EVALUACIÓN DE MDA Y MERODE EN EL DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN WEB

Guillermo Omar Pizarro Vásquez
Rafael Eduardo Rivadeneira Campodónico
Mónica Katiuska Villavicencio Cabezas
María Verónica Macías Cabezas
Facultad de Ingeniería en Electricidad y Computación
ESCUELA SUPERIOR POLITECNICA DEL LITORAL
Campus Gustavo Galindo, Guayaquil, Ecuador
gpizarro@fiec.espol.edu.ec
rrivaden@fiec.espol.edu.ec
mvillavi@espol.edu.ec
mmacias@fiec.espol.edu.ec

Resumen

El presente artículo trata sobre un caso de estudio en el cual utilizaron dos metodologías complementarias: MDA (Model Driven Architecture) y MERODE (Model-driven, Existence-dependency Relation Object-oriented DEvelopment). MDA como MERODE son metodologías que soportan el paradigma de la separación del modelo del dominio (modelo conceptual, modelo de empresa) de la aplicación de sus especificaciones técnicas, con el fin de generar soluciones más flexibles, y por tanto más fáciles de mantener. Dado que ya se ha demostrado en estudios anteriores los beneficios que estas metodologías aportan a los atributos de flexibilidad y de mantenibilidad de las aplicaciones; nuestro trabajo se enfocó en comprobar la incidencia en el tiempo de desarrollo que implica el uso de estas dos metodologías en conjunto. A fin de lograr nuestro objetivo, se realizó la estimación de tiempos usando COCOMO (COnstructive COst MOdel), durante el proceso se tomaron las mediciones correspondientes, y finalmente se analizó las diferencias entre los tiempos estimados y reales; obteniendo como resultado que el uso de MDA y MERODE en conjunto no afecta de manera significativa en los tiempos de desarrollo de la aplicación.

Palabras Claves: *MDA, MERODE, COCOMO, Arquitectura, Diseño, Orientado a Objetos, modelamiento del dominio.*

Abstract

This article discusses a case study in which they used two complementary methodologies: MDA (Model Driven Architecture) and MERODE (Model-driven, Existence-dependency Relation Object-Oriented Development). MDA as MERODE are methodologies that support the paradigm of the separation of domain model (conceptual model, business model) of the application of their technical specifications in order to generate flexible solutions, and therefore easier to maintain. As has already been shown in previous studies the benefits that these methodologies bring to the attributes of flexibility and maintainability of the applications, our work focused on verifying the impact on development time involved in using these two methods together. To achieve our goal, was conducted using the estimated time COCOMO (Constructive Cost Model), during the actual measurements were taken, and finally analyzed the differences between estimated and actual times, with the result that the use of MDA and MERODE together does not affect significantly the development time of the application.

KeyWords: *MDA, MERODE, COCOMO, Architecture, Design, Object Oriented, domain model*

1. Introducción

Tanto en los negocios como en las tecnologías se perciben cambios y avances, que en el campo de la Ingeniería de Software plantean el reto de crear aplicaciones que puedan adaptarse de forma simple y eficiente. Para poder implementar requerimientos emergentes se vuelve necesario utilizar alguna metodología, de tal manera que si existen cambios en el dominio del negocio, la implementación de los cambios necesarios en la aplicación no afecte de manera crítica a las actividades de la empresa; así mismo, hay que considerar las nuevas tecnologías que puedan servir para mejorar algún servicio que se haya implementado.

Según investigaciones realizadas [1], una forma de lograr mayor flexibilidad y un alto nivel de mantenibilidad en las aplicaciones de software es la *separación del "modelo del dominio" (modelo del negocio, o modelo conceptual) de las "diversas tecnologías" que pueden usarse en su implementación y que surgen a través de los tiempos*.

Dos metodologías que se apoyan en el concepto de separar el modelo del negocio de la tecnología asociada, son **MDA** y **MERODE**. **MDA** es una propuesta de la OMG (Object Management Group) para el desarrollo de software desde el diseño de modelos, proporciona una solución para los cambios de negocio y de tecnología, permitiendo construir aplicaciones independientes de la plataforma [2]; **MERODE** es una metodología de análisis OO basado en el paradigma del modelamiento del dominio [3].

