Dyna
ISSN: 0012-7353
dyna@unalmed.edu.co
Universidad Nacional de Colombia
Colombia

MONTENEGRO MARÍN, CARLOS ENRIQUE; GAONA GARCÍA, PAULO ALONSO; CUEVA LOVELLE, JUAN MANUEL; SANJUAN MARTÍNEZ, OSCAR

APLICACIÓN DE INGENIERÍA DIRIGIDA POR MODELOS (MDA), PARA LA CONSTRUCCIÓN DE UNA HERRAMIENTA DE MODELADO DE DOMINIO ESPECÍFICO (DSM) Y LA CREACIÓN DE MÓDULOS EN SISTEMAS DE GESTIÓN DE APRENDIZAJE (LMS) INDEPENDIENTES DE LA PLATAFORMA

Dyna, vol. 78, núm. 169, 2011, pp. 43-52
Universidad Nacional de Colombia
Medellín, Colombia

APLICACIÓN DE INGENIERÍA DIRIGIDA POR MODELOS (MDA), PARA LA CONSTRUCCIÓN DE UNA HERRAMIENTA DE MODELADO DE DOMINIO ESPECÍFICO (DSM) Y LA CREACIÓN DE MÓDULOS EN SISTEMAS DE GESTIÓN DE APRENDIZAJE (LMS) INDEPENDIENTES DE LA PLATAFORMA

APPLICATION OF MODEL-DRIVEN ENGINEERING (MDA) FOR THE CONSTRUCTION OF A TOOL FOR DOMAIN-SPECIFIC MODELING (DSM) AND THE CREATION OF MODULES IN LEARNING MANAGEMENT SYSTEMS (LMS) PLATFORM INDEPENDENT

CARLOS ENRIQUE MONTENEGRO MARÍN

M Sc., Dr. (c), Profesor Universidad Distrital, Bogotá, Colombia, cemontenegrom@udistrital.edu.co

PAULO ALONSO GAONA GARCÍA

M Sc., Profesor, Universidad Distrital, Bogotá, Colombia, - pagaonag@udistrital.edu.co

JUAN MANUEL CUEVA LOVELLE

PhD., Universidad de Oviedo, España,- cueva@uniovi.es

OSCAR SANJUAN MARTÍNEZ

PhD., Profesor, Universidad de Oviedo, España - sanjuan@uniovi.es

Recibido para revisar Abril 11 de 2011, aceptado Junio 20 de 2011, versión final Julio 25 de 2011

RESUMEN: El objetivo general es realizar un modelado específico de dominio para la construcción de módulos en sistemas de gestión del aprendizaje (LMS) independientes de la plataforma. Para esto, el punto de partida es un metamodelo para la construcción de un lenguaje específico de dominio (DSL) que con ingeniería dirigida por modelos (MDE) y aplicando las debidas transformaciones se consiga de un modelo independiente de la plataforma, el despliegue de este modelo se realizara en varias plataformas LMSs.

PALABRAS CLAVE: Arquitecrura Dirigida por Modelos (MDA); Metamodelo; Meta-metamodelo Ecore; Framework de Modelado para Eclipse (EMF); Metamodelo de Objetos Smiple (MOF); Framework de Modelado Gráfico(GMF), moodle.

ABSTRACT: The general objective is to make domain-specific modeling for the construction of modules of learning management systems (LMS) platform independent. For this, the start point is a metamodel for the construction a domain specific language (DSL), that with model-driven engineering (MDE) and applying the appropriate transformations are achieved from a independent model platform, the deployment of this model was made in several LMS platforms.

KEYWORDS: Model-Driven Architecture (MDA); Metamodel; Ecore Meta-metamodel; Eclipse Modeling Framework (EMF); Meta Object Facility (MOF); Graphical Modeling Framework (GMF), moodle.

1. INTRODUCCION

MDE (Model-Driven Engineering) o ingeniería dirigida por modelos es una propuesta que se ha venido trabajando desde hace varios años por autores como [1-5], los cuales plantean el uso de modelos como eje fundamental en todo el ciclo de vida de un proyecto de software y que reduce el tiempo y esfuerzo en el desarrollo, una de las aproximaciones de este planteamiento es MDA (Model-Driven Architecture) que es la proposición de la OMG (Object Manage Group) la cuál establece una serie de tecnologías a utilizar en la construcción de software bajo el esquema de MDE, este articulo aplica MDA al problema de la interoperabilidad entre LMSs (Learning Management System) y se orienta a presentar un framework independiente de la

plataforma y que empleando MDA logre desplegar módulos en plataformas LMS específicas, las secciones dos y tres abordan estas temáticas. La sección cuatro muestra el desarrollo de un prototipo en eclipse para dar solución al problema mediante la creación de un metamodelo bajo ecore, después la construcción de una herramienta DSM (Domain Specific Modeling) gráfica basada en ese metamodelo y con apoyo en Eclipse Modeling Framework (EMF), Meta Object Facility (MOF) y Graphical Modeling Framework (GMF) para la representación de un modelo independiente de la plataforma. Finalmente se mostrara como transformar el modelo construido con el DSM a código desplegable para algunas plataformas LMS, a fin de validar su funcionamiento, esto se hará con MOFScript. Por último en la sección cinco se realiza una discusión de las pruebas y validación de la herramienta.

