Desarrollo Dirigido por Modelos Basado en Componentes de Interfaz de Usuario

Doctorado en Ciencias Informáticas
Facultad de Informática - Universidad Nacional de La Plata

Autor: Pablo Martín Vera
GIDFIS (Grupo de Investigación, Desarrollo y Formación en Innovación de Software)
Departamento de Ingeniería e Investigaciones Tecnológicas Universidad Nacional de La Matanza. pvera@unlam.edu.ar

Directora: Claudia Pons
LIFIA (Laboratorio de Investigación y Formación en Informática Avanzada), Facultad de Informática, Universidad Nacional de La Plata. cpons@info.unlp.edu.ar

Co Directora: Carina González González
Universidad La Laguna, La Laguna, España. cjgonza@ull.es

Fecha de Exposición: 14 de Septiembre de 2015

Resumen

Esta tesis presenta una metodología de modelado para aplicaciones web móviles utilizando técnicas de desarrollo dirigido por modelos (MDD). Mediante la creación de sólo dos diagramas, un diagrama de datos y un diagrama de interfaz de usuario (que además incluye la navegación) es posible definir el comportamiento completo de una aplicación.

Por estar esta metodología basada en MDD incorpora dos transformaciones; la primera desde el modelo de datos a una versión inicial del modelo de interfaz de usuario, lo que reduce considerablemente el esfuerzo de modelado, ya que luego ese segundo modelo solo deberá ser adaptado a las necesidades particulares. La segunda transformación toma los modelos realizados y genera el código fuente completo, 100% funcional de una aplicación web móvil, además del script de la base de datos correspondiente. Ambos modelos están basados en una extensión conservativa de UML. El modelo de datos está basado en el diagrama de clases y el modelo de interfaz de usuario utiliza el diagrama de componentes de UML. Para poder especificar el comportamiento de la interfaz de usuario se definen una serie de componentes que a su vez pueden ser configurados con información tomada del modelo de datos. La configuración se basa en valores etiquetados propios para cada tipo de componente. Para facilitar el proceso de construcción de los modelos y su configuración se ha desarrollado una herramienta de soporte, que permite no solo modelar, sino también realizar las transformaciones establecidas en la metodología, obteniendo como resultado final una aplicación funcional sin escribir una sola línea de código.

Además esta tesis establece las ventajas de utilizar componentes configurables en el desarrollo dirigido por modelos, haciendo que el esfuerzo de programación se realice una única vez al establecer las transformaciones y que luego pueda ser aplicado a una amplia gama de aplicaciones de distintos dominios.

Palabras Clave: MDD, UML, COMPONENTES, MÓVIL, INTERFAZ DE USUARIO

1. Definición del Problema

El modelado de aplicaciones es un área subestimada por la industria de software. Especialmente en pequeñas y medianas empresas donde el modelado es considerado, en numerosas oportunidades, una pérdida de tiempo. En otros casos, los modelos solo se utilizan en etapas tempranas del desarrollo para la toma de requerimientos o como documentación inicial que luego no se actualiza con los cambios realizados durante el desarrollo de la solución, por lo que rápidamente quedan obsoletos. Por esa razón surge la idea de dar más importancia a los modelos. Los modelos pueden ser utilizados para generar de forma automática una aplicación o por lo menos parte de ella. "La automatización del proceso de desarrollo de software consiste en comenzar desde un alto nivel de representación de las características deseadas del software y derivar una aplicación ejecutable a partir de ella, posiblemente mediante un conjunto de pasos intermedios que permitan algún grado de interacción del usuario con el proceso de generación" [1]

El desarrollo de software mediante la construcción de modelos que permitan luego la generación automática de código fuente a partir de los mismos, es una tendencia iniciada hace varios años atrás. Puede encontrarse con diversos nombres: MDD (Model Driven Development), MDA (Model Driven Architecture), MDSE (Model Driven Software Engineering), MDE (Model Driven Engineering), entre otras. Sin embargo estas siglas tienen una categorización y relación entre ellas pero lo importante es que todas tienen algo en común, utilizan modelos y transformaciones.