Actualmente, ya se ha demostrado que trabajar con MERODE [3-4], con los recursos humanos y de herramientas que usualmente están disponibles para una típica PYME, nos proporciona una aplicación flexible; pero no había sido posible demostrar la forma en que el uso conjunto de estas metodologías (MDA y MERODE) podía influir en los tiempos de desarrollo finales de la aplicación, lo que es relevante porque muchas veces no usamos una determinada metodología porque nos resta celeridad en el desarrollo. Por ello nos propusimos realizar este caso de estudio, en el cual aplicando estas metodologías (MDA y MERODE) en el diseño y desarrollo de una aplicación Web, estimando los tiempos de desarrollo usando un método de estimación apropiado y tomando las métricas apropiadas, comprobaremos si es posible aplicar conjuntamente MERODE y MDA sin que ello afecte de manera significativa el tiempo estimado de desarrollo.

El presente artículo muestra los resultados del caso de estudio propuesto, y se encuentra organizado de la siguiente manera: en la sección 2 describiremos las arquitecturas de MDA y MERODE; seguidamente, en la sección 3 detallaremos cómo aplicamos las técnicas de MDA y MERODE en el caso de estudio y en la sección 4 definiremos las reglas de transformación utilizadas. Finalmente, en la sección 5 presentaremos los resultados obtenidos y las conclusiones.

2. Antecedentes

Debido a la importancia que tiene el desarrollo de software en el ámbito de los negocios, es necesaria la aplicación de métodos, técnicas y herramientas que nos ayuden a obtener productos de calidad y uno de los aspectos importantes a considerar es la flexibilidad, debido a que tanto en los negocios como en las tecnologías se perciben cambios y avances, que en el campo de la Ingeniería de Software plantean el reto de crear aplicaciones que se puedan adaptar de forma simple y eficiente.

Para poder implementar requerimientos emergentes se vuelve necesario utilizar alguna metodología, de tal manera que si existen cambios en el dominio del negocio, esto no afecte de manera crítica a sus actividades; así mismo, hay que considerar las nuevas tecnologías que puedan servir para mejorar algún servicio que se ha implementado.

3. Contenido

3.1. MDA, una Arquitectura Dirigida por Modelos

En el año 2000, el OMG publicó un artículo [5] en el que se presentaba una metodología donde todo se centraba en modelos. Su principal objetivo era diseñar software independiente de la plataforma, de tal manera que si el modelo de negocio exigía pasar -el software en producción- a otra tecnología, simplemente se genere el código respectivo a partir del modelo. Para algunos, esto llegó a ser una utopía [6].

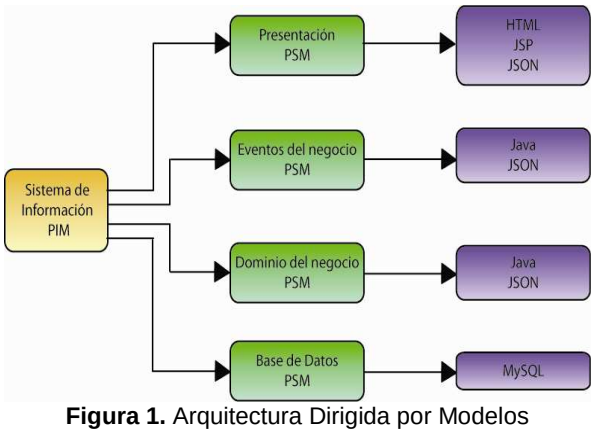

Figura 1. Arquitectura Dirigida por Modelos

MDA define modelos (Figura 1) en diversos niveles de abstracción, que van desde el modelo del dominio al modelo de especificaciones tecnológicas, cada uno con alcance definido. A continuación se listan estos niveles:

1. Modelo independiente de la plataforma (PIM).
2. Modelos específicos de la plataforma (PSM).

3. Modelos específicos de implementación (PSI).

El **PIM** es una vista que representa la especificación de un dominio, es decir, su estructura, sus restricciones (pre-condiciones y pos-condiciones), en fin, es tal la abstracción a la que se desea llegar, que podrá ser aplicada a diferentes tecnologías. El **PSM** es una vista específica de una tecnología definida. El **PSI** es el código del sistema a desarrollar, puede ser generado en parte o casi en su totalidad

3.2. MERODE, una metodología de análisis OO bajo el paradigma de modelamiento basado en el dominio

MERODE es una metodología de análisis OO, que al igual que MDA, se basa en el principio de que el modelamiento del dominio debe estar abstraído de las especificaciones tecnológicas.