2. ESTADO DEL ARTE

Esta propuesta enfrentara el problema de la interoperabilidad entre LMSs, a partir de la generación de una ontología para la posterior creación de un metamodelo. La generación de la ontología ya se ha tratado de hacer en proyectos como [6], aquí se pretende emplearla para modelar módulos dentro de un LMS, que sean desplegables en cualquier plataforma LMS, una propuesta similar pero desde otro enfoque se trata en [7-9], las cuáles facilitan una especificación integrada de las arquitecturas de diferentes plataformas LMS. Este framework independiente de la plataforma se puede utilizar para especificar y clasificar los LMSs existentes o futuras gestiones de ellos, para simplificar la migración de datos entre diferentes tipos de sistemas e-Learning, los artículos dejan abierta la problemática existente entre la multiplicidad de plataformas. En el trabajo [10] se muestra el gran problema de heterogeneidad, por la utilización de diversos lenguajes de programación sobre los cuales están hechos los LMSs y en [11] hablan sobre la falta de integración entre los LMSs existentes a pesar de sus funcionalidades similares.

Los trabajos [6-12] han tratado la problemática de la interoperabilidad entre LMSs, otra propuesta como [13], plantean la solución a este problema con la generación de servicios, y evidencian que esta característica la poseerán las próximas generaciones de LMSs, pero que los actuales no poseen, en [7-9] se plantea la creación de un framework que integre la especificación de una arquitectura general para los LMSs y utilice MDA para migrar de una plataforma a otra, las pruebas se realizaron sobre las plataformas OLAT y Moodle, pero únicamente trataron el área de administración de objetos de aprendizaje y derechos de autor, la mejor aproximación que se encontró a lo que se plantea en esta investigación, se halló en [14] y presentan un lenguaje visual y de texto, específico para la creación de diseños de aprendizaje independientes de la plataforma, algunas otras aplicaciones relacionadas con la temática son CDF[15], LMML[16], PALO[17], Targetam[18], pero estas no consideran los módulos que posee un LMS. La conclusión de esta indagación es que la solución a la problemática se ha atacado desde varios frentes, algunas desde la perspectiva de MDA pero ninguna contempla los módulos actuales que posee un LMS.

Para poder abordar esta problemática, se plantea el diseño y creación de una ontología LMS, trabajos como [19, 20] presentan una ontología muy genérica para LMSs, ó [21, 22] en los que se muestra una ontología para integrar e implementar un LMS en una organización, ó propuestas trabajadas en [23] que presentan una ontología un poco más completa pero enfocada en los niveles B de las especificaciones IMS LD [24]; estos modelos pueden ser tomados como base pero ninguno trata los módulos que componen un LMS. Posteriormente se creará un metamodelo con la ontología obtenida, trabajos como [25] lo han hecho pero nuevamente enfocados a niveles B del IMS LD, pero al igual que con las ontologías, ninguna de las propuestas suple las necesidades de este proyecto, con lo cuál es necesario la creación de este metamodelo para módulos que posee un LMS.

3. INGENIERÍA DIRIGIDA POR MODELOS (MDE)

La ingeniería dirigida por modelos surge como la respuesta de la ingeniería de software a la industrialización del desarrollo de software, MDA es la propuesta de la OMG, que centra sus esfuerzos en reconocer, que la interoperabilidad es fundamental y el desarrollo de modelos permite la generación de otros modelos que luego al ser juntados proveerán la solución a todo un sistema e independiza el desarrollo de las tecnologías empleadas [26].

Una buena interpretación de lo que es un modelo, un metamodelo y un meta-metamodelo se encuentra en [27], allí un metamodelo son esas herramientas

que permite la creación de un modelo, que es una descripción de uno o varios elementos del dominio o mundo real y finalmente el meta-metamodelo describe a esos metamodelos planteados, generando un grado de abstracción supremamente alto en el cuál coinciden todos los modelos, vea la Figura 1.

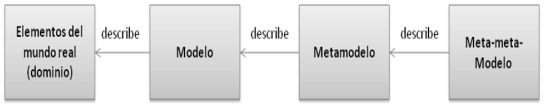


Figura 1 Modelo, metamodelo y meta-metamodelo.

Según [27] básicamente hay cuatro espacios de modelamiento, los niveles base M0 que son los elementos del mundo real, los niveles M1 que son los programas informáticos, los niveles M2 que sería la especificación de UML, ODM, Java, C#, XML u otras y que para este caso será el metamodelo a construir, finalmente están los niveles M3 que son los de mayor abstracción. Básicamente hay dos meta-metamodelos (M3), planteados; por un lado está MOF (Meta Object Facility) y por el otro EBNF (Extended Backus-Naur Form). La idea de generar estos niveles de abstracción tan altos, es proveer un mecanismo común que permita a través de transformación de un modelo a otro la interoperabilidad de los sistemas.

El artefacto que reúne los requerimientos del sistema se llama el CIM, el resultado de modelar este sistema es un PIM que se hace a través de un DSM construido previamente. Este DSM genera a través de un proceso de transformación un PSM, que por ultimo y nuevamente a través de otra transformación se convierte en código desplegable o ISM, este proceso se visualiza en la figura 2. En este trabajo se expondrá la creación de ese DSM basado en un metamodelo para la creación de módulos de comunicación para un LMS, su posterior modelamiento con la herramienta DSM creada, luego la transformación a código compatible con las plataformas LMS moodle, claroline y atutor, para finalmente hacer el despliegue y pruebas de los módulos creados.

4. INGENIERÍA DIRIGIDA POR MODELOS (MDE)CON ECLIPSE

A. Meta-metamodelo

El meta-metamodelo que se empleara será ecore que es el metamodelo que utiliza eclipse [28], este se encuentra en el paquete org.eclipse.emf.ecore y es la especificación más alta que existe en la pirámide de los modelos (M3), sobre ella se construirá el metamodelo del proyecto, la especificación de ecore se puede consultar en [29].