Un modelo es una representación abstracta de la realidad, en este caso los modelos que utilizaremos son modelos software y por lo tanto podrán representar partes o componentes de un sistema. Un modelo puede tener una representación gráfica y también una descripción textual, lo importante es que siga ciertas reglas para su conformación.

Una transformación es un proceso que toma como entrada un modelo y genera como resultado otro modelo o código fuente. Por ejemplo el enfoque MDA de la OMG [2] [3], utiliza diferentes tipos de modelos con diferentes niveles de abstracción. Partiendo de modelos independientes de la plataforma (PIM) hasta llegar a modelos específicos para cada plataforma (PSM). "El PIM permite una representación visual del modelo, utilizando un nivel alto de abstracción. Los detalles de los modelos ambientales pueden ser expresados y precisados claramente en UML sin utilizar ningún formalismo particular de la implementación… un PSM se desarrolla mediante un mapeo de un PIM a una plataforma computacional y a un lenguaje de programación específico" [4].

Para llegar desde el PIM al PSM se deben realizar transformaciones automáticas o semi-automáticas. El objetivo final de estas técnicas es automatizar el proceso de generación del código fuente permitiendo que los diseñadores se centren en los modelos, más que en proceso de codificación.

Sin embargo, el principal problema, es que la mayoría de las técnicas existentes son difíciles de utilizar y requieren de un arduo proceso detallando modelos y configurando transformaciones para lograr obtener código fuente utilizable y aun así la mayoría de ellas solo permite generar parte del código fuente. Algunos autores comparten la visión de que las metodologías de modelado para generación de código fuente son demasiado complejas "Para realizar un programa útil partiendo de un modelo, el modelo debe ser tan complicado que la gente que no puede programar no lo puede comprender y los que pueden programar preferirán escribir el código antes que realizar los correspondientes modelos" [5].

Es por este motivo que surge la necesidad de crear una metodología de modelado fácil de utilizar, que no requiera de modelos complejos y que como resultado se pueda obtener el código fuente completo de una aplicación.

2. Hipótesis

Es posible desarrollar una metodología de modelado que incluya toda la información necesaria para generar el código fuente completo de una aplicación realizando transformaciones automáticas, con pocos modelos, simples y fácilmente entendibles.

3. Contribución Científica

- Utilización de componentes configurables para facilitar la generación de código fuente en MDD
- Diseño de una metodología para el modelado de aplicaciones web móviles centradas en los datos.

4. Solución Propuesta

Como solución se plantea el diseño de una metodología de modelado basada en componentes de interfaz de usuario configurables que incluye toda la información necesaria para generar el código fuente completo de una aplicación. Esta metodología se denomina CBMDD (Component Based Model Driven Development)

Los pasos de la metodología pueden verse en la Figura 1 y son los siguientes:

1. El diseñador realiza el modelo de datos del sistema.
2. Tomando como base el modelo de datos, la herramienta de transformación genera de forma automática una primera versión del modelo de interfaz.
3. El diseñador realiza los ajustes al modelo de interfaz, cambiando la configuración y/o agregando nuevos componentes.
4. Tomando los dos modelos terminados, se realiza la segunda y última transformación que genera el script de la base de datos y el código fuente de la aplicación.

