógica de Negocios	Comunicación Asíncrona	MDA	Tecnología de Implementación	Tipo de Tecnología
WebML for RIA [9]	2006	✓	✓	✗	No	OpenLaszlo	Basadas en *plugin*
RUX-Model [10]	2007	✗	✗	✓	Si	Adobe Flex, OpenLaszlo	Basadas en *plugin*
WebML for RIA + RUX-Model [11]	2007	✓	✓	✓	No	OpenLaszlo, Adobe Flex, XAML	Basadas en *plugin*
WebML for RIA para aplicaciones web colaborativas [12]	2007	✓	✓	✓	No	OpenLaszlo	Basadas en *plugin*
Enfoque para diseño conceptual de RIA basado en procesos de negocios [13]	2008	✓	✓	✓	No	No especificado	-
OOWS Extension [14]	2009	✗	✗	✓	No	Adobe Flex	Basadas en *plugin*
OOH4RIA Extension [8]	2010	✓	✓	✓	Si	RichFaces	Basadas en *scripting*
Enfoque MDD para componentes GUI interactivos basados en HTML5 [15]	2013	✗	✗	✗	No	HTML5, Javascript	Basadas en *scripting*
Enfoque MDD para aplicaciones web de alta calidad [16]	2013	✗	✗	✓	Si	Javascript	Basadas en *scripting*
Enfoque MDD basado en UML para aplicaciones web MVP [17]	2014	✗	✗	✗	Si	GWT	Basadas en *scripting*
Enfoque MDD para prototipado de RIA [18]	2014	✗	✗	✓	No	RichFaces	Basadas en *scripting*
Proceso MDUID integrando patrones HCI [19]	2015	✗	✗	✗	No	JSP, Javascript, CSS	Basadas en *scripting*
Enfoque MDA para patrones MVP, DI y DAO para RIA [20]	2015	✗	✗	✗	Si	GWT	Basadas en *scripting*
Enfoque para la generación MDD de GUI de RIA usando IFML [21]	2015	✗	✗	✗	Si	JavaFX	Ambientes de ejecución específicos
Enfoque MDA para aplicaciones web [22]	2015	✓	✗	✓	Si	WCF	Basadas en *scripting*
Proceso MDD para MVC RIA enfocado en GUI [23]	2016	✗	✗	✗	Si	JavaFX	Ambientes de ejecución específicos
Enfoque MDD para GUI de RIA usando IFML y Ontologías [24]	2016	✗	✗	✗	Si	Adoble Flex	Basadas en *plugin*

un ASM para RIA. En él se presentan las diferentes funcionalidades incorporadas para modelar las características de distribución de datos, distribución de lógica de negocios y comunicación asíncrona entre cliente y servidor.

Para guardar datos en un cliente web, específicamente en un navegador web, creamos dos nuevas clases, un *objeto de valor en el cliente* (*ClientValueObject*), que extiende al objeto de valor del diagrama lógico original de MoWebA, y un *objeto estático en el cliente* (*ClientStaticObject*), que extiende a un objeto estático, un nuevo elemento introducido en el diagrama lógico para representar conjuntos de valores definidos de forma estática como propiedades de la clase. Estos nuevos objetos se diferencian de los originales a nivel de implementación, ya que tanto el *ClientValueObject* como el *ClientStaticObject* son mapeados a variables almacenadas en el navegador. El nombre de la clase corresponde al nombre de la variable y los valores de la clase con sus valores etiquetados asociados (*title* y *checked*), dan lugar a los valores de la variable. Además, estas nuevas clases permiten especificar un nivel de persistencia (*persistence*) que

Figure 1. Metamodelo utilizado para la definición de un ASM para RIA.



puede ser permanente (*permanent*) o temporal (*temporary*), permitiendo que la variable persista o no al término de una sesión o cierre del navegador. Si el nivel de persistencia es permanente se generará el objeto *Localstorage*[1] de HTML, mientras que si la persistencia es temporal se generará el objeto *Sessionstorage*[2] de HTML, ambos accedidos a través de Javascript.

En cuanto a procesos ejecutados en el cliente, creamos tres elementos que se ejecutan en el navegador web, el formulario enriquecido (*RichForm*), la entrada de texto enriquecida (*RichTextInput*) y la tabla enriquecida (*RichTable*).

El *formulario enriquecido* consiste en una especialización del elemento formulario (*Form*) del diagrama de contenido original de MoWebA. Contiene un atributo de autocompletado basado en historia (*historyAutocomplete*), que permite al navegador sugerir valores para completar las entradas del formulario, basados en valores previamente ingresados. A partir de cada instancia de un formulario enriquecido se generará un formulario HTML con la propiedad de autocompletado habilitada o no, según lo establecido en el modelo.