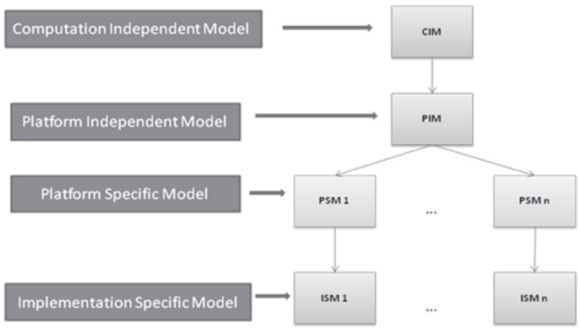


Figura 2 Modelos generados en MDA.

B. Metamodelo

Para la utilización de ecore en Eclipse es necesario tener instalado el plugin de EMF (Eclipse Modeling Framework), este plugin provee básicamente dos herramientas para construir un modelo basado en ecore, una el *Ecore Model* que es un editor manual que funciona en un estilo de árbol de navegación para la creación del modelo basado en ecore, la otra es el *Ecore Diagram* siendo este un editor grafico similar a las herramientas gráficas para la creación de diagramas de clases UML. Cualquiera de las dos forma que se utilice para crear el diagrama basado en ecore, genera un fichero XMI[30] (XML Metadata Interchange) que es una especificación para el intercambio de diagramas, en este caso se utilizará *Ecore Model.*

Para la construcción del metamodelo, en este trabajo se ha tomado únicamente lo concerniente a los módulos de comunicación que tiene un LMS, por ello, esta propuesta se basa en el metamodelo LMS planteado en [31] y la definición del dominio realizada en [32]. El metamodelo creado está construido sobre ecore y en la figura 3 se visualizan los módulos de comunicaciones. El metamodelo en esencia posee seis módulos llamados EClass en ecore que son: Forum, Chat, Wiki, Annoncement, News y Note, todos están relacionados con la EClass Communications, pues el módulo Communications puede tener cero o muchos módulos de los anteriores, esta EClass Communications hereda de la EClass Tools y todas las EClass están contenidas en

la EClass LMSModel, esta relación es obligatoria en todo metamodelo pues es quien representara el contenedor de EClasses, y allí será en donde se desplieguen los módulos a modelar o mejor dicho donde estarán contenidos, es necesario aclarar que la EClass LMSModel solo almacenara cero o una EClass Communications, esto tiene sentido pues no puede haber más de un modulo Communications en un LMS, ya que este a su vez almacena las herramientas de Communications tales como (Forum, Chat, Wiki, Annoncement, News y Note). Por último cada EClass tiene sus propios atributos que la compone, ver la figura 3.

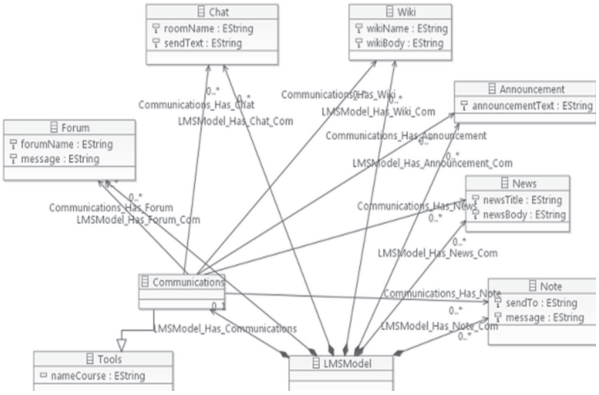


Figura 3 Metamodelo LMS de módulos de comulación.

C. Construcción del editor para el modelo o DSM

Como se está trabajando bajo eclipse, para esta etapa se empleó EMF Tooling (Graphical Modeling Framework Tooling) que hace parte del proyecto GMP (Graphical Modeling Project)[33]. El proceso para la construcción del DSL Grafico se visualiza en la figura 4 y una excelente guía para el desarrollo de herramientas de este tipo la encontramos en [34] y en los tutoriales que ofrece su Web oficial [35].

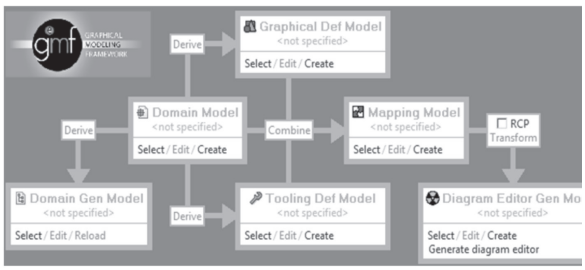


Figura 4 GMF Overview o dashboard

Según el dashboard de la figura 5 lo primero que se debe crear es el Domain Model, este corresponde al LMS metamodel descrito en la sección anterior. El siguiente paso es crear el Domain Gen Model, que es un modelo que permite transformar automáticamente el modelo ecore a código fuente. El código se genera aplicando patrones de transformación. El resultado es un conjunto de clases java, que serán utilizadas más adelante en la herramienta DSM.

Luego se crea el Graphical Def Model, este es usado para definir las figuras, nodos, conexiones, etc. El resultado es un fichero con la siguiente estructura; un Canvas (lienzo) en la raíz con una galería de figuras base que contiene elementos de Rectángulos, Etiquetas y Conexiones de Polilineas. Estas son usadas por el correspondiente elemento Nodo, Etiqueta del diagrama y Conexión para representar los temas del domain model.

El siguiente paso es la creacion del Tooling Def Model, este es usado para especificar la paleta (Pallete) de herramientas de creación, acciones, etc, para los elementos gráficos. Allí existe un elemento en el nivel superior "Tool Registry", en el que se encuentra una paleta (Palette). La "Palette" contiene un "Tools Group" con elementos de tipo "Creation Tool" para los nodos tema y conexiones para elementos de subtemas.

