MDD+

Mejorando el Proceso de Desarrollo de Software: Propuesta basada en MDD

Nro. Referencia: 14-INV-056

Investigador Principal: Luca Cernuzzi, Universidad Católica “Nuestra Señora de la Asunción”

Plataformas de Desarrollo para la Etapa de Implementación

Recopilación del Documento: Magalí González, Universidad Católica “Nuestra Señora de la Asunción”

Noviembre – 2016

Este proyecto es financiado por el CONACYT a través del programa PROCIENCIA con recursos del Fondo para la Excelencia de la Educación e Investigación (FEEI) del FONACIDE.

Tabla de contenido

1. Introducción

Este documento presenta una visión general sobre las plataformas de desarrollo para la implementación de herramientas relacionadas a la propuesta basada en Model Driven Development (MDD) para la mejora del proceso de desarrollo de software.

Por una parte, se presenta el proceso de generación de código para tecnologías actuales, como las Rich Internet Applications (RIAs) y las plataformas móviles, junto con una descripción de las arquitecturas, lenguajes y frameworks a ser utilizados para la implementación de herramientas.

Por otra parte, se presenta información similar para el proceso de generación de casos de prueba, siguiendo los enfoques de MDD y Test Driven Development (TDD).

2. Proceso de generación de código para tecnologías actuales

El proceso de generación de una aplicación adoptando la propuesta metodológica de MoWebA [1] consiste en realizar los siguientes pasos:

a) Definición del Modelo Independiente de Plataforma (PIM). Se definen modelos independientes siguiendo el enfoque MoWebA. Esto se realiza utilizando la herramienta MagicDraw[1] y adoptando los perfiles propios de MoWebA.

