EXPERIENCIA DE DESARROLLO DE UNA APLICACIÓN WEB UTILIZANDO LA METODOLOGÍA UWE Y EL LENGUAJE QVT EN LA TRANSFORMACIÓN DE MODELOS

A. Narváez[1], P. Baldeón[2], C. Hinojosa[3], D. Martínez[4]

1 Escuela Politécnica del Ejército, Ecuador, alexspaola@hotmail.com

2 Escuela Politécnica del Ejército, Ecuador, paulslk10@msn.com

3 Escuela Politécnica del Ejército, Ecuador, chinojosa@espe.edu.ec

4 Escuela Politécnica del Ejército, Ecuador, dmartinez@espe.edu.ec

RESUMEN

En este artículo se presenta la experiencia del desarrollo de un sistema, utilizando la metodología UML-BASED WEB ENGINEERING, con una orientación a la transformación de modelos. Se presentan las características principales y los beneficios de esta propuesta de desarrollo de software web. El principal problema que han acarreado los sistemas web es que se les ha dado un enfoque a lo largo de los años donde lo importante era generar código sin preocuparse por el proceso. La metodología que se explicará a lo largo del presente artículo es UWE (UML-BASED WEB ENGINEERING), esta metodología propone la transformación de modelos utilizando QVT como lenguaje y MDA como arquitectura. MDA propone cuatro niveles, para el desarrollo de software, estos son: CIM, PIM, PSM y el de código. Cada nivel propone diferentes modelos, estos modelos parten de un concepto general hasta llegar a un concepto particular. Con la implementación de la transformación de modelos se busca tener sistemas desarrollados prácticamente semiautomáticos, es decir, que se utilice una herramienta CASE. La selección de la herramienta va a depender si tiene soporte para los diagramas que utilice la metodología y sobre todo que tenga soporte para el lenguaje de transformación.

Palabras Clave: UWE, metamodelo, transformación de modelo, QVT

ABSTRACT

This article presents the experience of developing a system using the methodology UML-BASED WEB ENGINEERING, oriented to the transformation of models. We present the main features and benefits of the development of web systems. The main problem that web systems have brought is that they have been given a focus throughout the years where it was important to generate code without worrying about the process. The methodology that will be explained throughout this article is UWE (UML-BASED WEB ENGINEERING), this methodology proposes the transformation of models using MDA and QVT as language and architecture. MDA offers four levels to the development of software, these are: CIM, PIM, PSM and code. Each level offers different models, these models are based on a general concept down to a particular concept. With the implementation of the transformation of models have developed systems for virtually semi-automatic, ie you use a CASE tool. The choice of tool will depend on whether it has support for the diagrams that use the methodology and above all have support for language processing.

KeyWords: UWE, metamodel, model transformation, QVT

1. INTRODUCCIÓN

El nacimiento de la World Wide Web se dio de una forma poco organizada y pobre en diseño. La evolución que ha experimentado la Web tiene como base la necesidad de trabajar sobre algo formal, es decir, organizarla y normalizarla.

Para mejorar los sistemas web los desarrolladores se han enfocado más en el proceso que en las herramientas de desarrollo, (genera un problema) --- la solución es…. para el cual se han propuesto metodologías web, que permitan aplicar estándares, técnicas y enfoques que han hecho que estos sistemas vayan tomando más fuerza. Una metodología web que nació para mejorar y cubrir estas falencias es UWE, la cual se adhirió a diferentes estándares probados y que tenían aceptación por parte de los desarrolladores. Entre los estándares están: UML, XMI, MOF, MDA, QVT y XML.

El objetivo de este artículo es mostrar la experiencia que se tuvo al desarrollar un sistema web utilizando la metodología UWE y como se realizaron las transformaciones de modelos en los diferentes niveles que propone la arquitectura MDA, utilizando el lenguaje QVT. La herramienta CASE selecciona fue MagicDraw porque soporta el modelamiento de diagramas propios de la metodología UWE. Se utilizó esta herramienta porque permite reducir los tiempos de desarrollo de sistemas orientados a objetos gracias a que utiliza UML. Así también, el levantamiento de requisitos se fortaleció con el uso de la Norma IEEE 830 (ESPECIFICACIONES DE LOS REQUISITOS DEL SOFTWARE) porque la metodología UWE sólo propone el uso de casos de uso. El propósito del uso de esta norma es el de garantizar la calidad de la documentación en cuanto al levantamiento de requerimientos.

Las transformaciones de modelos que se explicaran a lo largo del presente artículo corresponden a los modelos pertenecientes al nivel PIM (Platform Independet Model). Los modelos utilizados para la transformación-fueron: Modelo Conceptual, Modelo de Navegación, Modelo de Presentación y el Modelo de Procesos.