El "mapping model" es el siguiente paso en el dashboard, el "mapping model" combina los tres modelos: el "Domain Model", el "Graphical Def Model", y el "Tooling Def Model".

El último paso es la creación del "Diagram Editor Gen Model", aquí se establecen las propiedades para la generación de código, similar al EMF genmodel. A partir de este modelo se obtiene un plugin para eclipse que contiene la herramienta DSM construida. La apariencia de la herramienta se muestra en la figura 5.

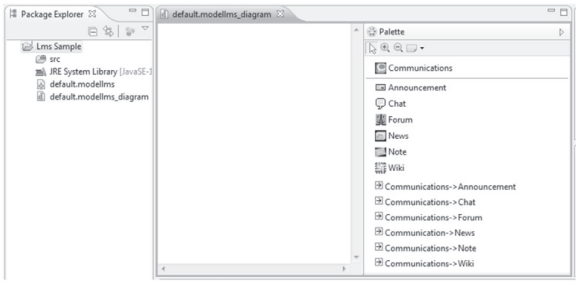


Figura 5 Herramienta DSM para modelar módulos en plataformas LMS.

D. Modelo

El diagrama que se obtenga como resultado de emplear el editor de DSM, tendrá asociado un fichero XMI[30] que basará su sintaxis en el metamodelo creado.

El funcionamiento de la herramienta es muy sencillo, para crear los módulos basta con arrastrar los nodos de la "Pallete" al área de trabajo, rellenar los campos y conectarlos respetando las siguientes reglas:

- Un cursos solo puede tener un modulo de "Communications".
- El modulo de "Communications" tiene cero o muchos: "Announcements", "Chat", "Forum", "News", "Note", y "Wiki".
- La herramienta valida los siguientes casos:
- Solo permite un modulo de "Communications".
- Los enlaces solo pueden corresponder entre el nodo "Communications" y su respectiva herramienta, por ejemplo "Communications -> Chat ", solo sirve para conectar el nodo "Communications" con el nodo "Chat", la herramienta no permite utilizarlo en otro caso.
- Cuando se genere el código a desplegar, aquellos nodos que estén sueltos (sin conexión) no serán tenidos en cuenta para la creación del curso.

En resumen, en esta sección se hablo de cómo crear una herramienta DSM gráfica (M1), basada en un metamodelo generando (M2) y que a su vez se basa sobre el meta-meta modelo ecore (M3), el siguiente paso es convertir este nuevo modelo que se obtuvo con la herramienta DSM y hacerle una transformación a código (M0), esta idea se explicara en la siguiente sección.

E. Proceso de generación de código

En este último paso, existen varias tecnologías que se integran a Eclipse, como son Acceleo[36], Xpand[37] y MOFScript[38], todas emplean el mismo principio, la creación de reglas de transformación basándose en un metamodelo. Estas reglas serán aplicadas al modelo, para generar código en el lenguaje deseado, este tipo de tecnologías reciben el nombre M2T o Model to Text. La transformación de un modelo a otro se llama de M2M o Model to Model, algunas herramientas como ATL[39] o OperationalQVT, realizan esta tarea. Para este caso se empleara MOFScript [38].

MOFScript es un lenguaje basado en reglas, presentado por la OMG, para realizar transformaciones de modelo a texto (M2T). Se puede instalar como un plugin de eclipse y tanto su instalación como su uso no son complicados de entender. Un excelente manual para el uso de MOFScript es [40].

A continuación se muestran los pasos a seguir para generar código a partir del modelo LMS creado con la herramienta DSM. Para poder comprender el contenido del fichero MOFScript es necesario entender cómo se desarrolla un modulo para cada una de las plataformas LMS sobre el cuál se desplegara el modelo. En el caso de moodle las mejores guías se encuentran en su página Web [41] vinculo Develoment, o en los trabajos [42] y [43], para el caso de claroline estan en [44] y para atutor en [45, 46]. Este articulo no ahondará en este tema, pues no es el objetivo.

Para transformar el modelo que se obtiene con la herramienta DSM creada hay que tener en cuenta que este modelo estará basado en el metamodelo LMS construido anteriormente, MOFScript tiene su propia sintaxis que pude ser consultada en [40].

Lo primero que se debe hacer es definir el modelo de entrada a las plantillas MOFScript, para ello se debe declarar la transformación con *texttransformation* darle un nombre a la trasformación y enviarle como parámetro de entrada el nombre del metamodelo con extensión .ecore, para este caso *modellms*, también se debe dar un nombre al modelo enviado, para este caso se llamo *mlms*. Esto se puede visualizar el siguiente fragmento de código Figura 6.

```
texttransformation
modellmsTransformationMoodle
(in mlms:"modellms") {
```

Figura 6 Declaración de una transformación en MOFScript.

Para crear un nuevo fichero se utiliza la instrucción "file" como se ve a continuación:

```
file ('mod_form.php');
```

Figura 7 Creación de fichero mod_form.php en MOFScript para moodle.

Y para escribir en ese fichero las cadenas deben ir entre el carácter " ' ". Para la declaración de funciones, primero se debe anteponer la palabra "module" seguida de el operados "::" y luego el código correspondiente, a continuación se muestran la función encabezado para las plantillas de transformación de moodle.

```
    module::encabezado()
    {
'<?php
require_once($CFG-
>dirroot.'+quote+'/course/moodleform_mod.php'
+quote+');
require_once($CFG-
>dirroot.'+quote+'/course/lib.php'+quote+');
require_once('+quote+'../config.php'+quote+');
class mod_plantilladsl_mod_form extends
 moodleform_mod {
    function definition() {'
    }
```

Figura 8 Función encabezado en MOFScript para moodle.