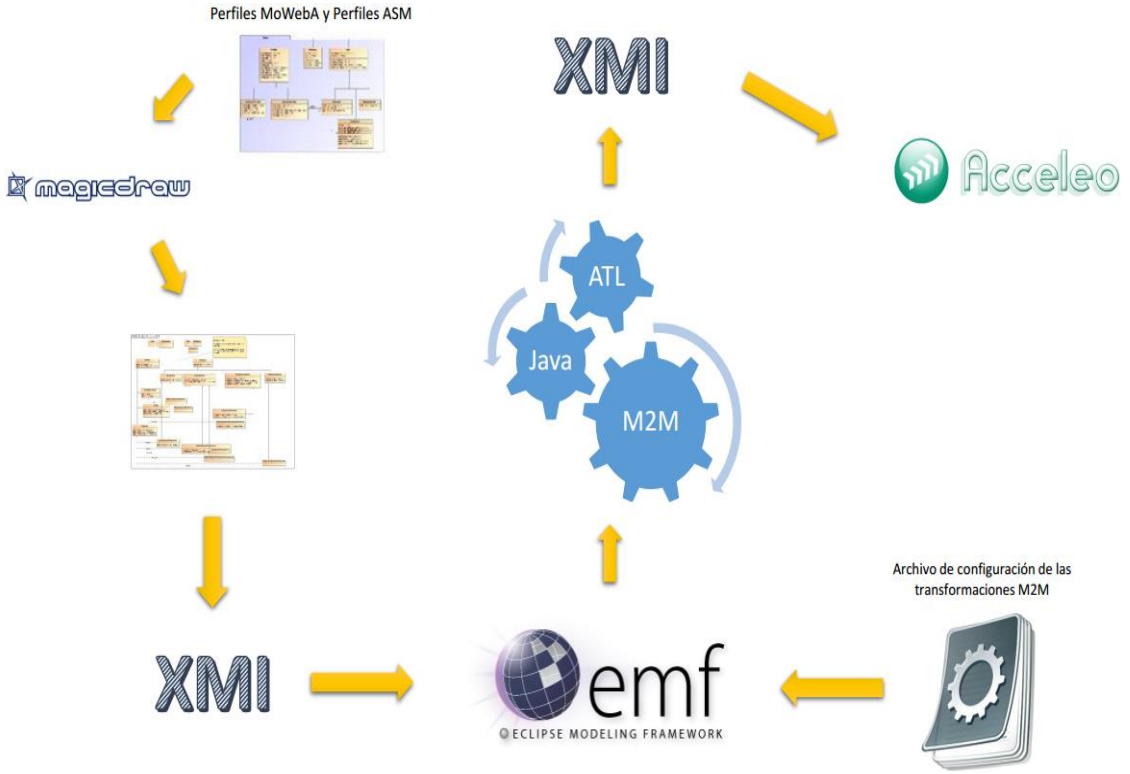

Figura 1. Esquema de transformación modelo a modelo, de PIM a ASM

[1] https://www.nomagic.com/products/magicdraw.html

b) Transformación del PIM al Modelo Específico de Arquitectura (ASM). Esto se realiza siguiendo el esquema de transformación que se presenta en la Figura 1. Para generar el modelo ASM se debe especificar la arquitectura destino de la aplicación. En el marco del proyecto se tienen dos arquitecturas posibles: RIAs y arquitecturas móviles (en la capa de persistencia y servicios en la nube). Es importante destacar que en el marco del proyecto se han definido los perfiles PIM, el ASM para RIA, persistencia y servicios en la nube para dispositivos móviles. El proyecto además contempla la definición de las reglas de transformación de modelo a modelo del PIM al ASM para la arquitectura RIA utilizando el lenguaje ATL[2].

c) Ajustes manuales al ASM y generación de código. Una vez generados los ASMs en forma automática, los diseñadores deberán realizar ajustes manuales al modelo obtenido, y luego aplicar las reglas de transformación para la obtención del código final. El proceso de generación de código se realiza utilizando la herramienta Acceleo[3] y las reglas de transformación de modelo a código para la arquitectura especificada. Si la arquitectura seleccionada es RIA, el proceso se realiza conforme a la Figura 2. En caso de haber seleccionado la arquitectura móvil, se deberá indicar si corresponde a la capa de persistencia, lo cual implicará seguir el esquema de la Figura 3, o bien la de servicios en la nube, que llevará al proceso indicado en la Figura 4. Para todas las arquitecturas, las reglas de transformación de modelo a código se definen usando la herramienta Acceleo. Las plataformas destinos para la arquitectura RIA se centran en los lenguajes HTML5[4], JavaScript[5], y el framework jQuery[6] (Figura 2). La persistencia para dispositivos móviles se genera en código SQLite[7] para las plataformas Android[8] y Windows Phone[9] (Figura 3). Por último, los servicios en la nube serán desarrollados para las plataformas Android e iOS[10], utilizando la arquitectura de servicios REST (Figura 4).

Figura 2. Esquema de transformación de modelo a código para la arquitectura RIA

[2] http://www.eclipse.org/atl/
[3] http://www.eclipse.org/acceleo/
[4] https://www.w3.org/TR/html5/
[5] https://www.javascript.com/
[6] https://jquery.com/
[7] https://sqlite.org/
[8] https://www.android.com/
[9] http://www.microsoft.com/windowsphone/
[10] http://www.apple.com/mx/ios