La *entrada de texto enriquecida* extiende a la entrada de texto (*TextInput*) del diagrama de contenido original de MoWebA. Este elemento introduce el atributo de autocompletado (*tagAutocomplete*), que se diferencia del autocompletado anterior en que los valores a sugerir son obtenidos a partir de una asociación con objetos de valor u objetos estáticos y sus especializaciones. Posteriormente, haciendo uso de jQuery UI[3], se generará la asociación entre entradas y valores y se desplegará la opción de autocompletado.

La *tabla enriquecida* especializa a la clase tabla (*Table*) del diagrama de contenido original de MoWebA, permite especificar el servicio del diagrama de nodos que se encargará de poblar la tabla. Además, agrega nuevas propiedades a la tabla para permitir paginación (*paging*), especificar

1. LocalStorage.https://www.w3.org/TR/webstorage/#the-localstorage-attribute
2. SessionStorage.https://www.w3.org/TR/webstorage/#the-sessionstorage-attribute
3. JQuery UI.https://jqueryui.com/

cantidad de registros por página (*pageLength*), permitir ordenamiento por columnas (*ordering*), buscar palabras en registros (*searching*), desplegar información (índice de página actual, cantidad de páginas, cantidad de registros) de la tabla (*tableInformation*) y especificar dónde se procesarán estas funciones (*processingLocation*), ya sea en el servidor (*server*) o en el cliente (*client*). El *plugin* de jQuery, Datatables[4] será utilizado para la implementación de una tabla enriquecida.

Para permitir comunicaciones asíncronas entre cliente y servidor creamos la clase denominada *llamada asíncrona* (*AsynchronousCall*). Esta clase es utilizada para realizar una solicitud AJAX (HTTP asíncrona) de un servicio. La llamada asíncrona se asocia a uno o muchos elementos de interfaz de usuario del diagrama de contenido y a cero o un servicio del diagrama de nodos. Cuando ocurre un evento sobre algún elemento de interfaz, se ejecuta un servicio. El servicio puede ser alguno especificado en el diagrama de nodos, o un servicio ejecutándose desde una URL ajena al sistema (por ejemplo un web *service*). La instancia de la llamada asíncrona puede ubicarse en el diagrama de contenido junto a sus elementos de interfaz de usuario asociados. La clase posee un nombre (*name*); una URL (*requestUrl*), utilizada en caso de que se solicite un servicio externo; un tipo de solicitud (*requestType*), que puede utilizarse para recuperar datos remotos (*retrieve*) o insertar o actualizar datos (*insert/update*); un tipo de respuesta (*responseType*), que puede ser en formato *html*, *json*, *jsonp*, *script*, *text*, *xml*; y un tipo de evento (*eventType*), que se aplica al elemento de interfaz de usuario y puede tomar los valores *blur*, *change*, *click*, *error*, *focus*, *keypress*, *load*. Además, la llamada asíncrona puede adjuntar ciertos parámetros a la solicitud. Cada parámetro consiste en una propiedad de la clase y posee un nombre y un valor. Cada llamada asíncrona generará un método *ajax()* de jQuery[5].

3.2. Reglas de Transformación

En el proceso de modelado, luego de la elaboración del ASM de la aplicación es posible pasar al proceso de generación de código a partir del modelo. Para ello, hemos definido reglas de transformación de modelo a texto haciendo uso de la herramienta Acceleo[6].

Las reglas de transformación siguen un enfoque basado en plantillas en el cual se especifican plantillas de texto con entradas para datos a extraer de los diagramas del modelo. Se utiliza el lenguaje MTL[7] para la definición de las plantillas y OCL[8] para realizar las consultas al modelo. Además, se utilizan servicios definidos en Java para extender a MTL con mayores funcionalidades.

Estas reglas se encargan de realizar un mapeo entre elementos definidos en los diagramas lógico y de contenido del modelo de la aplicación, con código en los lenguajes HTML5 y Javascript y las librerías jQuery, jQuery UI, y Datatables. Las reglas de transformación completas se encuentran en [25], anexo 2.

3.3. Proceso de Desarrollo RIA

La figura 2 ilustra el proceso propuesto para el desarrollo de RIA. Con el fin de modelar y posteriormente generar una RIA, se parte haciendo uso de la herramienta MagicDraw[9]. Mediante la importación en dicha herramienta de los perfiles de MoWebA para RIA propuestos, es posible realizar el modelo de una RIA que aproveche las diversas funcionalidades proporcionadas por este trabajo. Este modelo es exportado a un archivo en formato XMI, el cual a su vez es importado en la herramienta Acceleo. Esta última se vale de las reglas de transformación descritas en la sección anterior para realizar transformaciones de modelo a texto, generando el código en las tecnologías HTML5, Javascript, jQuery, Datatables y jQuery UI, correspondiente a la implementación final de la RIA. Las herramientas propuestas se encuentran disponibles en http://www.dei.uc.edu.py/proyectos/mddplus/herramientas/mowebaria/.