Para invocar a las funciones sencillamente se llaman en donde sean necesarias, para contar cuantos módulos (EClass) hay en el modelos se utiliza la función "size()", provista por las conexiones, esto se debe hacer por cada nodo, a continuación se muestra un ejemplo para los módulos Forum.

```
self.LMSModel_Has_Forum_Com.size();
```

Figura 9 Contador de módulos Forum dentro de un modelo.

Luego se recorren todos los módulos que están dentro del modelo y conectados al modulo Communications, se invoca a la función correspondiente con parámetros de entrada los nombres ingresados en el modelo y se incrementa un contador de los módulos creados correctamente. A continuación se muestra el código correspondientes para el modulo Forum en moodle, esto se debe hacer por cada modulo.

```
elf.LMSModel_Has_Communications.Communications_
Has_Forum->forEach(forum:mlms.Forum)
{
   if (forum.forumName = null or forum.message
= null){
      numForumSN = numForumSN+1;
   }
   else{
      newline(1);
      tab(2);
      'def_forum_dsl
('+quote+forum.forumName+quote+','+quote+forum.
message+quote+');';
      numForumY = numForumY+1;
      }
}
```

Figura 10 Recorrido para los módulos Forum en MOScript para moodle.

Finalmente aplicando las tecnologías de transformación sobre el modelo creado se obtiene el código en el formato definido para su despliegue sobre la plataforma seleccionada, en este caso moodle, claroline o atutor. Si desea descargar todo el proyecto se puede hacer de http://www.vicegd.com/KiwiDSMCode.rar

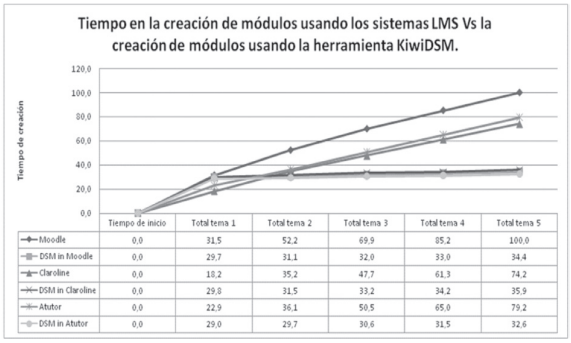
5. PRUEBAS Y VALIDACION DE LA HERRAMINSTA DSL

El objetivo de esta sección es validar el metamodelo planteado y comprobar si la hipótesis "Las herramientas para diseño basado en modelos reducen el tiempo de desarrollo en los proyectos" es válida. Las pruebas medirán el tiempo y esfuerzo (usabilidad) de los usuarios para crear exactamente los mismos módulos en moodle, claroline, atutor y la herramienta DSM creada, cada uno de los usuario creara 5 temas y cada uno de los temas tendrá en este orden: 1 Chat, 1 Forum, 1 Wiki, 1 Announcement, 1 News y 1 Note a todas las personas inscritas en ese curso, la información de cada modulo será la misma en todos los casos y los valores se han normalizado al mayor.

De acuerdo con [47] el esfuerzo se define como "cantidad de ocasiones en las que usuario selecciona o ingresa algún tipo de información al sistema". La medición de tiempos y esfuerzo se realizo para todos los sistemas (la herramienta DSM y las plataformas LMS) y se midió: tiempo y esfuerzo en la creación de cada herramienta por tema, tiempo y esfuerzo en la creación de todas las herramientas que conforman un tema, tiempo y esfuerzo total en la creación de los cinco temas acompañados de sus herramientas.

La figura 11 muestra una comparación respecto al tiempo en la creación de módulos con la propia plataforma LMS versus la utilización de de la herramientas DSM construida y se visualiza que la utilización de una herramienta DSM reduce los tiempos a partir de la primera generación, y que esta diferencia crece a favor de de la herramienta DSM (gastando menos tiempo) respecto a cualquier plataforma LMS a medida que se generan más temas con sus respectivos módulos. Así se ve que el mayor y menor tiempo al finalizar la creación del quinto tema con la herramienta DSM es 35.9% y 32.6% respectivamente, mientras que con las plataformas LMS es de 74.2% y 100% respectivamente, mostrando así una reducción bastante

considerable en tiempo que favorece la utilización de herramientas DSM en plataformas de educación virtual y que se podría extender a diversos campos de acción.

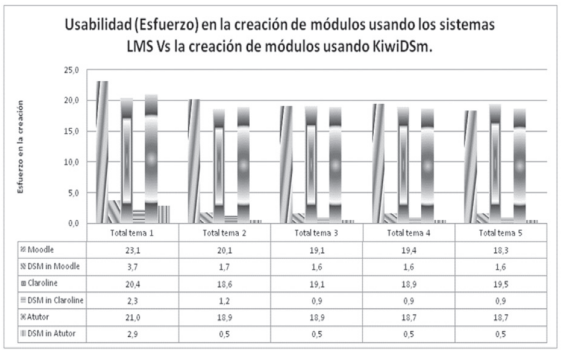


	Tiempo de inicio	Total tema 1	Total tema 2	Total tema 3	Total tema 4	Total tema 5
Moodle	0,0	31,5	52,2	69,9	85,2	100,0
DSM in Moodle	0,0	29,7	31,1	32,0	33,0	34,4
Claroline	0,0	18,2	35,2	47,7	61,3	74,2
DSM in Claroline	0,0	29,8	31,5	33,2	34,2	35,9
Atutor	0,0	22,9	36,1	50,5	65,0	79,2
DSM in Atutor	0,0	29,0	29,7	30,6	31,5	32,6

Figura 11 Tiempo en la creación de módulosusando los sistemas LMS Vs la creación de losmismos módulos con la herramienta DSM.