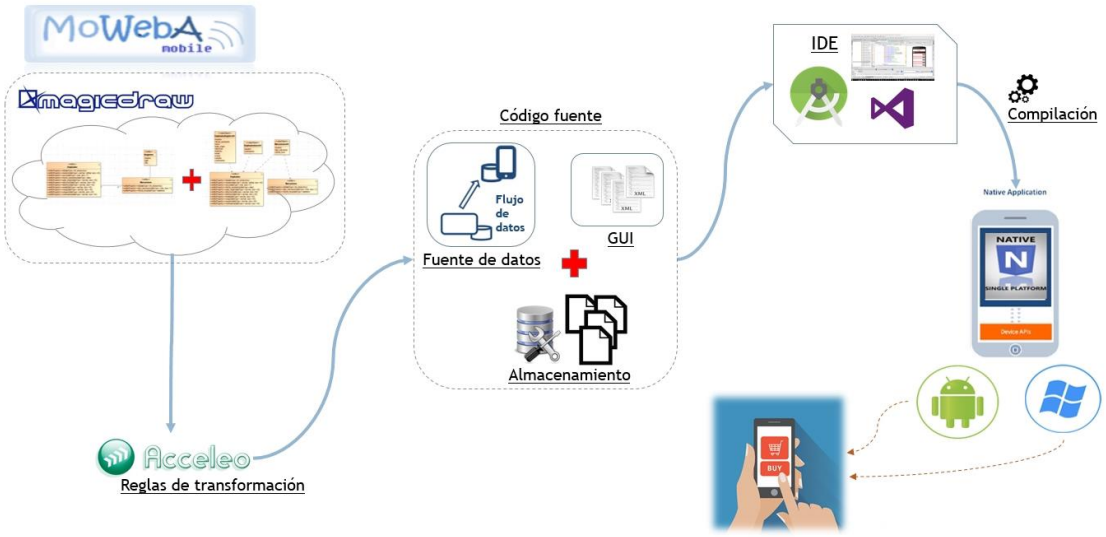

Figura 3. Esquema de transformación de modelo a código para la persistencia en dispositivos móviles

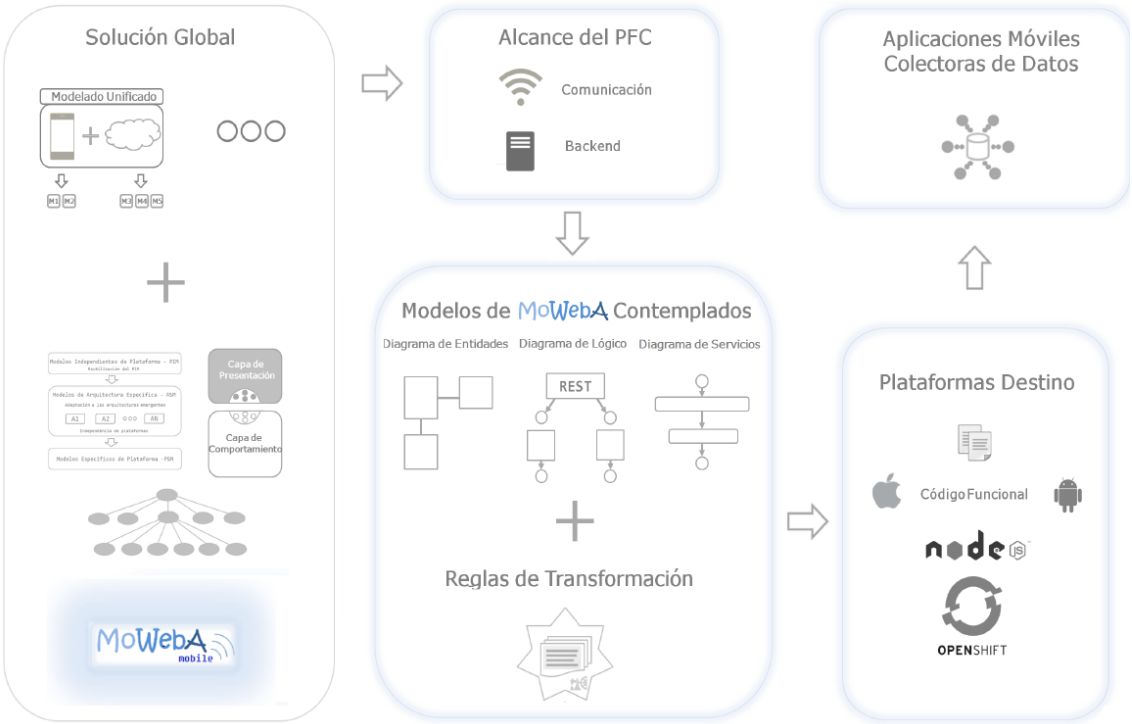

Figura 4. Esquema de transformación de modelo a código para servicios en la nube de dispositivos móviles

3. Generación de casos de prueba aplicando técnicas de Model Driven Development (MDD) y Test Driven Development (TDD)

La definición y generación de casos de prueba utilizando las herramientas a ser provistas por nuestra propuesta conlleva los pasos ilustrados en la Figura 5.