3.4. Ejemplo de Modelado y Generación de Código

A continuación presentamos un ejemplo de modelado y generación de código de una aplicación haciendo uso de la propuesta. La aplicación consiste en un sistema de marcación de empleados, en el cual un usuario invitado puede registrarse como empleado para posteriormente iniciar sesión y poder realizar marcaciones de entrada o salida. Si el usuario es un supervisor, es capaz de observar las marcaciones realizadas por los empleados. Por motivos de espacio, solo ejemplificamos un diagrama, el cual contempla la característica de lógica de negocios en el cliente. El ejemplo completo que cubre todas las características puede consultarse en [25], capítulo 4.

En la figura 3 presentamos un diagrama del ASM para RIA que aprovecha la lógica de negocios en el cliente haciendo uso de la tabla enriquecida. Se tiene una clase con estereotipo $<<richTable>>$, representando una tabla enriquecida con operaciones que pueden realizarse del lado del cliente. Se utilizan los valores etiquetados para habilitar ordenamiento, paginación y búsqueda del lado del cliente. Además, se especifica el número máximo de registros por página (quince) y se habilita la información de la tabla. Por último, se fija el servicio *Obtener Marcaciones* como encargado de proveer registros a la tabla.

A la izquierda de la figura 4 observamos el código generado a partir del diagrama anterior. Éste contiene código jQuery DataTables asociado a la tabla enriquecida *Marcaciones* permitiendo proveer las distintas funcionalidades especificadas. A la derecha de la figura se puede observar la tabla enriquecida desplegada.

4. Datatables.https://datatables.net/
5. JQuery.https://jquery.com/
6. Acceleo. https://eclipse.org/acceleo/
7. MTL.http://www.omg.org/spec/MOFM2T/1.0/
8. OCL.http://www.omg.org/spec/OCL/
9. MagicDraw. http://www.nomagic.com/products/magicdraw.html

Figure 2. Proceso de modelado y generación de RIA.