Este artículo está estructurado de la siguiente manera: la sección 2 explica la metodología UWE, cual es el metamodelo de UWE y el metamodelo de QVT. La sección 3 muestra los diferentes niveles que propone la arquitectura MDA y cuáles son los diagramas que se transformarán. En la sección 4 se explica las diferentes transformaciones que propone la metodología UWE. En la sección 5 se muestran los resultados que se obtienen cuando se realiza la transformación de modelos. En la sección 6 se hace referencia a algunos trabajos relacionados a la transformación de modelos y a la utilización de QVT. En la sección 7 se muestra la conclusión a la que se llegó tras realizar el proyecto de desarrollo de software y finalmente en la sección 8 se muestran las referencias bibliográficas.

2. METODOLOGÍA UWE (UML-BASED WEB ENGINEERING)

2.1 Introducción

UWE nació a finales de la década de los 90 con la idea de encontrar una forma estándar para analizar y diseñar modelos de sistemas web. El objetivo por el cual nació esta metodología fue utilizar un lenguaje común o por lo menos definir un metamodelo basado en el mapeo a lo largo de las diferentes etapas. En esa época UML prometía convertirse en un estándar para el modelamiento de sistemas. Por este motivo, UWE se adhirió a UML y no a otra técnica de modelado. UWE se ha adaptado a las nuevas características de los sistemas web como transacciones, personalizaciones y aplicaciones asíncronas, y por otro lado ha evolucionado para incorporar técnicas de ingeniería de software como el modelamiento orientado a aspectos y nuevos lenguajes de transformación para mejorar la calidad del diseño. [1] (Rossi, Schwabe, & D. Olsina, 2008)

2.2 Definición

UWE es una metodología basada en el Proceso Unificado y UML para el desarrollo de aplicaciones Web, cubre todo el ciclo de vida de las aplicaciones Web. Su proceso de desarrollo se basa en tres frases principales: la fase de captura de requisitos, la fase de análisis y diseño y la fase de la implementación.

2.3 Características

La metodología UWE define vistas especiales representadas gráficamente por diagramas en UML, tales como el modelo de navegación y el modelo de presentación.

Los diagramas se pueden adaptar como mecanismos de extensión basados en estereotipos que proporciona UML. Estos mecanismos de extensión son los que UWE utiliza para definir estereotipos que son los que finalmente se utilizarán en las vistas especiales para el modelado de aplicaciones Web. De esta manera, se obtiene una notación UML adecuada para un dominio específico a la que se conoce como "Perfil UML" [1] (Rossi, Schwabe, & D. Olsina, 2008).

Un perfil de UML consiste en una jerarquía de estereotipos y un conjunto de restricciones. Los estereotipos son utilizados para representar instancias de las clases. La ventaja de utilizar los perfiles de UML es que casi todas las herramientas CASE de UML los reconocen. Los modelos deben ser fácilmente adaptables al cambio en cualquier etapa del desarrollo.

2.4 Meta Modelo de UWE

El meta modelo de UWE es una extensión del meta modelo de UML 2.0, esto quiere decir que UWE utiliza todos los elementos de los modelos sin ninguna modificación. Los elementos de los modelos que añade UWE están relacionados por herencia con al menos un elemento del modelo de UML.

Para crear modelos de UWE para aplicaciones Web se puede utilizar cualquier herramienta CASE que soporte perfiles o extensiones UML.

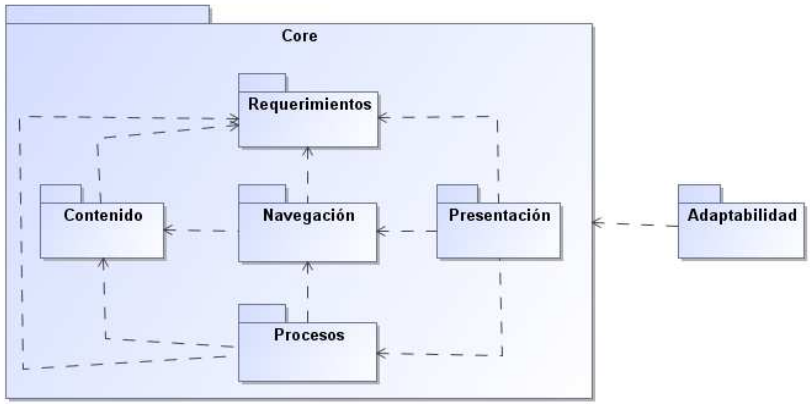


Figura 1: Meta Modelo (Rossi, Schwabe, & D. Olsina, 2008)

- El paquete de Requerimientos comprende la extensión de UWE para los Casos de Uso
- Los paquetes de Navegación y Presentación corresponden a los modelos iguales de UML.
- Los paquetes de Contenido y Procesos actualmente se los utiliza como una parte para mostrar que UWE permite al diseñador desarrollar modelos de contenido y proceso usando todas las características de UML.
- El paquete de Adaptabilidad representa la característica más importante de las aplicaciones Web.