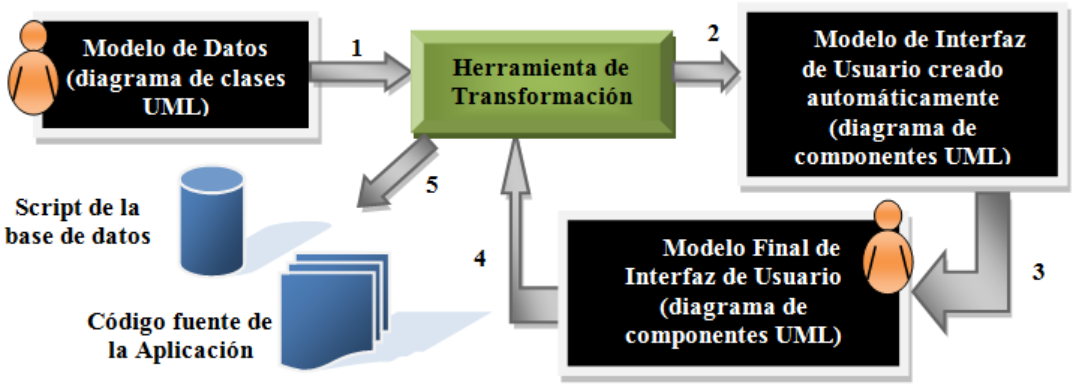

Fig 1. Pasos de la metodología de modelado

Ambos diagramas están basados en diagramas tradicionales de UML pero extendidos mediante estereotipos y valores etiquetados. Todas las extensiones realizadas se definen mediante un perfil de UML. Todos los nuevos estereotipos definidos heredan de alguna metaclase de UML generando de esta forma una extensión conservativa. "Conservativa significa que los elementos de modelo del metamodelo de UML no se modifican… todos los nuevos elementos se relacionan por herencia hacia al menos un elemento del modelo del metamodelo de UML" [6].

4.1. Modelo de Datos

El primer paso es realizar el modelo de datos de la aplicación, para ello se utiliza el diagrama de clases de UML donde cada clase representa una tabla de la base de datos. En este diagrama se define información adicional que luego facilitará la creación de la base de datos y la visualización de información en el sistema. Esta información adicional se expresa en el diagrama de clases mediante estereotipos que permiten determinar la clave primaria de una clase utilizando el estereotipo Identifier y una descripción textual de dicha clase mediante el estereotipo Descriptor. Adicionalmente se utiliza el estereotipo enumeration ya presente en UML para definir por ejemplo distinto estados de un modelo.

4.2. Modelo de Interfaz de Usuario

En este modelo se diseñaran las distintas pantallas que conformaran el sistema, determinando además, cuál es el flujo de navegación entre dichas pantallas. Para modelar la interfaz de usuario se utilizará el diagrama de componentes de UML, donde cada componente se corresponderá con una pantalla del sistema.

Para identificar el tipo de pantalla que se desea modelar se crearon estereotipos que se aplican a los componentes de UML para tipificarlos. Los estereotipos disponibles son: Login, List, Search, Menu, CRUD y UpdateView.

Cada tipo de componente tiene definido una serie de valores etiquetados que permite configurar su comportamiento para adaptarlo a la aplicación que se desea modelar. Algunos de estos valores etiquetados son comunes para la mayoría de los componentes y otros son particulares para cada uno de ellos. El detalle de configuración de cada componente y de sus valores etiquetados puede verse en [7].

Existen algunas opciones de configuración que son comunes como ser un título general del componente, que se utilizará para mostrar al usuario en la pantalla que se encuentra y la barra de navegación que está presente en todos los componentes que se configura mediante una serie de links que llevan a otros componentes. Además cada componente incluye opciones de configuración propia que permiten definir cómo será la interfaz de usuario. A continuación se enumeran dichas opciones de configuración y su muestran pantallas generadas automáticamente por la herramienta de modelado de CBMDD.

- Login: Este componente permite la autenticación de un usuario en el sistema. Además de permitir configurar contra que tabla se realizará la autenticación, incluye la posibilidad de permitir que el usuario en lugar de ser ingresado, sea seleccionado, esto es útil en los dispositivos móviles ya que el ingreso de texto es una tarea dificultosa. La Figura 2 muestra dos pantallas de login, una en la cual el usuario debe ser ingresado y la otra donde directamente se selecciona.

Fig. 2. Pantallas generadas automáticamente a partir de componentes del tipo Login

- List: Este componente muestra listados de información. Su configuración permite: (a) Definir qué datos se van a mostrar; (b) Definir datos adicionales que se pueden mostrar en una segunda fila de información correspondiente a cada ítem. (c) Establecer filtros por defecto que se aplican al listado visualizado; (d) Determinar el orden del listado.
  La Figura 3 muestra dos ejemplos de pantallas generadas a partir de componentes del tipo List. La imagen de la izquierda muestra un listado donde la información se presenta en varias líneas, una línea principal con un título y una línea adicional mostrando información complementaria, este caso con la categoría y fecha de vencimiento. En el lado derecho de la imagen se ve un listado con un solo dato mostrando la descripción de las categorías, pero en la barra de navegación además de la opción para volver al menú principal hay un botón para ir a la pantalla que permite agregar una nueva categoría.