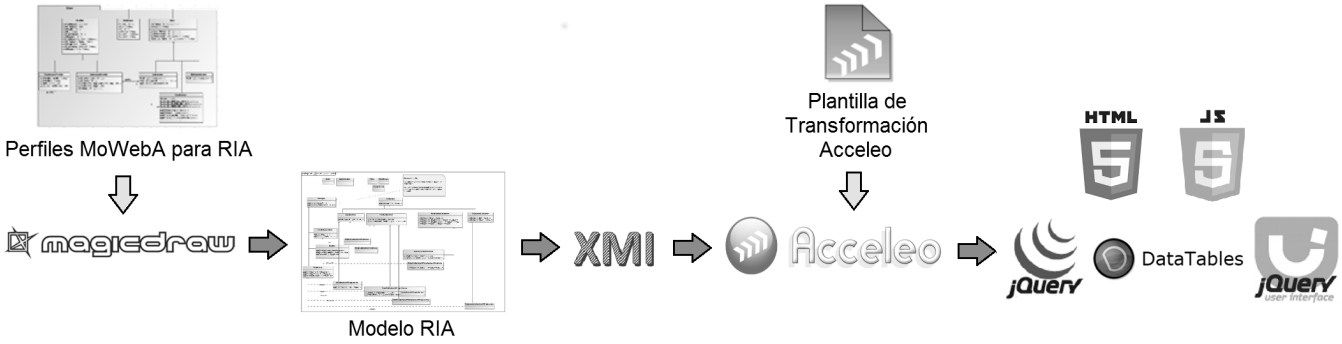
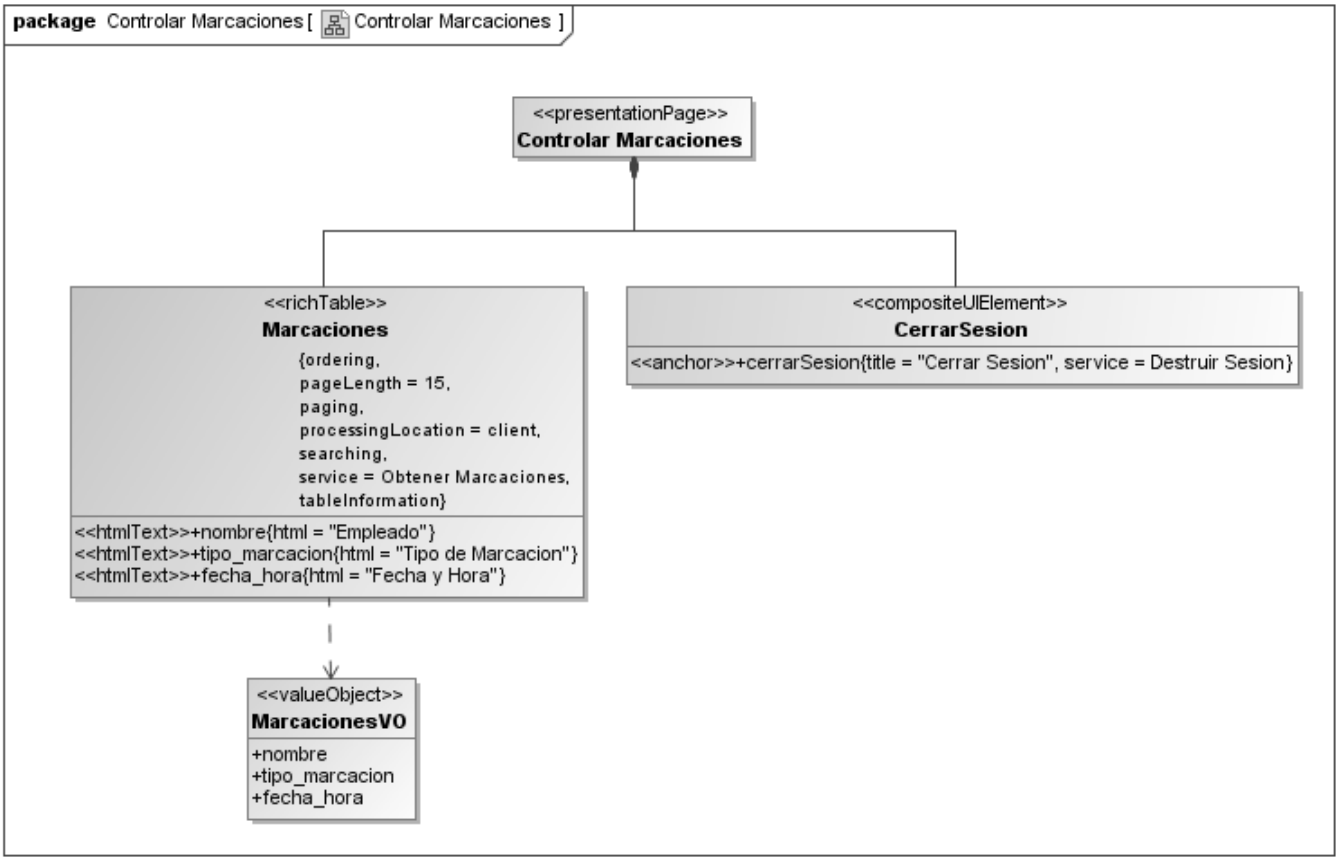


Figure 3. Diagrama del ASM para RIA con operaciones de lógica de negocios en el cliente.