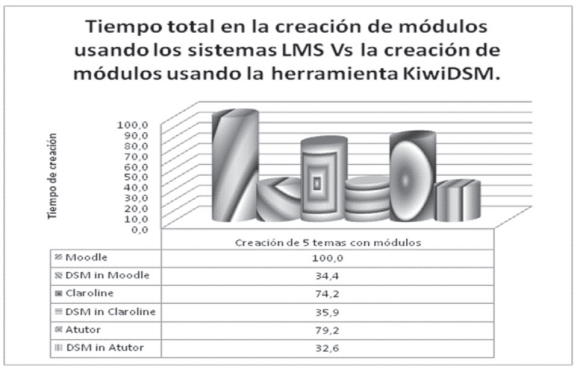
Un comportamiento muy similar se ve en la medición de esfuerzo, como lo muestra la figura 12, en la cuál y en contraste de lo que ocurre con el tiempo, la diferencia de esfuerzo se ve plasmada desde el principio en favor de hacer menos esfuerzo con la herramienta DSM, logrando mínimas y máximas diferencias hasta del 16.7% y 20.8% respectivamente, en ambos casos aparecen en la generación del primer tema.

Por último en la figuras 13 y 14, se muestran consolidados los resultados totales de las pruebas respecto a tiempo y esfuerzo, en la creación de módulos con la herramienta DSM versus la creación de módulos en cada plataforma y es evidente en la comparativa, la reducción en los dos casos logrando un máximo de reducción del 67.4% en tiempo y 72.5% en esfuerzo.



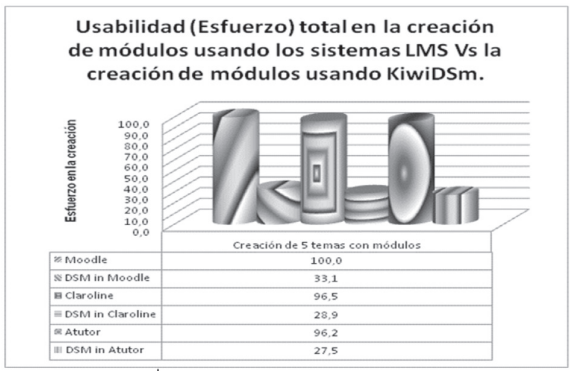
	Total tema 1	Total tema 2	Total tema 3	Total tema 4	Total tema 5
Moodle	23,1	20,1	19,1	19,4	18,3
DSM in Moodle	3,7	1,7	1,6	1,6	1,6
Claroline	20,4	18,6	19,1	18,9	19,5
DSM in Claroline	2,3	1,2	0,9	0,9	0,9
Atutor	21,0	18,9	18,9	18,7	18,7
DSM in Atutor	2,9	0,5	0,5	0,5	0,5

Figura 12 Esfuerzo (usabilidad) en la creación de módulos usando los sistemas LMS Vs la creación de los mismos módulos con la herramienta DSM.



Moodle	100,0
DSM in Moodle	34,4
Claroline	74,2
DSM in Claroline	35,9
Atutor	79,2
DSM in Atutor	32,6

Figura 13 Tiempos totales en la creación de módulos usando los sistemas LMS Vs la creación de los mismos módulos con la herramienta DSM.



Moodle	100,0
DSM in Moodle	33,1
Claroline	96,5
DSM in Claroline	28,9
Atutor	96,2
DSM in Atutor	27,5

Figura 14 Esfuerzos (Usabilidad) totales en la creación de módulos usando los sistemas LMS Vs la creación de los mismos módulos con la herramienta DSM.

Para la validación del metamodelo y la herramienta DSM creada se han utilizado redes de petri y se basaron en las propiedades de los requisitos definidos en el metamodelo, con el objetivo de validar, que el modelo construido cumplía los requisitos del metamodelo, dichas pruebas se hicieron a los módulos desplegados en las tres plataformas LMS.

Se definieron un conjunto de pruebas que describen la actividad que se puede realizar con la herramienta DSM hasta su despliegue en las plataformas LMS, dichas pruebas corresponden a diferentes comportamientos que se puede dar para cumplir un requisito. Para cada prueba se hizo la siguiente formalización:

Prueba = { <**fórmula, resultado**> } donde,
fórmula Є a una composición <entrada, salida>
resultado = 1 (verdadero), si la fórmula modela un comportamiento válido y 0 (falso), si la fórmula modela un comportamiento inválido.

Para estas pruebas se incluyeron comportamientos válidos y no válidos. Si la prueba pasa satisfactoriamente en un comportamiento no válido indica que el modelo tiene fallas. Cada prueba se modelo con una red de petri, con el fin observar su comportamiento y así encontrar errores en el modelo.

Una Red de Petri: es una cuádrupla **R = {P, T, I, O}** donde,

P es un conjunto finito y no vacío de nodos, dadas como condiciones
T es un conjunto finito y no vacío de transiciones, dadas como eventos
$P \cap T = 0$
I:P x T → función de entrada
O:T x P → función de salida

A continuación se describirán las pruebas definidas para la creación del módulo Foro, la red de petri resultante se puede visualizar en la Figura 16.