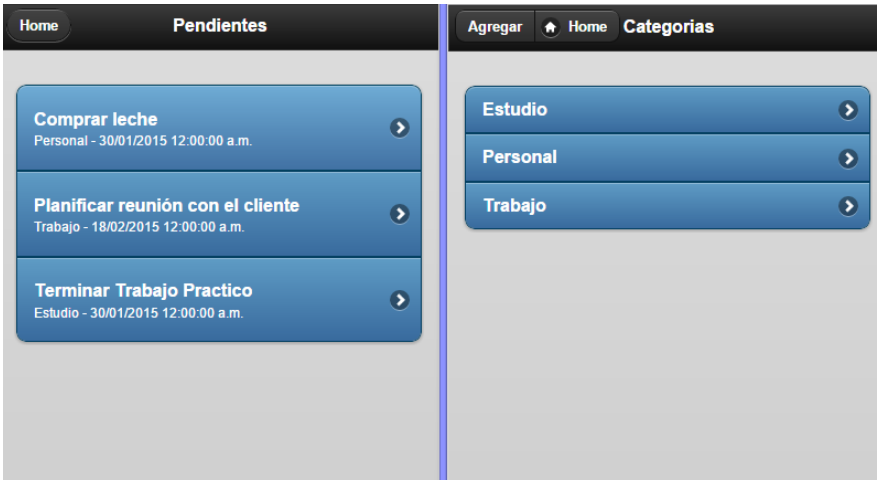

Fig. 3. Pantallas generadas automáticamente a partir de componentes del tipo List

- Search: es un componente que permite mostrar un listado de información pero el mismo puede ser filtrado en tiempo de ejecución por el usuario. Incluye la misma configuración del componente List pero capacidad de definir sobre que campos se permitirá realizar la búsqueda y el tipo de control asociado al mismo. Los tipos de filtros disponibles a aplicar son: (a) Texto Libre; (b) Selección Simple; (c) Selección Múltiple; (d) Rango de Fechas; (e) Verdadero o Falso.
  La Figura 4 muestra un ejemplo de los filtros de búsqueda que se generan en la interfaz de la aplicación.

Fig. 4. Pantalla de búsqueda generada automáticamente a partir de un componente Search

- Menu: este componente permite mostrar al usuario una pantalla con un menú. Su configuración es simple ya que solo se deben definir las opciones de dicho menú mediante una serie de links que apuntan a otros componentes tal como puede verse en la Figura 5.

Fig. 5. Pantalla generada automáticamente a partir de un componente del tipo Menu

- CRUD: es un componente para crear, modificar, visualizar y eliminar datos. Por defecto este componente solicita al usuario el ingreso de todas las propiedades de la clase que se está editando, pero incorpora una potente configuración que permite:
  - Definir valores por defecto al crear o actualizar un objeto
  - Evitar completar determinadas propiedades de una clase al crear o actualizar un objeto
  - Crear registros adicionales de otras tablas al crear o actualizar un objeto, lo que es muy útil para crear tablas de log.
  - Definir propiedades opcionales

  La Figura 6 muestra una pantalla de edición donde para cada propiedad se generan automáticamente los controles adecuados para que el usuario pueda ingresar los datos. Puede verse que para clases relacionadas se genera un control de selección con la descripción de la otra entidad como es el caso de la categoría.

Fig. 6. Pantalla de edición generada automáticamente a partir de un componente del tipo CRUD

- UpdateView: este componente permite realizar operaciones de actualización sobre objetos incorporando características especiales que lo diferencian del CRUD. En muchos sistemas es posible que un único objeto deba ser actualizado parcialmente donde solo algunas de sus propiedades se deban completar y otras sean mostradas en modo de solo lectura. Este control es muy útil para sistemas donde un objeto va pasando por diferentes estados y en cada estado se registran distintos valores. Para actualizar cada uno de esos estados podría configurarse un componente UpdateView diferente. La Figura 7 muestra un ejemplo de una pantalla generada por un componente UpdateView donde se configuraron 3 propiedades de solo lectura (título, categoría y fecha de vencimiento) y solo se permite editar al usuario la descripción. En este caso además al actualizar los datos el estado de la tarea cambia automáticamente a Completa.