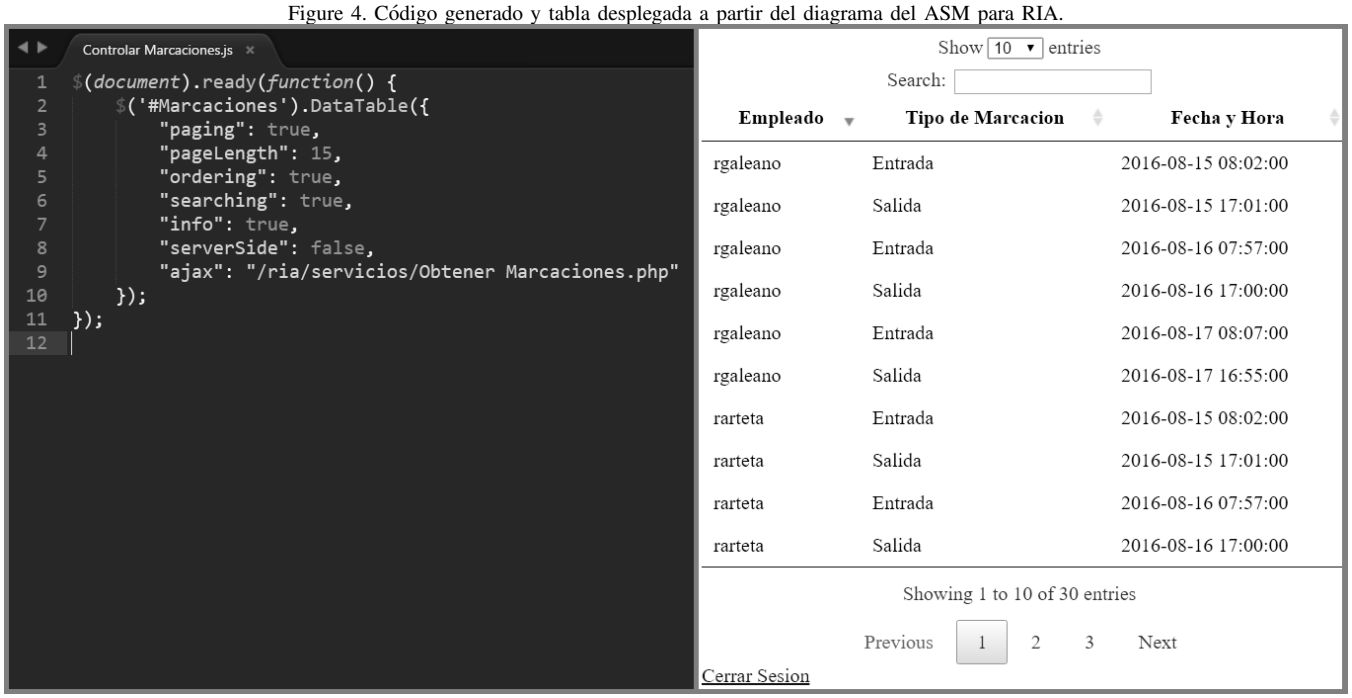


4. Una experiencia de validación

Hemos realizado una validación preliminar de la propuesta enfocándonos en la usabilidad, una importante característica de la calidad del software. La ISO 9241-11 [26] provee una guía sobre usabilidad y la define como "la medida en que un producto puede ser utilizado por usuarios específicos para lograr objetivos específicos con eficacia, eficiencia y satisfacción en un contexto específico de uso". Además, la eficacia se define como "la exactitud y la integridad con la que los usuarios logran los objetivos específicos". La eficiencia se relaciona con "los recursos empleados en relación con la exactitud y la integridad con la que los usuarios alcanzan los objetivos". La satisfacción se define como "la ausencia de incomodidad, y las actitudes positivas hacia el uso del producto".

Haciendo uso de la plantilla GQM (Goal-Question-Metric) [27], el objetivo de la validación se enuncia de la siguiente manera: *analizar* el enfoque MoWebA para el desarrollo de RIA *con el propósito de* valorar su usabilidad *con respecto* a la eficacia, eficiencia y satisfacción *desde el punto de vista del* desarrollador *en el contexto de* estudiantes del último año de la carrera de Ingeniería Informática de la Universidad Católica modelando y generando el código de

Figure 4. Código generado y tabla desplegada a partir del diagrama del ASM para RIA.



RIA usando esta propuesta. A partir de este objetivo y del hecho de que un enfoque MDD puede separarse en procesos de modelado y de generación de código, se establecieron las siguientes preguntas de investigación:

• **PI1**: ¿Qué eficacia, eficiencia y satisfacción presenta el proceso de modelado del enfoque propuesto?

• **PI2**: ¿Qué eficacia, eficiencia y satisfacción presenta el proceso de generación de código del enfoque propuesto?

Estas preguntas, además de guiar el desarrollo de la experiencia, son empleadas posteriormente para la presentación de los resultados.

4.1. Caso y Unidad de Análisis

El caso consiste en un proyecto de desarrollo de una RIA haciendo uso del enfoque MDD propuesto. La aplicación solicitada consiste en un sistema de inscripción de estudiantes a la carrera de Ingeniería Informática compuesto por dos páginas con elementos RIA. Se solicitó a los alumnos modelar los diagramas necesarios, generar código a partir de ellos y realizar ajustes manuales al código generado.
Las unidades de análisis corresponden a la fase de modelado y a la fase de generación de código.

4.2. Procedimientos

La experiencia fue desarrollada por alumnos del último año de la carrera de Ingeniería Informática de la Universidad Católica. Estos alumnos poseen experiencia en procesos basados en modelos adquirida en una materia anterior y en procesos dirigidos por modelos desarrollada en la materia en la cual se realizó esta experiencia.

En una primera sesión de trabajo de 2 horas y media, esta propuesta fue presentada y explicada a los alumnos, y se desarrolló con ellos el proceso de modelado y generación de código del ejemplo del sistema de marcación de empleados que se comentó anteriormente. En una segunda sesión de trabajo de también 2 horas y media, se solicitó que los alumnos modelen y generen el código de la RIA correspondiente al caso presentado en la sección anterior. Los estudiantes recibieron un documento con los requisitos del sistema. Con ello, iniciaron el proceso de modelado, que fue temporizado para obtener datos relacionados a la eficiencia. Al acabar de modelar, los alumnos contestaron un cuestionario ASQ (After Scenario Questionnaire) [28]. Este cuestionario permite conocer una percepción de la satisfacción de una persona con respecto a la realización de una tarea. Posteriormente, los alumnos procedieron a generar el código de la aplicación y a realizar ciertos ajustes manuales para la obtención de la RIA final y funcional. De nuevo, los tiempos fueron medidos. Además, al final del proceso de generación y ajustes al código, los alumnos contestaron un cuestionario ASQ relacionado a esa tarea. Finalmente, los modelos y código generados y ajustados por los alumnos fueron corregidos de manera a obtener tasas de éxito (en relación a la eficacia) para el proceso de modelado y para el proceso de generación de código.