En el análisis de la aplicación MERODE separa los objetos en capas: la *capa interna*, formada por los objetos propios del dominio, la *capa media*, que contiene los objetos del sistema de información; y la *capa externa*, los objetos de interfaz de usuario.

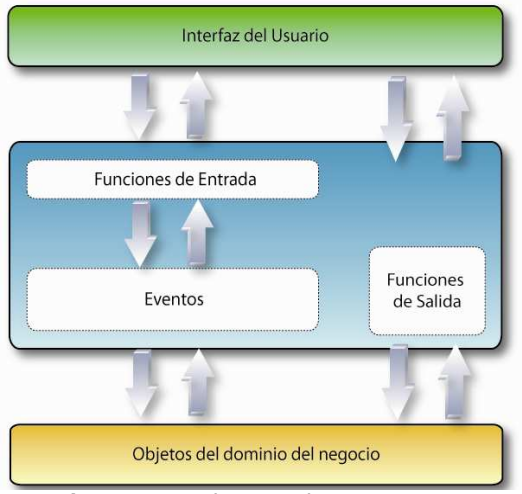

Figura 2. Arquitectura de MERODE

MERODE nos brinda una arquitectura (Figura 2) provista de las especificaciones que un negocio necesita, es decir, los requerimientos del negocio en ***una capa para los objetos del dominio***; las especificaciones funcionales que provienen del hecho de la realización de procesos que implican la manipulación de datos en otra ***capa denominada funcional*** que contiene: *eventos*, que a su vez son accedidos mediante *funciones de entrada* y que proporciona información consultando los objetos del dominio mediante una *función de salida*; por último tenemos la ***capa de interfaz del usuario***, aquella a la que accede el usuario como tal.

Por último, pero no menos importante, MERODE nos ofrece un control de calidad, que nos asegura representar un buen análisis de sistema, mediante la representación de un solo tipo de relación entre los objetos, denominada relación de dependencia-existencia; además de las técnicas necesarias para representar los aspectos estáticos y dinámicos del modelo del dominio y un conjunto de reglas que permiten la verificación automática de la consistencia interna entre las diferentes vistas del modelo [8].

3.3. COCOMO.

Es un modelo matemático de base empírica utilizado para estimación de costes de software. Incluye tres sub modelos, cada uno ofrece un nivel de detalle y aproximación cada vez mayor, a medida que avanza el proceso de desarrollo del software: básico, intermedio y detallado.

El *modelo básico* nos proporciona una aproximación rápida del esfuerzo.

El *modelo intermedio* añade al modelo básico quince modificadores opcionales para tener en cuenta en el entorno de trabajo, son los siguientes:

- de software: RELY (confiabilidad requerida), DATA (tamaño de la Base de Datos) y CPLX (complejidad del producto).
- de hardware: TIME (apremios de funcionamiento Run-time), STOR (apremios de la memoria), VIRT (volatilidad del ambiente virtual de la máquina), TURN (tiempo de respuesta).
- de personal: ACAP (calificación de los analistas), AEXP (experiencia del personal), PCAP (calificación de los programadores), VEXP (experiencia del personal en la máquina virtual), LEXP (experiencia en el lenguaje).
- de proyecto: MODP (uso de prácticas modernas de programación), TOOL (uso de herramientas de desarrollo de software), SCED (limitaciones en el cumplimiento de la planificación).

El *modelo detallado* tiene dos mejoras con respecto al anterior: son dependientes de la fase sobre las que se realizan las estimaciones; y establece tres jerarquías de niveles de producto que son: módulo, subsistema y sistema [8].