Fig. 7. Pantalla generada automáticamente a partir de un componente del tipo UpdateView

4.3. Herramienta de Modelado

La metodología de modelado fue desarrollada basada en UML utilizando una extensión conservativa mediante estereotipos y valores etiquetados, esto hace que sea posible crear los modelos en cualquier herramienta basada en UML ya que los mecanismos de extensión están presentes en todas ellas. Sin embargo la configuración de cada valor etiquetado es particular y requiere recordar la sintaxis para realizar una configuración correcta. Este es un primer motivo por el cual se decidió desarrollar una herramienta que permita facilitar dicha configuración, el segundo motivo es que dicha herramienta dará soporte además a las trasformaciones entre modelos.

La herramienta es una aplicación Web desarrollada en C# bajo el framework Microsoft ASP.NET V4 utilizando Web Forms [8]. La misma fue desarrollada con los siguientes objetivos:

1. Facilitar el modelado evitando que el diseñador deba recordar la configuración particular de cada componente.
2. Permitir la interoperabilidad con otras aplicaciones de modelado mediante XMI.
3. Validar la correcta configuración de los componentes en el caso de que se realice con una herramienta externa.
4. Realizar las transformaciones automáticas previstas en la metodología.

Además la herramienta fue desarrollada para ser escalable permitiendo:

- Agregar fácilmente nuevos componentes.
- Permitir reutilizar controles para configurar los componentes.
- Poner a disposición clases para generar fácilmente código fuente en cualquier lenguaje.

5. Resultados

Se desprende de las secciones anteriores que mediante la herramienta construida es posible realizar los modelos y aplicar las transformaciones. Generándose los scripts de la base de datos y el código fuente de la aplicación. Por otra parte al comenzar un proyecto la herramienta muestra los estados de los diagramas (por ejemplo: diagrama de clases – sin datos; Generar Componentes – no generado; etc.). Estos estados cambian a medida que se trabaja con la herramienta. También permite evitar por ejemplo que el usuario cambie el diagrama de clases y no vuelva a generar el de componentes. Permitiendo que exista una trazabilidad entre los modelos y transformaciones aplicadas (ver figura 8).