2.5 Desarrollo dirigido por modelos

MDA es un marco de trabajo que utiliza modelos en el proceso de desarrollo de software. MDA propone la definición y uso de modelos a diferente nivel de abstracción, así como la posibilidad de la generación automática de código a partir de los modelos definidos y de las reglas de transformación entre dichos modelos [2] (Ciberaula, 2010). El mayor beneficio que propone la separación del sistema es que las definiciones se pueden reutilizar para generar la implementación de nuevos sistemas en diferentes tecnologías.

2.5.1 Funcionamiento de MDA

El objetivo de OMG era estandarizar el conjunto de servicios para el desarrollo de sistemas, es decir implementar un framework para sistemas distribuidos. Por este motivo el segundo framework adoptado en el año 2001 fue MDA. MDA es un enfoque para la utilización de modelos en el desarrollo de software.

La idea principal de la Arquitectura Dirigida por Modelos es separar la especificación de las funcionalidades de las especificaciones de la implementación de los sistemas para una determinada plataforma. Una frase pro-

pia de esta arquitectura es: “Diseñar una vez, construir en cualquier plataforma”. [3] (OMG, 1997-2010).

1. CIM (Computation Independent Business Model)

Los requerimientos del sistema son modelados en la fase CIM, donde se describe las características que va a tener el sistema. Este modelo también es conocido como modelo de dominio o modelo de negocio. La automatización del uso del sistema no se incluye en esta fase. Este modelo muestra lo que se espera realice el sistema, además es la base para el desarrollo de los modelos PIM y PSM.

2. PIM (Platform Independent Model)

Describe el funcionamiento del sistema sin mostrar detalles de la plataforma en la que va a trabajar. Este modelo va a estar listo para cualquier tipo de arquitectura.

3. Modelo de Plataforma

En este modelo el arquitecto escogerá una o varias plataformas que permita la implementación del sistema con las características deseadas.

4. Mapeo

Proporciona especificaciones para la transformación de PIM a PSM. El modelo de plataforma va a determinar la naturaleza del mapeo.

5. Transformación

Consiste en convertir el modelo de marcas PIM al modelo PSM, este proceso se puede realizar de forma manual o automática.

6. Registro de Transformación

Este registro muestra un mapa desde el elemento del PIM hacia su correspondiente elemento del PSM y muestra las partes usadas en el mapeo para cada transformación.

7. PSM (Platform Specific Model)

Este modelo va a servir para la implementación del sistema si proporciona toda la información necesaria y detallada o puede actuar como un modelo PIM si nuevamente se va a redefinir el modelo.

El PSM va a proporcionar diferente información como: código fuente, interconexión del programa, especificaciones de carga, descriptores de despliegue y otras configuraciones específicas.

8. Enfoques de transformación

Se pueden realizar transformaciones de marcas, de meta modelos, de modelos, de diseño de aplicación y de modelo de combinación

2.5.2 Transformaciones usando QVT

QVT es una parte importante para el enfoque dirigido por modelos porque entrega medios para manipular los modelos. Proporciona un lenguaje de transformación para que el usuario pueda aplicarlo para definir y ejecutar un conjunto de transformaciones y de esta manera poder lograr el paradigma de transformación de PIM a PSM [4] (Ignjatovic).

QVT posee una naturaleza híbrida, es decir una parte declarativa y una parte imperativa

- Parte declarativa

  - Lenguaje de Relaciones: proporciona un lenguaje de transformación de usuario final capaz de manejar tareas de transformación complejas.

  - Lenguaje de Núcleo: se expresa en términos de los estándares EMOF/OCL, es de bajo nivel pero todavía comprensible.

- Parte imperativa

  - Lenguaje de Mapeo Operativo: permite definir transformaciones usando un enfoque imperativo o permite complementar las relaciones de transformación con operaciones imperativas implementando las relaciones.

  - Caja Negra (MOF): se puede permite derivar operaciones MOF a partir del Lenguaje de Relaciones para poder conectar cualquier implementación MOF.