4. Aplicación de las técnicas de MDA y MERODE en el caso de estudio

El caso de estudio en el que vamos aplicar MDA y MERODE, consiste en el diseño e implementación de un Sistema de Administración de Eventos de índole científico. Es importante mencionar que anteriormente ya se ha implementado, como Proyecto de Tesis, un producto llamado AppVlir8, el mismo que es también un Portal Web de Administración de Eventos, nuestra idea es retomar el diseño de este software y modificarlo, corrigiendo defectos y agregando mejoras

que han sido identificadas durante el tiempo en que el software ha estado en producción.

Los módulos que se implementaron en el rediseño de este Sistema, fueron: Módulo de Administración Central (MAC), Módulo de Suscripción de Usuarios (MSU), Módulo de Administración de Eventos (MAE) y el Módulo de Convocatoria y Evaluación de Artículos (MCE).

A continuación, vamos a definir los tres modelos utilizados en la aplicación: el PIM, el PSM y el PSI.

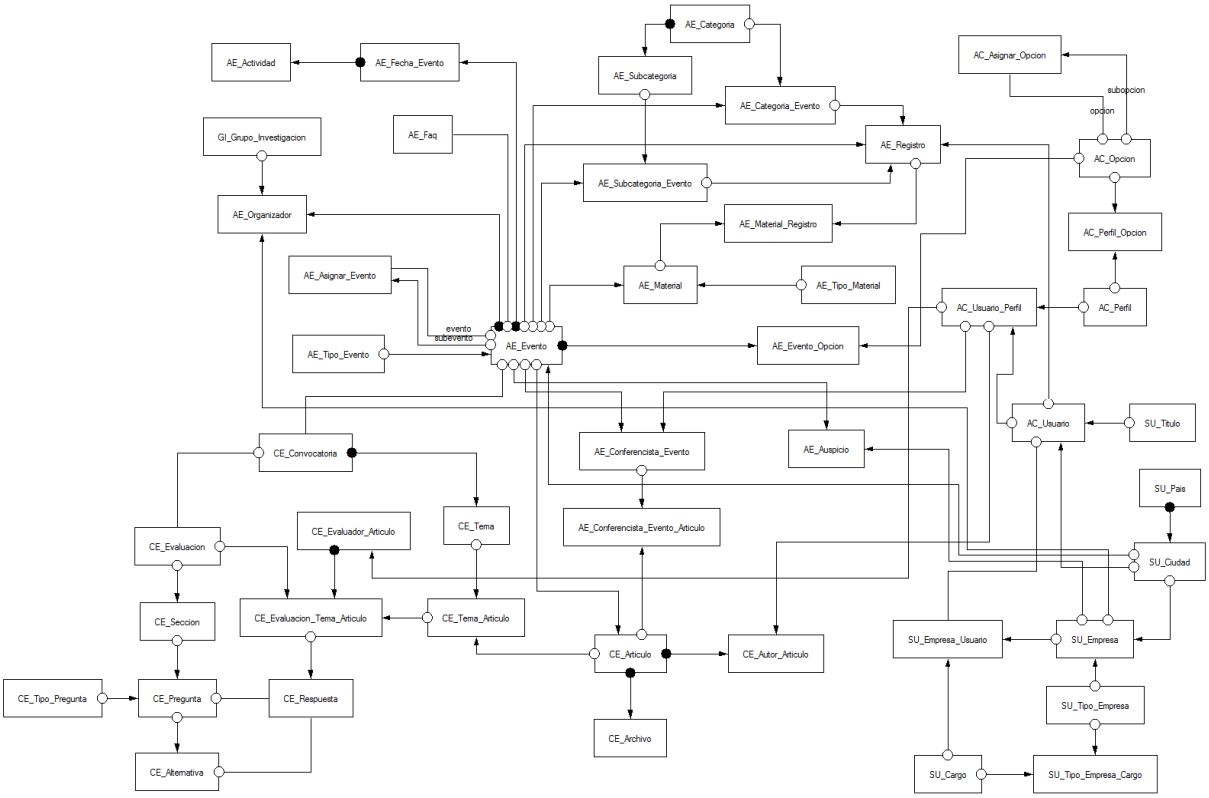

Figura 3. PIM del Sistema Web para la Administración de Eventos Científicos (WebSAE)

4.1. Modelo Independiente de la Plataforma (PIM)

La elaboración del PIM fue diseñado mediante el análisis de MERODE. En este apartado se va a explicar en breves rasgos la Aplicación Web que se desea implementar para luego mostrar el resultado de lo que se desea generar mediante MERODE.