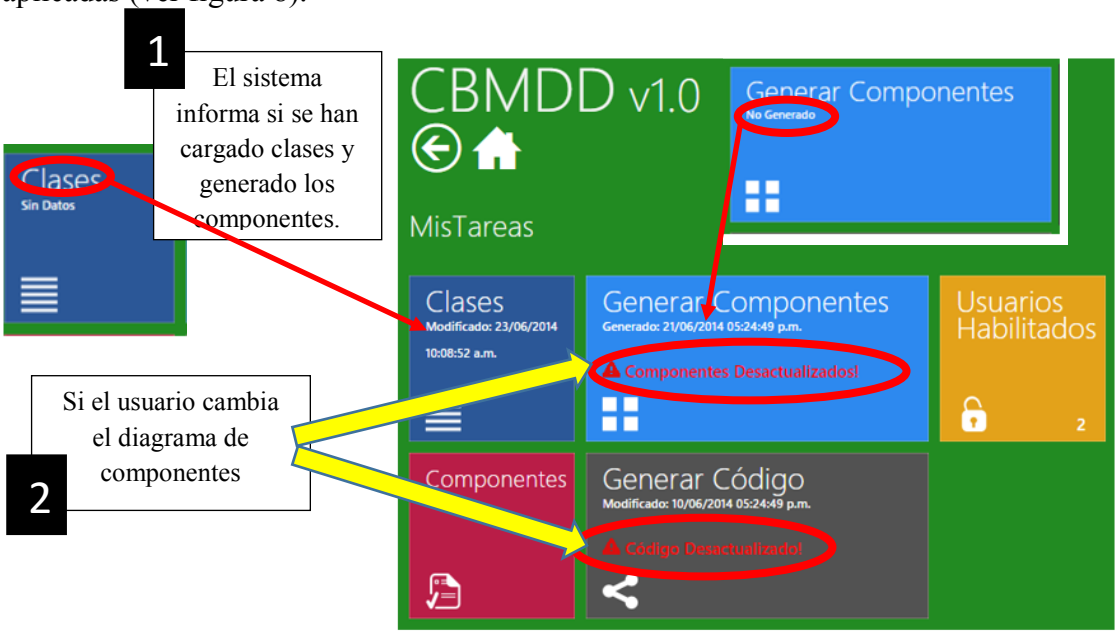

Fig.8. Menú de acciones de un proyecto de modelado

Se han realizado diversas aplicaciones con distintos grados de complejidad pudiendo generarse el código fuente funcional de ellas a través de la herramienta, también se ha trabajado en la interoperabilidad permitiendo que pueda exportarse el XMI del modelo y luego importarse en otra herramienta de modelado. Para esto se creó un parser de XMI 2.2 que brinda interoperabilidad con el EA (Enterprise Architecture) permitiendo importar en la herramienta los modelos previamente construidos EA o exportarlos al EA.

Las opciones para importar y exportar modelos mediante XMI se agregaron tanto para el modelo de datos como para el modelo de componentes. En el caso del proceso de importación también se realiza una validación para comprobar que el modelo que se está importando este correctamente configurado.

6. Conclusiones

El empleo de MDD permite poner el foco de atención en las primeras etapas haciendo que los modelos sean la parte más importante, pudiendo transformarse y evolucionar hasta obtener el código fuente de la aplicación. Es claro que el esfuerzo entonces no está puesto en la programación sino en el diseño. En contrapartida, algunas metodologías de MDD requieren realizar gran cantidad de modelos con inter-relaciones complejas, en donde el esfuerzo final utilizado en las etapas de modelado termina siendo equiparado con el que se requeriría para desarrollar una aplicación de cero de forma tradicional. La presente tesis, plantea la importancia de centrarse en los modelos y como principal contribución teórica se construye una metodología que extiende de forma conservativa a

UML, mediante la cual con 2 modelos es posible efectuar transformaciones automáticas de forma transparente al usuario final y obtener el código fuente funcional de una aplicación. Esta metodología se planteó considerando diversos aspectos: (1) reutilización de componentes, (2) facilidad en el modelado, (3) extensión conservativa de uml y (4) completitud de la información.

Además se desarrolló una herramienta que permite implementar la metodología efectivamente, chequear la trazabilidad entre los modelos generados, importar modelos desde otra herramienta de modelado facilitando la interoperabilidad, efectuar las transformaciones automáticas y generar finalmente como resultado tanto los scripts de la base de datos como el código fuente de la aplicación.

La herramienta desarrollada facilita aún más la tarea de modelado ya que evita que el diseñador deba lidiar con la sintaxis concreta de la configuración de los componentes brindando interfaces gráficas de configuración y generación de los modelos. Por otra parte, incorpora los procesos de transformación necesarios en la metodología.

Cabe destacar que la herramienta de soporte fue desarrollada en forma modular permitiendo rápidamente la incorporación de nuevos componentes y nuevos valores etiquetados para permitir la expansión futura de la metodología. También brinda clases para facilitar la tarea de la programación de las transformaciones para generar código fuente incorporando el concepto de templates lo que permite generar con un mismo modelo, código para distintas plataformas.

Si bien la metodología CBMDD ha sido planteada para dispositivos móviles es posible adaptarla a otras plataformas. En el caso de que la interfaz de usuario que se desee modelar tenga una distribución de pantalla más compleja es posible agregar nuevos componentes así como también agregar parámetros de configuración a los componentes existentes según las necesidades del caso.

7. Trabajos Futuros

- Creación de nuevos templates para generar interfaces para distintas plataformas y distintos lenguajes.
- Incorporar opciones de interoperabilidad con otras herramientas de modelado.
- Hacer más poderosos los templates de creación de código fuente incorporando parámetros a los mismos para generar interfaces con distintas opciones sin necesidad de crear un nuevo template. Por ejemplo podría seleccionar si la navegación primaria es persistente (siempre fija) o transitoria (se despliega un menú con una opción) en la pantalla. O incluso permitir seleccionar imágenes para que el menú principal sea icónico en lugar de textual (patrón Springboard).
- Incorporación de nuevos controles o nuevas configuraciones para hacer más poderoso el modelo.