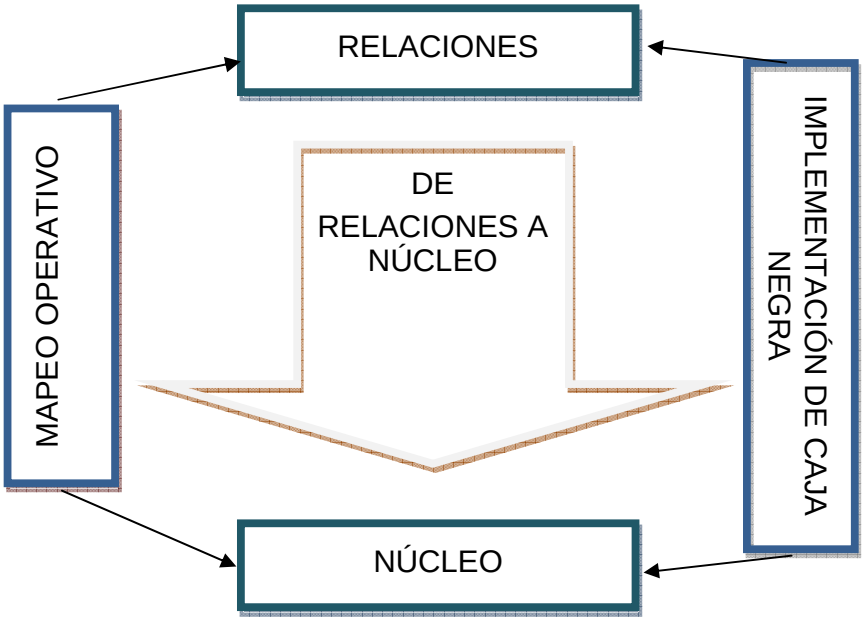


Figura 2: Las relaciones entre metamodelos QVT [5] (Object Management Group, 1997 - 2011)

3. TRANSFORMACIÓN DE MODELOS

En esta sección se describe el método utilizado para realizar la transformación de modelos basándose en el patrón de MDA. La Figura 3 muestra el patrón dirigido por modelos y como se realiza la transformación entre niveles.

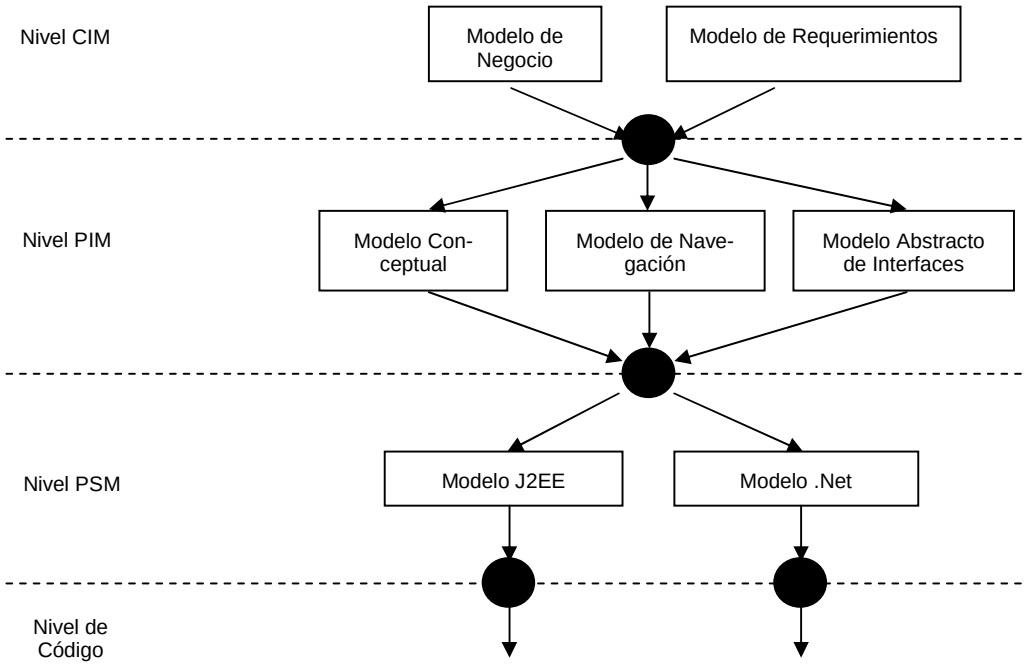


Figura 3: Patrón de Modelos [6] (Departamento de Lenguajes y Sistemas Informáticos)

En el nivel CIM se elaboran el modelo de negocios y el modelo de requerimientos los mismos que serán transformados para pasar al nivel PIM, donde se obtendrá el modelo conceptual, el modelo de navegación y el modelo abstracto de interfaces una vez que se tienen estos modelos mediante la transformación se pasará al nivel PSM, en este nivel se obtendrá el modelo específico de acuerdo a la plataforma en la que se desarrolle el sistema, y finalmente se pasa a la generación de código.