Las principales actividades que se realizaron en esta etapa de elaboración del PIM fue la documentación de levantamiento de nuevos requerimientos, la petición de mejora de otros y la anulación de unos pocos; además del estudio de impacto para el mantenimiento respectivo ante los cambios que se iban a realizar, se presenta el EDG (Existence Dependency Graph) correspondiente luego del análisis y diseño del Sistema (Figura 3).

Otra acotación importante es el hecho de haber realizado una estimación de tiempos mediante COCOMO [9] para obtener una referencia en el momento de comparar las métricas reales obtenidas a partir de este proyecto.

4.2. Modelo Específico de la Plataforma (PSM)

Este modelo fue mapeado del PIM hecho según el análisis en MERODE a UML, respetando las correspondientes restricciones. En nuestro estudio, hemos separado el PIM en varios módulos para una mejor comprensión del dominio. La división en varios módulos es necesaria no sólo para una mejor concepción del diseño, si no para identificar las partes independientes que conforman el PIM.

A continuación, se muestra el PSM del Módulo de Administración Central (Figura 4) perteneciente a la Aplicación Web, en el cual, se pueden gestionar los diferentes perfiles que tiene un usuario en la aplicación y por ende las operaciones que ese usuario puede realizar dependiendo del perfil o perfiles que posea.

