Automatic Code Generation for Language-Learning Applications

G. Sebastián, R. Tesoriero, and J. Gallud

***Abstract*—Language-learning applications define exercises that are pedagogical tools to introduce new language concepts. The development of this type of applications is complex due to the diversity of language-learning methodologies, the variety of execution environments and the number of different technologies that can be used. This article proposes a complete Model-Driven Architecture (MDA) approach, from the definition of the Computational Independent Model (CIM layer) to the Implementation Specific Model (ISM layer), and the process of the necessary transformations for the automatic generation of the source code (in HTML and JavaScript) of language-learning applications. To carry out the model-to-model and model-to-text transformations, the ATLAS Transformation Language (ATL) and Acceleo transformation languages have been used respectively. The proposal has been validated through the modeling and the complete automatic generation of source code of two Learning Activity Mechanisms (LAM), which are used within methodologies such as Duolingo and Busuu: LAM Image-Audio-Text and LAM Audio-Text Options.**

***Index Terms*—Code generation, Language-learning applications, Model-driven architecture.**

I. Introducción

DEBIDO a la globalización y el uso extensivo de Internet, las personas están cada vez más interesadas en aprender un segundo e incluso un tercer idioma. Desde una perspectiva académica, el proceso de aprendizaje de idiomas extranjeros está definido por metodologías y respaldado por tecnología. Las aplicaciones de aprendizaje de idiomas se componen de ejercicios de aprendizaje, que son herramientas pedagógicas para introducir nuevos conceptos lingüísticos (vocabulario nuevo, gramática, etc.). El desarrollo de este tipo de aplicaciones es complejo debido a la diversidad de metodologías de aprendizaje de idiomas, la variedad de entornos de ejecución (web, móvil y de escritorio) y la cantidad de tecnologías diferentes que pueden utilizarse.

Este trabajo ha sido parcialmente financiado por la Junta de Comunidades de Castilla-La Mancha (España) y cofinanciado por el Fondo Europeo de Desarrollo Regional, con motivo del proyecto de investigación "Tecno-CRA".

G. Sebastián, Instituto de Investigación en Informática de Albacete. Universidad de Castilla-La Mancha, C/ Investigación, 2, 02071, Albacete, España, (email: gabriel.sebastian@uclm.es).

R. Tesoriero, Escuela Superior de Ingeniería Informática de Albacete. Universidad de Castilla-La Mancha, Avda. de España, s/n, 02071, Albacete, España, (email: ricardo.tesoriero@uclm.es).

J. A. Gallud. Escuela Superior de Ingeniería Informática de Albacete. Universidad de Castilla-La Mancha, Avda. de España, s/n, 02071, Albacete, España, (email: jose.gallud@uclm.es).

Este estudio describe la *arquitectura dirigida por modelos* (MDA) y el proceso de las transformaciones necesarias para la generación automática del código de este tipo de aplicaciones, basada en los meta-modelos presentados en [1] y [2]. En [2] se propone un enfoque basado en modelos para el desarrollo software de diferentes metodologías de aprendizaje de idiomas. Estas metodologías consisten en diferentes ejercicios de aprendizaje que se ejecutan en diferentes plataformas. La propuesta [1] describe un meta-modelo que define las entidades y sus relaciones para definir ejercicios de aprendizaje para aplicaciones de aprendizaje. Este meta-modelo permite pues a los diseñadores, el desarrollo de aplicaciones de aprendizaje de idiomas. Estos trabajos [1][2] ilustran la expresividad y el poder de reutilización para modelar actividades de aprendizaje de metodologías como *Duolingo* [3], *Babbel* [4] y *Busuu* [5].

La versión de *Eclipse Modeling Tools* utilizada en este trabajo, es la *IDE 2019-06*, que funciona tanto en *MacOS* como en *Windows*, incluyendo ATL versión 4.0.1 (*ATLAS Transformation Language*) [8], ACCELEO (ver. 3.7.8) [9], el editor reflexivo de modelos (ver. 1.21.0) y los *plugins* de transformaciones (ver. 1.21.0).