Para esta investigación se utilizó la herramienta CASE MagicDraw. Esta herramienta tiene la ventaja de que maneja todos los elementos de UML y para realizar los diagramas de la metodología UWE utiliza el complemento MagicUWE. Este complemento permite adicionar a la herramienta el menú donde se encuentran los cinco tipos de diagramas de la metodología y también la opción para realizar las transformaciones de los diagramas.

La versión de esta herramienta que se utilizó para desarrollar este proyecto no maneja las transformaciones entre niveles, pero si maneja las transformaciones entre diagramas. Las transformaciones que realiza esta herramienta son:

- Diagrama de Contenido a Diagrama de Navegación
- Diagrama de Navegación a Diagrama de Presentación
- Diagrama de Navegación a Diagrama de Estructura de Procesos
- Diagrama de Navegación a Diagrama de Flujo de Procesos

Las transformaciones entre niveles y entre diagramas se dan gracias a QVT, que es el lenguaje que define el proceso de mapeo de las clases de un nivel hacia otro nivel. El proceso de mapeo se lo puede realizar gracias a las etiquetas o marcas que proporciona QVT. Estas marcas representan un concepto en el nivel PSM el mismo que es un elemento en el nivel PIM, lo que permite realizar la transformación de estos modelos.

La figura 4 muestra la representación gráfica de una clase y como se transforma en código según el lenguaje seleccionado.

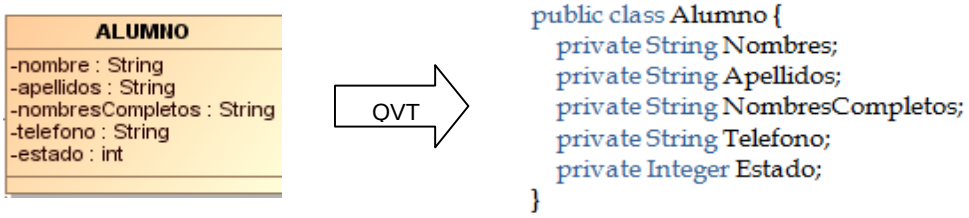


Figura 4: Ejemplo de la transformación

4. DISEÑO E IMPLEMENTACIÓN

Las fases que utiliza la metodología UWE son: Análisis, Diseño e Implementación. La metodología no define una fase clara para realizar el levantamiento de requerimientos, por lo tanto en el desarrollo de este proyecto se decidió utilizar un estándar como la IEEE 830 para definir los requerimientos funcionales y no funcionales. Gracias a este estándar se puede tener una fase de análisis más completa porque esto se une al modelo de casos de uso. En la fase de diseño se realizaron los diferentes modelos que soporta la metodología como:

- Modelo Conceptual
- Modelo de Navegación
- Modelo de Presentación
- Modelo de Procesos

La principal característica de la metodología UWE es el modelamiento y la transformación de modelos, donde se busca desarrollar aplicaciones Web de una manera sistemática y semiautomática. Las transformaciones entre modelos que se realizaron son:

- **Transformación de Diagrama de Contenido a Diagrama de Navegación**

Primero se debe realizar el diagrama de contenido, el mismo que es un diagrama UML de clases. Con ayuda de la herramienta CASE se selecciona el menú MagicUWE, luego Transformations y finalmente la opción Conten -> Navigation. En la Figura 5 se muestra una clase con su interface y en la Figura 6 se muestra el diagrama de

Navegación después de realizar la transformación.

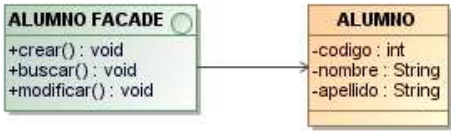


Figura 5: Diagrama de Clases

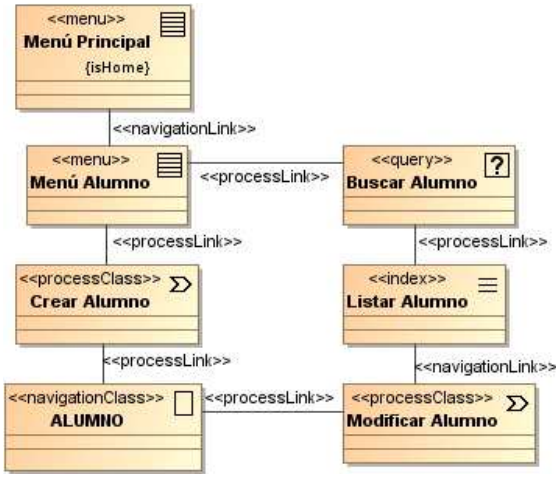


Figura 6: Diagrama de Navegación

En el diagrama de Navegación se incluyen todas las clases como clase de navegación, y se debe aumentar las clases de procesos que representan los métodos.