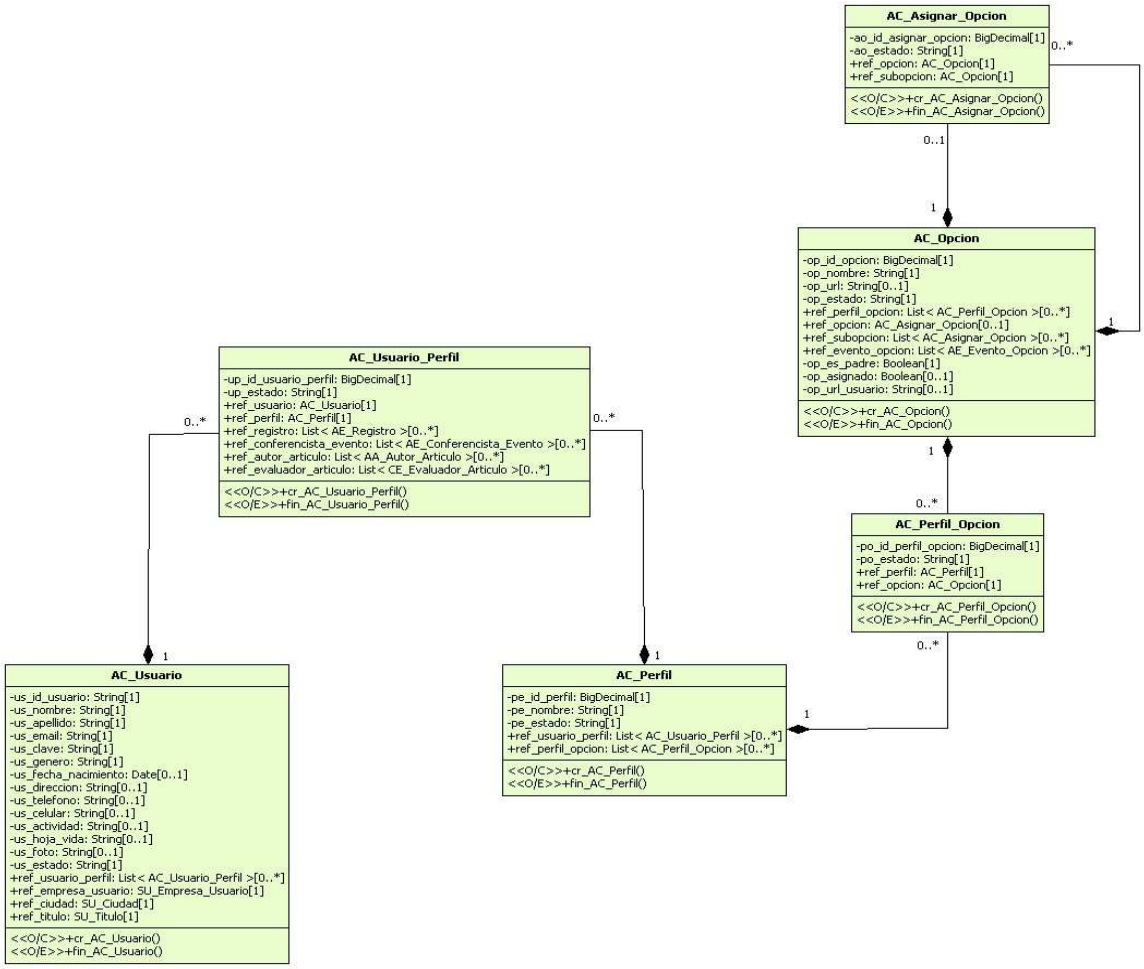

Figura 4. PSM del Módulo de Administración Central (MAC)

4.3. Modelo Específico de la Implementación (PSI)

A partir del PSM, se puede generar código a diferentes plataformas. En nuestro caso el Sistema Web de Administración de Eventos (WebSAE) hemos decidido mantenerlo en la plataforma J2EE –como en su implementación original- siguiendo la especificación JSP 2.0.

Para la diagramación en UML y la generación del código se ha utilizado la herramienta CASE Open Source llamada StarUML.

En primera instancia para la creación del PSM usamos la herramienta llamada Magic Draw, con ella generamos las clases Java de la capa del dominio y el correspondiente código SQL para la creación de la Base de Datos. Debido a que uno de los objetivos de nuestro Proyecto de Grado, era utilizar herramientas que estuviesen fácilmente disponibles para las PYMES desarrolladoras de software, escogimos la herramienta de código abierto StarUML, pero con ella sólo pudimos generar las clases Java del dominio.

5. Reglas de Transformación

5.1. Del PIM al PSM

La primera regla es transformar el EDG en un diagrama de clases. Esta tarea la realizamos de forma manual debido a que no encontramos cómo automatizar todo el proceso de conversión. Cabe mencionar, que el EDG fue hecho mediante la herramienta MERMAID y los diagramas de clases los elaboramos con StarUML manualmente.

Con respecto a la transformación del EDG, la relación de ***existencia-dependencia*** considerada en el análisis de MERODE, al pasarla a UML, ésta se convierte en una ***relación de composición***.

5.2. Del PSM al PSI

5.2.1. Capa del Dominio

Del Diagrama de Clases que se elaboró en el paso anterior se generó el código de esta capa (en Java). Así mismo se generó el correspondiente DDL para la creación de la Base de Datos (en nuestro caso

MySQL) (Figura 5). Este fue el único paso que se realizó de manera automática.

Figura 5. Regla de transformación de la capa del dominio

5.2.2. Capa de Control de Eventos

Antes de mencionar cómo se transformaron los eventos (en los que participan los objetos del dominio) a código, es importante anotar que en MERODE existen dos reglas para realizarlo: 1) implementar una sola clase por cada evento; y 2) en una sola clase implementar todos los eventos, para salvaguardar la atomicidad de los mismos.