4.3. Resultados

A continuación, por cada pregunta de investigación, se presentan los datos que se han recabado para contestarlas y los análisis realizados a partir de dichos datos.

PI1: ¿Qué eficacia, eficiencia y satisfacción presenta el proceso de modelado del enfoque propuesto?

La Tabla 2 presenta las mediciones de usabilidad para el proceso de modelado realizado durante la experiencia. En promedio, observamos que los alumnos culminaron el modelo de la aplicación con una tasa de éxito del 85 %, en un tiempo de 44,2 minutos, y con un puntaje ASQ de 2.67 (los puntajes del ASQ van del 1 al 7, y los valores más cercanos al 1 son los que denotan un mayor nivel de satisfacción).

Table 2. MEDICIONES DE USABILIDAD PARA EL PROCESO DE MODELADO.

Escenario	Tasa de Éxito Promedio	Tiempo de Finalización Promedio	Satisfacción Promedio
Modelado	85 %	44.2 min.	2.67

Analizando estos resultados, podemos ver que se obtuvo una tasa de éxito satisfactoria. La mayor dificultad al momento de modelar se dio al aplicar erróneamente algunos valores etiquetados y al olvidar incluir un botón de envío en los formularios.

En cuanto al tiempo de modelado, si bien no podríamos afirmar con certeza que corresponde a un tiempo favorable o desfavorable, debido a que para ello deberíamos de compararlo contra otro enfoque o proceso que implemente la misma aplicación, por nuestra experiencia en modelado y desarrollo nos parece un tiempo razonable.

El puntaje obtenido a partir del cuestionario ASQ refleja un nivel de satisfacción bueno por parte de los alumnos con el proceso de modelado propuesto.

De lo anterior podemos derivar que se obtuvo una buena eficacia y satisfacción en el proceso de modelado.

PI2: ¿Qué eficacia, eficiencia y satisfacción presenta el proceso de generación de código del enfoque propuesto?

En la tabla 3 presentamos las mediciones de usabilidad para el proceso de generación de código de la experiencia. En promedio, la generación automática de código fue culminada con una tasa de éxito del 85 %, en un tiempo de 3.6 minutos, y con un puntaje ASQ de 2.20.

Table 3. MEDICIONES DE USABILIDAD PARA EL PROCESO DE GENERACIÓN DE CÓDIGO.

Escenario	Tasa de Éxito Promedio	Tiempo de Finalización Promedio	Satisfacción Promedio
Generación de Código	85 %	3.6 min.	2.20

A partir de estos resultados, podemos notar que la tasa de éxito resulta igual de satisfactoria que la tasa de éxito obtenida para el proceso de modelado. Sin embargo, creemos que la tasa de éxito para el proceso de generación de código pudo haber obtenido un mejor porcentaje, ya que ésta se vio directamente afectada por el modelo de entrada utilizado. Si bien se generaron líneas de código a partir de todos los modelos, éstas no fueron completas, debido a las imperfecciones de los modelos desarrollados por los alumnos, utilizados como entrada para el generador de código. Destacamos que se realizaron correcciones in situ de los modelos elaborados, pero no se llegaron a detectar todas las correcciones necesarias por la limitación de tiempo en la sesión de trabajo.

En relación al tiempo de generación de código, podemos observar que se obtuvo un tiempo bastante reducido. Este tiempo podría generalizarse para la generación de todo tipo de aplicaciones ya que requiere de pasos estrictamente mecánicos y predefinidos, sin gran variación en la intervención del desarrollador. Además, este tiempo es independiente del tamaño de la aplicación a generar, debido a que un aumento en el tamaño de la aplicación no generaría un incremento significativo en el tiempo.

En cuanto al puntaje ASQ obtenido, podemos apreciar que es ligeramente superior al del proceso de modelado, indicando un mayor nivel de satisfacción para el proceso de generación de código.

Por lo tanto, podemos concluir que el proceso de generación de código se realizó con una buena eficacia, eficiencia y satisfacción.