- **Transformación de Diagrama de Navegación a Diagrama de Presentación**

En esta transformación se utiliza el diagrama de navegación que se obtuvo anteriormente. Nuevamente se selecciona el menú MagicUWE, luego Transformations y finalmente la opción Navigation -> Presentation. En la Figura 7 se muestra el diagrama de Presentación.

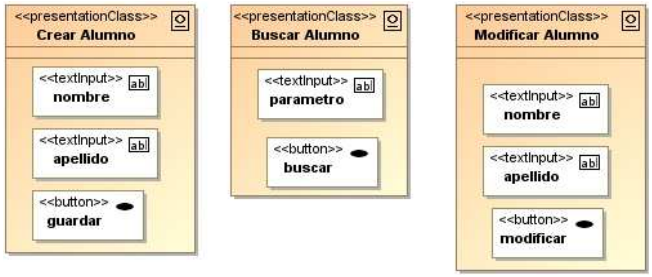


Figura 7: Diagrama de Presentación

El diagrama de Presentación muestra todos los elementos que se incluyen en las diferentes páginas web.

- **Transformación de Diagrama de Navegación a Diagrama de Estructura de Procesos**

De igual manera se debe utilizar el diagrama de navegación para transformarlo a un diagrama de Estructura de Procesos, para esto se debe seleccionar el menú MagicUWE, luego Transformations y finalmente la opción Navigation -> Process Structure. En la Figura 8 se muestra el diagrama de Estructura de Procesos.

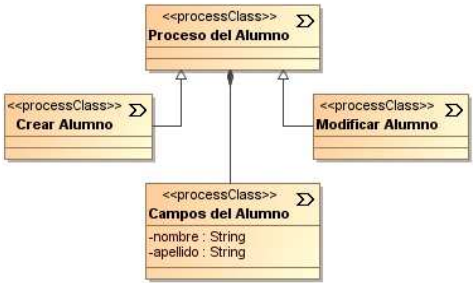


Figura 8: Diagrama de Estructura de Procesos

El diagrama de Estructura del Proceso describe las relaciones entre las diferentes clases de proceso.

- **Transformación de Diagrama de Navegación a Diagrama de Flujo de Procesos**

El diagrama de flujo de procesos también se obtiene partiendo del diagrama de navegación, para esto se debe seleccionar el menú MagicUWE, luego Transformations y finalmente la opción Navigation -> Process Flows. En la Figura 9 se muestra el diagrama de Flujo de Procesos.