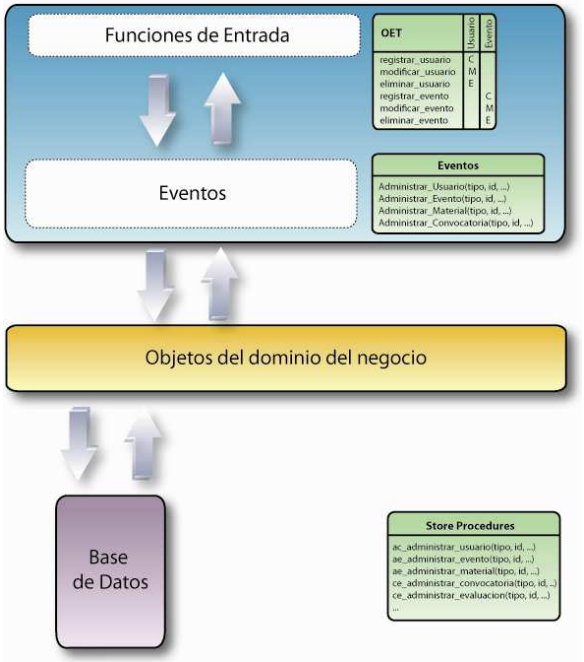

Figura 6. Regla de Transformación de la Capa de Control de Eventos

Nosotros utilizamos un *híbrido*, es decir, por cada objeto se creó una clase-evento en la que se realizaban los eventos de crear, modificar y eliminar. Por ejemplo: para el objeto *AC_Usuario* se creó la clase *Administrar_Usuario* y dentro de esta clase-evento diferenciamos el tipo de evento a realizar. Con esto nos ahorramos una gran cantidad de clases debido fundamentalmente a la reutilización de código.

Así mismo, por cada tipo de evento híbrido, se creó un store procedure. Siguiendo el ejemplo anterior, ante la clase-evento *Administrar_Usuario* se creó el store procedure *administrar_usuario*. (Figura 6)

5.2.3. Capa de Control de Eventos

En esta capa también seguimos reglas para la correspondiente transformación. Para la mayoría de los objetos del dominio creamos un directorio del lado del cliente y en cada directorio un sub directorio con el evento a implementar. Por ejemplo: con el objeto *AC_Usuario* del lado del servidor, se creó el directorio *usuario* del lado del cliente de la aplicación, y dentro de ese directorio, los sub directorios: *crear*, *modificar* y *eliminar*, cada directorio puede ser accedido dependiendo del perfil asignado a un usuario.

Para la implementación de seguridades a ciertos directorios de la Aplicación Web, se utilizaron los filtros que nos proporciona la arquitectura J2EE.

6. Resultados

Por cada módulo que fue diseñado e implementado, realizamos una estimación inicial de tiempo en meses, la cual se muestra en el Gráfico 1, para el cálculo de los factores considerados en la estimación del tiempo mediante COCOMO se definen a continuación.

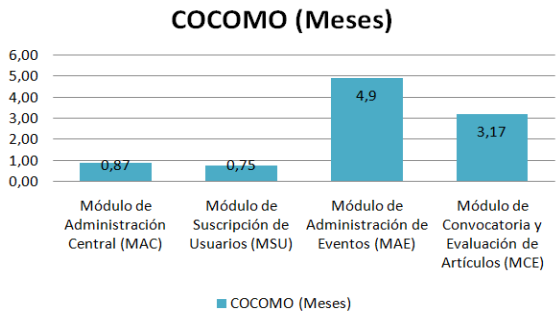

Gráfico 1. Estimación Realizada en COCOMO.

Con respecto al producto: RELY (Bajo), DATA (Nominal), CPLX (Bajo).

Con respecto al personal: ACAP (Muy Alto), AEXP (Muy Alto), PCAP (Alto), VEXP (Alto), LEXP (Muy Alto).

Con respecto al proyecto: MODP (Muy Alto), TOOL (Muy Alto), SCED (Muy Alto).

Durante el desarrollo de cada módulo se levantaron métricas, las mismas que fueron procesadas y cuyos resultados se muestran en el Gráfico 2.

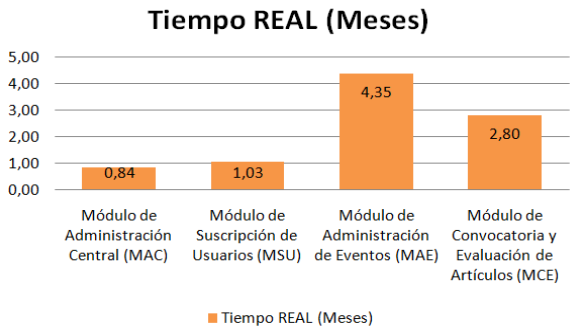

Gráfico 2. Tiempos Reales de cada Módulo.

En el gráfico 3 se muestra el porcentaje de error entre el tiempo estimado y el tiempo real; por ende, se puede verificar que en MAC el porcentaje es -3,11% no hubo problemas debido a que era solo código JAVA y se conocían los requerimientos, además ya se encontraba desarrollada la Base de Datos (MySQL) y el código java de la capa del dominio que fueron generados de manera automática, esto contribuyó al rápido desarrollo de este módulo.