Se tienen los siguientes nodos y transiciones:

P = {p1, p2, p3, p4, p5, pFin}

T = {t1, t2, t3, t4, t5, t6}

p1 = Esperando elementos en el Canvas.

p2 = Nodo Communications creado en el Canvas.

p3 = Nodo Forum creado en el Canvas.

p4 = Dato forumName asignado al Forum.

p5 = Dato message asignado al Forum.

pFin = Módulo Forum desplegado en la plataforma.

t1 = Se selecciono el nodo Communications de la paleta de herramientas y se coloco en el Canvas.

t2 = Se selecciono el nodo Forum de la paleta de herramientas y se coloco en el Canvas.

t3 = Se selecciono la conexión Communications -> Forum de la paleta de herramientas y se conecto el nodo Communications con el nodo Forum.

t4 = Se asigno información al campo forumName del nodo Forum.

t5 = Se asigno información al campo message del nodo Forum.

t6 = Se realizan las debidas transformaciones y despliegue sobre la plataforma LMS.

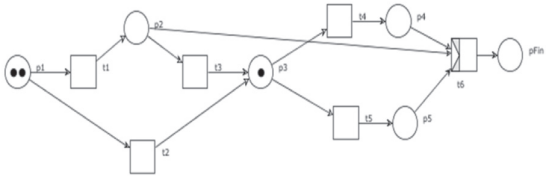


Figura 15 Red de petri para la creación de módulos Forum

Ejemplo de Comportamiento valido:

Prueba1 = {<p1, t1>, <t1, p2>, <p1, t2>, <t2, p3>, <p3, t4>, <p3, t5>, <t4, p4>, <t4, p5>, <t6, pFin>,1}

En esta prueba se obtuvo como resultado 1, lo que indica que este comportamiento se da en el modelo, y por consiguiente cumple con el requisito.

Ejemplo Comportamientos no valido:

Prueba2 = {<p1, t2>, <t2, p3>, <p3, t4>, <t4, p4>, <p3, t5>, <t5, p5>, 0}

En esta prueba se obtuvo como resultado 0, lo que indica que este comportamiento no se da en

Así mismo se crearon redes de petri para los demás módulos (Announcement, Chat, Wiki, News y Note) con sus respectivas pruebas de validación y se concluyo que el metamodelo es válido para todos los casos, cumpliendo el requisito de homogeneidad con todos los LMSs, eso quiere decir que es independiente de la plataforma ósea interoperable entre todos los LMSs.

6. CONCLUSIONES Y TRABAJO FUTURO

Basándose en los resultados de las pruebas, se puede afirmar que MDE es aplicable en el contexto de diseño de cursos para LMSs y que reduce significativamente el tiempo y esfuerzo en la construcción y despliegue de cursos sobre plataformas LMS.

El metamodelo planteado es homogéneo con los LMSs trabajados y su valides se soporta sobre las simulaciones manejadas con redes de petri, logrando ser este metamodelo un punto de partida para la construcción de un metamodelo LMS más completo que integre mas módulos y plataformas LMS, de igual

forma se podría realizar con la definición y validación del metamodelo planteado un proceso inverso en el cuál, los módulos que ya estén creados en un LMS específico se conviertan a instancias del metamodelo y de esta forma ya se podría hacer un proceso de transformación a cualquier otra plataforma LMS.

En esta propuesta se han explorado los niveles M0, M1, M2 y M3 planeados en MDE, El metamodelo LMS construido representa el nivel M2, el meta-metamodelo Ecore es el nivel M3, el resultado de modelar los módulos con la herramienta DSM construida representa al nivel M1 y finalmente, las conversiones y despliegue de esos módulos construidos en las plataforma moodle, claroline y atutor representan el nivel M0.

REFERENCIAS

[1] García-Díaz, V. et al., Talisman mde: Mixing MDE principles, Journal of Systems and Software, vol. 83, pp. 1179-1191, 2010.

[2] García-Díaz, V. et al., Talisman mde Framework: An Architecture for Intelligent Model-Driven Engineering, in Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living. vol. 5518, S. Omatu, et al., Eds., ed: Springer Berlin / Heidelberg, pp. 299-306, 2009.

[3] Grob, H. L. et al., Model Driven Architecture (MDA): Integration and Model Reuse for Open Source eLearning Platforms, e-Learning and educations e, vol. eleed, Aug 2004.

[4] Kleppe, A. et al., MDA Explained: The Model Driven Architecture™: Practice and Promise Addison Wesley, 2003.

[5] Yonglin, L. et al., A transformation model from DEVS to SMP2 based on MDA, Simulation Modelling Practice and Theory, vol. 17, pp. 1690-1709, 2009.

[6] Dzemydiene, D. et al., An approach for managing educational resources in an adaptive e-learning system, International Journal of Information and Communication Technology Education, July, 200 P, 2007.

[7] Bizoňová, Z. et al., Model Driven e-Learning Platform Integration, in 2nd European Conference on Technology Enhanced Learning EC-TEL PROLEARN 2007 Doctoral Consortium, Crete, Greece, pp. 8-14, 2007.

[8] Bizonova, Z. and Ranc, D., Model Driven LMS Platform Integration, in Telecommunications, 2007. AICT, The Third Advanced International Conference, pp. 25-25. 2007.

[9] Bizonova, Z. and Ranc, D. Interoperability and Reuse Between Systems in eLearning, in World Conference on Educational Multimedia, Hypermedia and Telecommunications 2008, Vienna, Austria, pp. 1700-1705, 2008.

[10] Grob, H. L. et al., eleed - Model Driven Architecture (MDA): Integration and Model Reuse for Open Source eLearning Platforms, 2010.