4.4. Amenazas a la Validez

En cuanto a la validez interna, que tiene que ver con el grado de confianza en una relación causa-efecto entre los factores de interés y los resultados observados, se puede comentar que la experiencia de validación se llevó a cabo con alumnos, los cuales, todos poseen el mismo nivel de experiencia en cuanto a un proceso MDD, evitando de esta forma participantes con conocimientos desbalanceados. Para evitar plagios los alumnos fueron supervisados y se les prohibió la comunicación entre pares.

La validez externa, que representa el grado hasta el que los resultados alcanzados pueden generalizarse se ve afectada por el hecho de que la experiencia fue desarrollada con alumnos, los cuales no nos permiten asegurar que los resultados puedan ser generalizados a una población objetivo correspondiente a los desarrolladores de aplicaciones web que utilizan MDD. Además, la cantidad de alumnos involucrados en el proyecto fue de cinco alumnos, que si bien, no corresponde a una cantidad suficiente para fines estadísticos, alcanza al menos para sacar primeros juicios, que luego deben ser corroborados con experimentos y/o experiencias con mayor cantidad de personas para poder generalizarlos con precisión. Además el caso desarrollado consistió en un caso acotado, no obstante, se ha buscado un caso que contemple el desarrollo de una RIA que tenga en cuenta todas las características en cuestión.

En cuanto a la validez del constructo, que refleja hasta qué punto las medidas que se han realizado se adecuan a lo que el investigador tiene en mente y a lo que se está investigando, se han seleccionado datos que normalmente son utilizadas para medir los aspectos de calidad en cuestión. Además se han utilizado cuestionarios estándares, que son considerados confiables y válidos [28], [29] para evaluar la experiencia de los alumnos sin la intervención del investigador.

En relación a la fiabilidad, que indica la dependencia de los datos y su análisis respecto de un investigador específico y la capacidad de replicar el mismo estudio y obtener los mismos resultados, se ha respetado la literalidad de los datos obtenidos, tanto de la documentación, de los tiempos medidos y de los cuestionarios, evitando la introducción de sesgos a través de la interpretación.

5. Conclusión

En este trabajo presentamos una solución basada en MDD para el desarrollo de RIA, y de manera más general, para aplicaciones Web enriquecidas, incorporando las características de distribución de datos, distribución de lógica de negocios, comunicación asíncrona entre cliente y servidor, y a la vez, adoptando el estándar de MDA.

La propuesta se basa en MoWebA y extiende sus modelos y herramientas a través de un Architecture Specific Model (ASM). Para la generación del código correspondiente a la RIA final, a diferencia de las demás propuestas existentes, se han adoptado tecnologías basadas en *scripting* (HTML5, Javascript, jQuery, jQuery Datatables y jQuery UI), que son las más comúnmente utilizadas para el desarrollo de RIA.

La validación preliminar que hemos realizado, considerando tanto el proceso de modelado como la posterior generación de código de la aplicación Web enriquecida resultante, arroja resultados positivos en relación a la realización de las tareas de modelado, los tiempos de desarrollo, y la satisfacción de los usuarios en relación a la propuesta. Queda pendiente realizar ulteriores validaciones de la propuesta, como experimentos formales o casos de estudio en un contexto industrial o comercial.

References

[1] P. Fraternali, G. Rossi, and F. Sánchez-Figueroa, "Rich internet applications," *Internet Computing, IEEE*, vol. 14, no. 3, pp. 9–12, 2010.

[2] M. Busch and N. Koch, "Rich internet applications. state-of-the-art," Ludwig-Maximilians-Universität München, München, Germany, techreport 0902, 2009.

[3] G. Toffetti, S. Comai, J. C. Preciado, and M. Linaje, "State-of-the-art and trends in the systematic development of rich internet applications," *Journal of Web Engineering*, vol. 10, no. 1, pp. 070–086, 2011.

[4] M. Brambilla, J. Cabot, and M. Wimmer, "Model-driven software engineering in practice," *Synthesis Lectures on Software Engineering*, vol. 1, no. 1, pp. 1–182, 2012.

[5] OMG, "Mda specifications," available at http://www.omg.org/mda/specs.htm. [Online]. Available: http://www.omg.org/mda/specs.htm

[6] M. González, L. Cernuzzi, and O. Pastor, "A navigational role-centric model oriented web approach-moweba," *International Journal of Web Engineering and Technology*, vol. 11, no. 1, pp. 29–67, 2016.

[7] I. López, M. González, N. Aquino, and L. Cernuzzi, "Una propuesta basada en model driven architecture para el soporte de rich internet applications," in *Memorias de la XIX Conferencia Iberoamericana de Software Engineering*, Ecuador, Quito, Apr. 2016, pp. 25–38.

[8] S. Meliá, J. Gómez, S. Pérez, and O. Díaz, "Architectural and technological variability in rich internet applications," *IEEE Internet Computing*, vol. 14, no. 3, pp. 24–32, 2010.