8. Publicaciones Efectuadas

1. Revista Colombiana de Computación (RCC) “La interfaz de usuario como punto de partida para la creación automática de aplicaciones móviles – Un enfoque basado en MDD”. 2015
2. International Journal of Information Technologies and Systems Approach (IJITSA) - Volúmen 8, Número 2, pp 80 – 100. “Component Based Model Driven Development – An Approach for Creating Mobile Web Applications from Design Models”. Special Issue on HCI 2015
3. XX Congreso Argentino de Ciencias de la Computación (CACIC). “Automatic Creation of Mobile Web Applications from Design Models“. Universidad Nacional de La Matanza, San Justo, Buenos Aires, Argentina. 2014
4. XVI Workshop de Investigadores en Ciencias de la Computación (WICC). “Generación Automática de Aplicaciones Web Móviles Mediante Componentes Configurables”. Universidad Nacional de Tierra del Fuego, Ushuaia, Tierra del Fuego, Argentina. 2014

5. Third International Conference on Software and Emerging Technologies for Education, Culture, Entertainment, and Commerce (SETECEC): New Directions in Multimedia Mobile Computing, Social Networks, Human-Computer Interaction and Communicability. “Tool for developing Mobile Web Application from UI Models – Based on CBHDM Methodology”. Venecia, Italia 2014
6. XIX Congreso Argentino de Ciencias de la Computación (CACIC). “Modeling Complex Mobile Web Applications from UI Components – Adding Different Roles and complex Database Design”. Universidad CAECE, Mar del Plata, Buenos Aires, Argentina 2013.
7. XLII Jornadas Argentinas de Informática (42 JAIIO). “Metodología de Modelado de Aplicaciones Web Móviles Basada en Componentes de Interfaz de Usuario”. Facultad de Matemática, Astronomía y Física de la Universidad Nacional de Córdoba, Córdoba Capital, Córdoba, Argentina. 2013
8. XV Workshop de Investigadores en Ciencias de la Computación (WICC). “Metodología de Modelado de Aplicaciones Web Móviles Basada en Componentes”. Universidad Autónoma de Entre Ríos (UADER), Paraná, Entre Ríos, Argentina. 2013
9. XVII Congreso Argentino de Ciencias de la Computación (CACIC). “MDA based Hypermedia Modeling Methodology using reusable components”. Universidad Nacional del Sur, Bahía Blanca, Buenos Aires, Argentina. 2012
10. XV Workshop de Investigadores en Ciencias de la Computación (WICC) - “Utilizando el Enfoque MDA para la Construcción de Aplicaciones Web Móviles Centradas en los Datos”. Universidad Nacional de Misiones, Posadas, Misiones, Argentina. 2012
11. VII Congreso Colombiano de Computación (7CCC). “User Interface and Navigation Modeling Methodology for Mobile Hypermedia Systems”. Medellín, Colombia.2012

Referencias

[1] Brambilla, M. C. (2012). *Model-driven software engineering in practice* (Vol. 1). Synthesis Lectures on Software Engineering.

[2] Kleppe, A., Warmer, J., & Wim, B. (2003). *MDA explained: the model driven architecture: practice and promise.* Addison-Wesley Professional.

[3] OMG. (2003). *MDA Guide Version 1.0.1.* Retrieved 11 04, 2014, from http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf

[4] Papajorgji, P., Beck, H. W., & Braga, J. L. (2004). n architecture for developing service-oriented and component-based environmental models. *Ecological Modelling, 179*(1), 61-76.

[5] Steimann, F., & Kühne, T. (2005, 12). Coding for the Code. (ACM, Ed.) *Queue - Managing Megaservice, 3*(10), 44-51.

[6] Rossi, G., et al. *Web Engineering: Modelling and Implementing Web Applications.* 2008.

[7] Vera P., Pons C. Gonzales C, Giulianelli D., Rodriguez R. *MDA based Hypermedia Modeling Methodology using reusable components.* s.l. : XVIII Congreso Argentino de Ciencias de la Computación, 2012.

[8] Microsoft. *ASP.NET Web Forms.* [Online] [Cited: 01 05, 2015.] http://www.asp.net/web-forms.