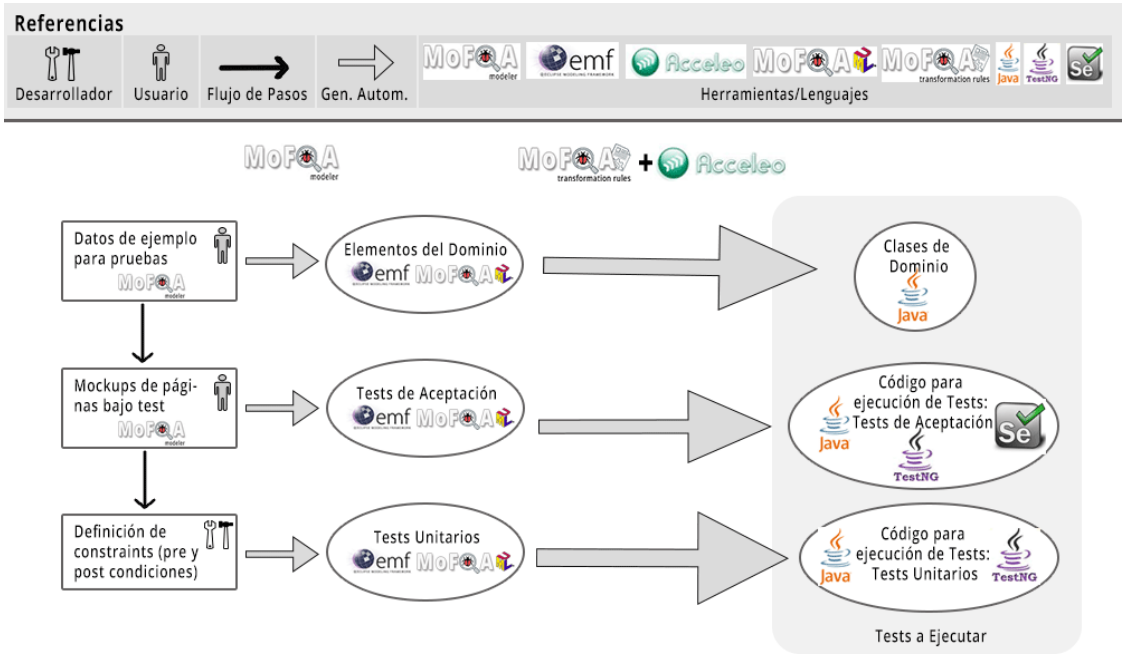

Figura 5. Esquema general para la generación de casos de prueba siguiendo los enfoques MDD y TDD

Para cada funcionalidad a implementar el usuario puede definir las páginas que deberán visualizarse al finalizar la implementación. MoFQA ofrecerá una herramienta de modelado simple (MoFQA *modeler*) para definir los siguientes elementos:

- Páginas con sus componentes (texto, campos de formularios, botones, mensajes de alerta).
- Datos de ejemplo que deberán visualizarse en los componentes de las páginas durante la ejecución de las pruebas de aceptación.
- Resultados de las interacciones con los diferentes componentes de las páginas.

MoFQA *modeler* toma los elementos definidos por el usuario y los transforma en modelos (EMF[11] versión 5, enriquecido con perfiles UML definidos especialmente para nuestra propuesta). Estos modelos representan los elementos del dominio del software bajo verificación y pruebas de aceptación abstractas.

Utilizando los perfiles definidos, el desarrollador puede enriquecer los modelos generados por el usuario. Los elementos que MoFQA le permite definir son:

[11] ***Eclipse Modeling Framework:*** https://www.eclipse.org/modeling/emf/

- Resultados esperados (*test oracles*) luego de la ejecución de unidades funcionales del código.
- Pre-condiciones para la ejecución de cada unidad funcional.
- Post-condiciones para la ejecución de cada unidad funcional.

Los modelos de usuario y desarrollador representan pruebas abstractas (de aceptación y unitarios, respectivamente). La transformación de pruebas abstractas a concretas se realiza utilizando la herramienta Acceleo, en conjunto con las reglas de transformación definidas por MoFQA (MoFQA *transformation rules*). Estas reglas permiten generar código en Java[12] (frameworks TestNG[13] y Selenium[14]).

El código generado puede ser utilizado para ejecutar las pruebas de aceptación y unitarias sobre el software bajo verificación, en un entorno de desarrollo integrado como Eclipse[15].

Referencias

[1] Magalí González, Luca Cernuzzi, and Oscar Pastor. A navigational role-centric model oriented web approach - MoWebA. *International Journal of Web Engineering and Technology (IJWET)*, 11(1):29–67, 2016.

[12] **Java:** https://www.java.com/es/
[13] **TestNG:** http://testng.org/doc/index.html
[14] **Selenium:** http://www.seleniumhq.org/
[15] **Eclipse:** http://www.eclipse.org/