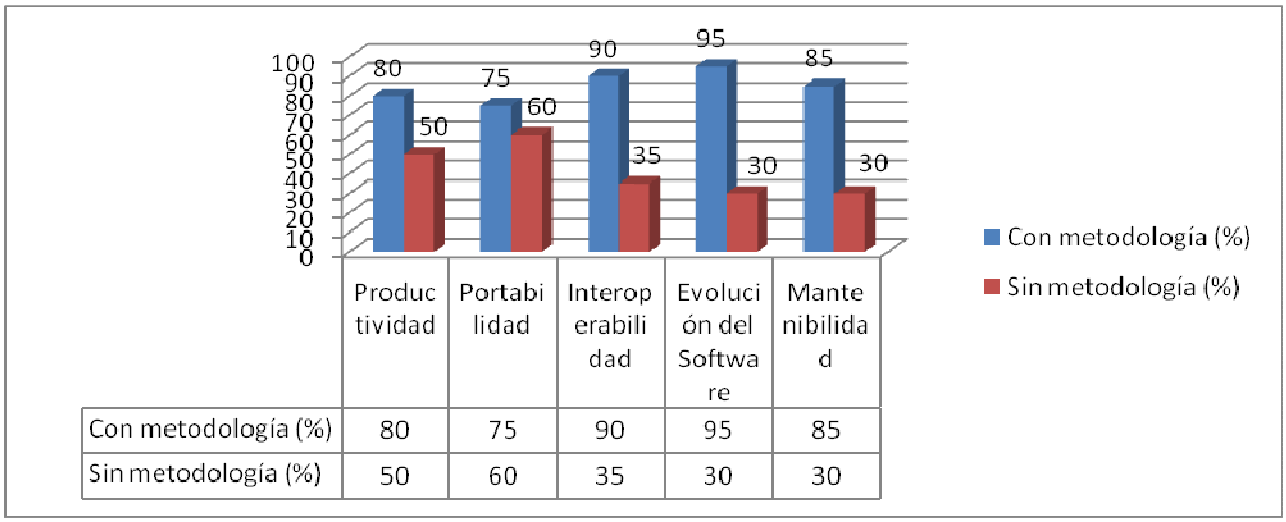
Figura 9: Diagrama de Flujo de Procesos

5. RESULTADOS

Los resultados que se obtuvieron permiten determinar que el desarrollo de sistemas web cada vez tiene más aceptación porque se pueden tener sistemas desarrollados más robustos y sobre todo con una coherencia semántica y sintáctica en todas las fases de desarrollo. Según la investigación realizada se pudo determinar que las diferentes metodologías presentan nuevas y mejores propuestas para enfrentar de mejor manera la complejidad y demanda de requerimientos por parte de los usuarios. Es por esto que las metodologías orientadas a la web están siendo más minuciosas en su evolución con el fin de cubrir todos los ámbitos del desarrollo. Un claro ejemplo de esto es la metodología UWE, porque desde su nacimiento tuvo la idea de cubrir todas las fases de desarrollo, por lo que optó por adherirse a estándares probados y aceptados por los desarrolladores como: UML, XMI, MOF, MDA, QVT y XML. Como se puede apreciar este artículo muestra el caso práctico del desarrollo de un sistema web utilizando UWE y QVT para realizar las trasformaciones de modelos. Para compensar la debilidad de la metodología en cuanto a la fase de análisis, se utilizó el estándar IEEE 830 que muestra las directrices para realizar el levantamiento de requerimientos. Con la transformación de modelos se obtienen diagramas con una misma semántica en los diferentes niveles que presenta la arquitectura MDA, es decir, que la estructura de nombres y tipos de datos se mantienen. Cuantitativamente los resultados que se obtienen cuando se realizó la transformación de modelos son:

- *Productividad*: se pueden tener sistemas web más robustos y con una mejor estructura semántica. Esto se puede evidenciar gracias a que cada vez los sistemas web tienen mayores y mejores características que cubren las necesidades actuales de los usuarios.
- *Portabilidad*: esta característica permite ejecutar un sistema en diferentes plataformas, por esta razón los desarrolladores no tienen que preocuparse si su sistema funcionará en una u otra plataforma. Dentro de la portabilidad también se habla de la adaptabilidad, la capacidad de instalación y la coexistencia.
- *Interoperabilidad*: se refiere a la capacidad del producto software para interactuar con uno o más sistemas especificados.
- *Evolución del software*: cada vez las metodologías se preocupan de satisfacer las necesidades que tiene el usuario, es decir, busca estar a la par de la evolución de los requisitos.
- *Mantenibilidad*: es la capacidad del producto software para permitir que una determinada modificación sea implementada

En la figura se muestra los resultados de un sistema que implementa la metodología UWE y otro que no utiliza la metodología alguna para desarrollar el sistema.



	Productividad	Portabilidad	Interoperabilidad	Evolución del Software	Mantenibilidad
Con metodología (%)	80	75	90	95	85
Sin metodología (%)	50	60	35	30	30

Figura 10 Cuadro de Resultados

6. TRABAJOS RELACIONADOS

Diversas son las herramientas que permiten realizar transformaciones de modelos. A continuación se detalla las herramientas más extendidas.

- **ATL**

ATL forma parte del framework de gestión de modelos AMMA que se encuentra integrado en Eclipse y EMF. Utiliza un lenguaje propietario llamado ATLAS para definir transformaciones que se ejecuta sobre JAVA y que proporciona un entorno de depuración.

ATL posee un algoritmo de ejecución preciso y determinista, sin embargo no se apoya sobre ningún método formal. No proporciona mecanismos para la validación de las transformaciones. Para llevar a cabo transformaciones compuestas se necesita ejecutar cada una de las transformaciones participantes una a una.

Aunque la sintaxis de ATL es muy similar a la de QVT, ésta no es interoperable con QVT.

- **VIATRA**

Viatra actualmente forma parte del framework VIATRA2 , que ha sido escrito en Java y que se encuentra integrada en Eclipse. Viatra provee un lenguaje textual para describir modelos y metamodelos, y transformaciones llamados VTML y VTCL respectivamente.

La naturaleza del lenguaje es declarativa y está basada en técnicas de descripción de patrones, sin embargo es posible utilizar secciones de código imperativo. Se apoya en métodos formales como la transformación de grafos (GT) y la máquina de estados abstractos (ASM) para ser capaz de manipular modelos y realizar tareas de verificación, validación, seguridad, así como una temprana evaluación de características no funcionales como fiabilidad, disponibilidad y productividad del sistema bajo diseño.

Como puntos débiles podemos resaltar que Viatra no se basa en los estándares MOF y QVT.