Para MSU hubo un retraso, esto fue en parte al tiempo de aprendizaje de algunas librerías. Sin embargo, el tiempo de aprendizaje invertido en este módulo, nos sirvió para que en los demás módulos (MAE y MCE), se muestre un progreso en la disminución de tiempos.

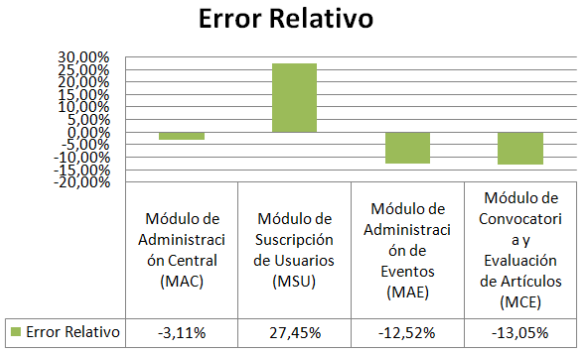

Gráfico 3. Porcentajes de error con respecto a lo estimado con el tiempo real.

7. Conclusiones

Dado los resultados, se ha demostrado que usando COCOMO, estos tiempos estimados no se ven afectados de mayor forma por el uso de MERODE y MDA en el análisis y desarrollo del Sistema; y que dadas las ventajas de flexibilidad que ha demostrado el uso de éstas metodologías en el producto final, se comprueba que es factible y recomendable el uso de las mismas.

Algunas ideas surgieron en la implementación de este sistema para encontrar la forma de automatizar pasos que se realizaron manualmente, sobre estas ideas se exponen a continuación:

En la transformación del PIM al PSM, se generó de manera manual el UML correspondiente del EDG. Se ha estudiado la posibilidad de generar el UML (dependiendo de la herramienta CASE: StarUML, MagicDraw, etc.) a partir del archivo XML que nos proporciona MERMAID, para la automatización de esta transformación.

Con respecto a la capa de control de eventos, también seguimos un estándar que puede ser generado a partir de las clases del dominio. Esto sería de gran ayuda, debido a que se ahorraría una gran cantidad de código repetitivo.

8. Referencias

[1] Verónica Macías. “Modelamiento basado en el dominio: Estado del Arte”, I Jornada de Ingeniería de Software, Guayaquil-Ecuador, Noviembre 2004.

[2] María Victoria Di Libero, “Arquitectura Dirigida por Modelos”, Buenos Aires-Argentina.

[3] Salomón Herrera, Verónica Macías. “Correspondencia entre la metodología de Análisis MERODE (Object Oriented Bussiness Controller) y el Esquema de Diseño MVC (Model View Controller) de la Arquitectura J2EE (Java 2 Enterprise Edition) para el Desarrollo de una Aplicación Web”, ANDESCON, Cusco – Perú. Octubre-2008.

[4] Karina Chong, Verónica Macías, Monique Snoeck. “Experiences with the use of MERODE in the development of a Web Application”. ESPOL – VLIR, componente 8 Ingeniería de Software, Guayaquil-Ecuador. Enero 2008.

[5] Richard Soley and the OMG Staff Strategy Group. “Model Driven Architecture”, Object Managment Group. November-2000.

[6] Dave Thomas. “MDA: Revenge of the Modelers or UML Utopia?”, IEEE Software, ROI in the Software Industry, May/June 2004, Vol. 21, No. 3. pp. 15-17.

[7] M. Snoeck, G. Dedene, M. Verhelst, A. Depuydt, “Object-Oriented Enterprise Modelling with MERODE”, Leuven University Press, 1999 pp. 12-16.

[8] Snoeck M., Michels C., and Dedene G., “Consistency by construction: the case of MERODE”, Conceptual Modeling for Novel Application Domains, ER 2003 Workshops ECOMO, ICWMQ, AOIS and XSDM Proceedings, October 2003, p. 105-117.

[9] Barry W. Boehm, Chris Abts, A. Winsor Brown, Sunita Chulani, Bradford K. Clark, Ellis Horowitz, Ray Madachy, Donald Reifer, Bert Steece. “Software Cost Estimation with COCOMO II”, Prentice Hall, 2000.