[11] Moreno, N. and Romero, J. R., Recent Research Developments in Learning Technologies (2005), presented at the A MDA-based framework for building interoperable e-learning platforms, Badajoz, Spain, 2005.

[12] Perniu, L. et al., Eds., Elmset Project Contents of Proceedings Electronics 2006, Embedded Systems, Electronic Medical Equipment, Education in Electronics. Bulgaria: Technical University of Sofia, Bulgaria, 2006.

[13] Muñoz-Merino, P. J. et al., Enabling interoperability for LMS educational services, Computer Standards & Interfaces, vol. 31, pp. 484-498, 2009.

[14] Dodero, J. M. et al., An extensible approach to visually editing adaptive learning activities and designs based on services, Journal of Visual Languages & Computing, vol. 21, pp. 332-346, 2010.

[15] Verbert, K. and Duval, E., Towards a global architecture for learning objects: a comparative analysis of learning object content models, in ED-MEDIA 2004 World Conference on Educational Multimedia, Hypermedia and Telecommunications location: Lugano, Switzerland June 21-26, 2004.

[16] Slavin, R. E., Cooperative Learning: Theory, Research and Practice. Boston, 1995.

[17] Rodriguez-Artacho, M. et al., Using a high-level language to describe and create Web-based learning scenarios, in Frontiers in Education Conference, 1999. FIE '99. 29th Annual, vol.2, pp. 13A2/1-13A2/6, 1999.

[18] MICHAEL, K., Interoperable Community Platforms and Identity Management in the University Domain, ed, 2002.

[19] Heiyanthuduwage, S. R. and Karunaratne, D. D., A Learner Oriented Ontology of Metadata to Improve

Effectiveness of Learning Management Systems, International Journal of the Computer, the internet and management, vol. 14, 2006.

[20] Srimathi, H., Knowledge Representation of LMS using Ontology, International Journal of Computer Applications, vol. Volume 6 – No.3, 2010.

[21] Díaz-Antón, G. and Pérez, M., Hacia una ontología sobre LMS, presented at the Hacia una ontología sobre LMS, Colima, México, 2005.

[22] G. Díaz-Antón and M. A. Pérez, Towards an Ontology ,of LMS A Conceptual Framework, presented at the 8th International Conference on Enterprise Information Systems, Paphos - Cyprus, 2006.

[23] Amorim, R. R., et al., A Learning Design Ontology based on the IMS Specification, Educational Technology & Society, vol. 9, pp. 38-57, 2006.

[24] Consortium, I. G. L.,. Learning Design Specification. 2003. Available: http://www.imsglobal.org/learningdesign/

[25] Boticario, J. G. and Santos, O. C., An Open IMS-Based User Modelling Approach for Developing Adaptive Learning Management Systems, Journal of Interactive Media in Education, 2007.

[26] Stephen, J. M. et al., MDA Distilled: Principles of Model-Driven Architecture Addison Wesley, 2004.

[27] García Díaz, V. and Cueva Lovelle, J. M., Ingeniería Dirigida por Modelos, Ed. Oviedo, 2010.

[28] T. E. Foundation. 2010. http://www.eclipse.org/. Available: http://www.eclipse.org/

[29] I. C. a. others., Package org.eclipse.emf.ecore. 2006. Available: http://download.eclipse.org/modeling/emf/emf/javadoc/2.6.0/org/eclipse/emf/ecore/package-summary.html

[30] Group ,O. M., MOF 2.0/XMI Mapping, Version 2.1.1, Ed: Object Management Group, 2007, p. 120.

[31] Montenegro, C. et al., "Generation of metamodel in ecore with start point in an ontology for learning management systems (LMS), Journal of Web Engineering, Pending 2011.

[32] Montenegro, C. et al., Modeling and comparison study of modules in open source lms platforms with cmapstool, International Journal of Interactive Multimedia and Artificial Intelligence newsletter, 2010.

[33] Foundation, T. E. Graphical Modeling Project (GMP), 2010. Available: http://www.eclipse.org/modeling/gmp/

[34] Budinsky, F. et al., EMF: Eclipse Modeling Framework: Addison-Wesley, 2009.

[35] Foundation, T. E. GMF Tutorial. 2010. Available: http://wiki.eclipse.org/GMF_Tutorial

[36] Obeo, Acceleo. 2010. Available: http://www.eclipse.org/acceleo/

http://www.acceleo.org/pages/home/en

[37] Foundation, T. E. XPand. 2010. Available: http://wiki.eclipse.org/Xpand

[38] Foundation, T. E. MOFScript 2010. Available: http://www.eclipse.org/gmt/mofscript/

[39] Foundation, T. E. ATL, 2010. Available: http://www.eclipse.org/atl/

[40] J. Oldevik, MOFScript User Guide Version 0.8 (MOFScript v 1.3.6), 2009.

[41] Moodle. (2011, Feb). Moodle. Available: http://moodle.org/

[42] R. IVORRA, Tutorial: Creaciòn de un mòdulo actividad. Moodle (1.9.3), 2009.

[43] Gonzàlez, A. Guìa de apoyo para el uso moodle 1.9.4 Usuario Desarrollador, Informatica, Universidad de Oviedo, Oviedo, 2009.

[44] Consorcio Claroline. Claroline, 2008. [Internet]. Available: http://www.claroline.net

[45] ATutor. Module Development Documentation. 2011. Available: http://atutor.ca/development/documentation/modules.html#structure

[46] ATutor. ATutor Learning Managment Tools. 2011. Available: http://atutor.ca/

[47] Yamada, S. et al., Software-reliability growth with a Weibull test-effort: a model and application, Reliability, IEEE Transactions on, vol. 42, pp. 100-106, 1993.