No obstante, pretende soportarlos en un futuro mediante mecanismos de importación y exportación integrados en el framework.

- **EPSILON**

Epsilon es una plataforma desarrollada como un conjunto de plug-ins sobre Eclipse. Presenta el lenguaje metamodelo independiente Epsilon Object Language que se basa en OCL. Puede ser utilizado como lenguaje de gestión de modelos o como infraestructura a extender con nuevos lenguajes específicos de dominio. Tres son los lenguajes definidos en la actualidad: Epsilon Comparison Language (ECL), Epsilon Merging Language (EML), Epsilon Transformation Language (ETL), para comparación, composición y transformación de modelos respectivamente.

7. CONCLUSIONES Y TRABAJO FUTURO

Este artículo muestra como se puede desarrollar un sistema web implementando la metodología UWE. De esta metodología se trata de exponer los beneficios de realizar la transformación de modelos utilizando QVT como lenguaje de transformación. Para implementar la transformación de modelos es necesario tener claros los diferentes niveles y sus características principales. Dentro de cada nivel se elaboran los modelos que van a ser transformados hasta llegar al código fuente del sistema. Gracias a la transformación de modelos se obtiene un sistema desarrollado de una manera sistemática y semiautomática, esto gracias a que se puede utilizar una herramienta CASE que brinde soporte para este tipo de transformaciones. Implementando la transformación de modelos se puede tener sistemas web más robustos ya que el desarrollo no sólo se queda en la generación de código y páginas sino que se tiene una buena documentación y diferentes modelos que se pueden implementar para cualquier plataforma.

En el caso práctico que se desarrolló y se muestra en este artículo se utilizó la transformación de modelos, gracias a la implementación de esta técnica se pudo reducir el tiempo de modelamiento de los diferentes diagramas que sugiere la metodología UWE. También gracias a la transformación de modelos se puede garantizar que los modelos en sus diferentes etapas van a tener consistencia en cuanto a la semántica de las variables, tipos de datos y valores porque la transformación es semiautomática porque se utilizó una herramienta CASE. Con la consistencia de los modelos se busca que los desarrolladores no tengan que preocuparse por la generación de código porque este proviene de las transformaciones sucesivas de los modelos en los diferentes niveles que propone la arquitectura MDA.

El lenguaje QVT se encarga de realizar la transformación de los modelos. QVT asegura que la transformación de modelos tenga coherencia gracias a que realiza un mapeo de las diferentes variables utilizadas y las transforma según el modelo seleccionado. De igual manera este proceso es totalmente transparente para los desarrolladores porque la herramienta CASE se encarga de este proceso.

La transformación de modelos es semiautomática porque es necesaria la intervención del desarrollador para que maneje la herramienta CASE, es decir, que ingrese los datos iniciales de los modelos, para este caso los requerimientos que representan el modelo conceptual.

En un trabajo futuro se debe buscar una herramienta que tenga soporte para QVT y se puedan realizar las transformaciones entre niveles, es decir que realice el mapeo de cada modelo. Además para que se puede tener una línea de partida para generar un sistema web para diferentes plataformas como propone la arquitectura MDA.

8. REFERENCIAS BIBLIOGRÁFICAS

[1] Rossi, G. P., Schwabe, O., & D. Olsina, L. (2008). *Web Engineering - Modelling and Implementing Web Applications.* Londres: Springer

[2] *Ciberaula*. (2010). Obtenido de http://java.ciberaula.com/articulo/introduccion_mda/

[3] *OMG*. (1997-2010). Obtenido de http://www.omg.org/cgi-bin/doc?omg/03-06-01

[4] Ignjatovic, M. (s.f.). Obtenido de http://www.sofismo.ch/links/OMG-QVT_ObjektSpektrum_2006_E.pdf

[5] *Object Management Group.* (1997-2011). Recuperado el 2011, de http://www.omg.org/spec/QVT/1.0/PDF

[6] *Departamento de Lenguajes y Sistemas Informáticos.* (s.f.). Obtenido de http://www.lsi.us.es/docs/doctorado/memorias/MemoInvestigArturoHTorres.pdf

[7] Calero, C., Moraga, M., & Piattini, M. (2010). *Calidad del producto y proceso software.* Madrid: RA-MA.

[8] *CMSMATRIX*. (s.f.). Recuperado el 24 de Abril de 2011, de http://cmsmatrix.org/matrix/cms-matrix

[9]*Corporación Sybven*. (2010). Recuperado el 2010, de http://www.corporacionsybven.com/portal/index.php?option=com_content&view=article&id=246

[10] *Desarrolloweb.com*. (s.f.). Recuperado el 2010, de http://www.desarrolloweb.com/articulos/497.php