[9] A. Bozzon, S. Comai, P. Fraternali, and G. T. Carughi, "Conceptual modeling and code generation for rich internet applications," in *Proceedings of the 6th international conference on Web engineering*. ACM, 2006, pp. 353–360.

[10] M. Linaje, J. C. Preciado, and F. Sánchez-Figueroa, "A method for model based design of rich internet application interactive user interfaces," in *Web Engineering*. Springer, 2007, pp. 226–241.

[11] J. C. Preciado, M. Linaje, S. Comai, and F. Sanchez-Figueroa, "Designing rich internet applications with web engineering methodologies," in *Web Site Evolution, 2007. WSE 2007. 9th IEEE International Workshop on*. IEEE, 2007, pp. 23–30.

[12] G. T. Carughi, S. Comai, A. Bozzon, and P. Fraternali, "Modeling distributed events in data-intensive rich internet applications," in *Web Information Systems Engineering–WISE 2007*. Springer, 2007, pp. 593–602.

[13] M. Brambilla, J. C. Preciado, M. Linaje, and F. Sanchez-Figueroa, "Business process-based conceptual design of rich internet applications," in *Web Engineering, 2008. ICWE'08. Eighth International Conference on*. IEEE, 2008, pp. 155–161.

[14] F. Valverde and O. Pastor, "Facing the technological challenges of web 2.0: A ria model-driven engineering approach," in *International Conference on Web Information Systems Engineering*. Springer, 2009, pp. 131–144.

[15] M. Huber and P. Brune, "Model-driven development of interactive web user interfaces with html5." in *MODELSWARD*, 2013, pp. 249–252.

[16] J. L. H. Agustin and P. C. Del Barco, "A model-driven approach to develop high performance web applications," *Journal of Systems and Software*, vol. 86, no. 12, pp. 3013–3023, 2013.

[17] R. Esbai, M. Erramdani, and S. Mbarki, "Model-driven transformation for gwt with approach by modeling: From uml model to mvp web applications," *International Review on Computers and Software (I. RE. CO. S.)*, vol. 9, no. 9, pp. 1612–1620, 2014.

[18] M. L. Bernardi, G. A. Di Lucca, and D. Distante, "Model-driven fast prototyping of rias: From conceptual models to running applications," in *Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on*. IEEE, 2014, pp. 250–258.

[19] E. Yigitbas, B. Mohrmann, and S. Sauer, "Model-driven ui development integrating hci patterns." *LMIS@ EICS*, vol. 2015, pp. 42–46, 2015.

[20] R. Esbai and M. Erramdani, "Model-to-model transformation in approach by modeling: From uml model to model-view-presenter and dependency injection patterns," in *Information and Communication Technologies (WICT), 2015 5th World Congress on*. IEEE, 2015, pp. 1–6.

[21] S. Roubi, M. Erramdani, and S. Mbarki, "A model driven approach to generate graphical user interfaces for rich internet applications using interaction flow modeling language," in *Intelligent Systems Design and Applications (ISDA), 2015 15th International Conference on*. IEEE, 2015, pp. 272–276.

[22] J. L. H. Agustin, "Model-driven web applications," in *Science and Information Conference (SAI), 2015*. IEEE, 2015, pp. 954–964.

[23] S. Roubi, M. Erramdani, and S. Mbarki, "Modeling and generating graphical user interface for mvc rich internet application using a model driven approach," in *Information Technology for Organizations Development (IT4OD), 2016 International Conference on*. IEEE, 2016, pp. 1–6.

[24] N. Laaz and S. Mbarki, "A model-driven approach for generating ria interfaces using ifml and ontologies," in *Information Science and Technology (CiSt), 2016 4th IEEE International Colloquium on*. IEEE, 2016, pp. 83–88.

[25] G. Nuñez, M. González, and L. Cernuzzi, "Un enfoque MDD para el desarrollo de RIA," Proyecto final, Universidad Católica "Nuestra Señora de la Asunción", 2017, available at http://www.dei.uc.edu.py/proyectos/mddplus/documentos/.

[26] S. ISO, "9241-11. 1998," *Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs)–Part II Guidance on Usability*, 1998.

[27] V. R. Basili and H. D. Rombach, "The tame project: Towards improvement-oriented software environments," *IEEE Transactions on software engineering*, vol. 14, no. 6, pp. 758–773, 1988.

[28] J. R. Lewis, "Psychometric evaluation of an after-scenario questionnaire for computer usability studies: the asq," *ACM SIGCHI Bulletin*, vol. 23, no. 1, pp. 78–81, 1991.

[29] J. Sauro, "Measuring usability with the system usability scale (sus)," 2011.