Las arquitecturas MDA, desacoplan la funcionalidad de una aplicación, de la tecnología de desarrollo o de implantación de la misma. Esto permite a los desarrolladores crear Modelos Independientes de Plataforma (PIM) y Modelos Específicos de Plataforma (PSM) para derivar el código fuente de la aplicación semi-automáticamente utilizando transformaciones de modelo. El meta-modelo presentado en [1] y [2] permite modelar lo necesario para el desarrollo de actividades de casi cualquier metodología de aprendizaje de idiomas: los contenidos de la metodología de aprendizaje, los flujos de trabajo de actividades de aprendizaje, los recursos de medios de aprendizaje, los mecanismos de interacción en las actividades de aprendizaje y el modelo de actividad de aprendizaje.

Se ha desarrollado un editor de modelado gráfico que utiliza el *framework GMF* para crear, editar y verificar los modelos de metodología de aprendizaje generados con el meta-modelo propuesto [2]. Ha sido desarrollado con un *plugin* de Eclipse que emplea el *Eclipse Modeling Framework* (EMF) [6] para seguir los estándares MDA OMG. El meta-modelo se definió en OclInEcore [7], que es un dialecto de OMG *Essential Meta-Object Facility* (EMOF) enriquecido con OCL. Este lenguaje se utiliza para definir los invariantes del modelo y las consultas que son la base para la verificación del modelo. Las transformaciones necesarias se han desarrollado de acuerdo

con el meta-modelo propuesto. Se ha realizado un desarrollo completo de transformaciones modelo a modelo y modelo a texto para la generación del código fuente de las aplicaciones con actividades de aprendizaje de idiomas utilizando los lenguajes de transformación ATL [8] y Acceleo [9] respectivamente.

A. Algunos Trabajos Relacionados

Si bien no se han encontrado trabajos que presenten meta-modelos para modelar lo necesario para el desarrollo de metodologías de aprendizaje de idiomas, en [10] se presenta un meta-modelo que permite definir modelos que representan ecosistemas de aprendizaje. Está basado en el patrón arquitectónico para ecosistemas de aprendizaje, y se puede implementar en contextos reales para resolver problemas de aprendizaje y de gestión del conocimiento. Para garantizar la calidad del meta-modelo del ecosistema de aprendizaje en [10] se presenta su validación a través de reglas de transformación Modelo a Modelo en ATL [8].

La arquitectura propuesta en este trabajo, incluye todos los niveles de abstracción clásicos del enfoque MDA, pero se constata que el nivel superior de abstracción, Modelos Independientes de la Computación (CIM), y su transformación a PIM rara vez se discute en trabajos de investigación. En [11] se representa un enfoque que permite dominar la transformación de CIM a PIM de acuerdo con MDA. El método se basa en crear un buen nivel CIM, utilizando reglas de construcción, para facilitar la transformación al nivel PIM. Dichas reglas de transformación, implementadas con ATL [8], aseguran una transformación semiautomática de CIM a PIM. Tiene un enfoque que se ajusta a la MDA, ya que permite considerar la dimensión empresarial en el nivel CIM. Para modelar el nivel de PIM usa UML, porque es el más recomendado por MDA en PIM. Otra característica interesante de la propuesta, es que da como resultado un conjunto de clases, organizadas de acuerdo con el *Model View Controller* (MVC).

El desarrollo de transformaciones de modelos implica una complejidad inherente al dominio de transformación, además de la complejidad del desarrollo de software en general. En [12] se presenta un *framework* que permite la especificación de la transformación de modelos con un alto nivel de abstracción, y después los transforma semi-automáticamente en modelos a un bajo nivel de abstracción, hasta el código de transformación. La propuesta es interesante, porque integra los elementos esenciales involucrados en el desarrollo de la transformación de modelos y permite la abstracción de los detalles tecnológicos.

En la propuesta de este trabajo se genera la presentación y los comportamientos de actividades de enseñanza de idiomas como una aplicación web (generando código HTML, CSS y JavaScript). En este sentido es interesante el trabajo [13] que presenta un enfoque MDA relativo al diseño de aplicaciones web basadas en *CodeIgniter*. En particular, en [13] se describe un meta-modelo (considerado como PSM) del *framework* PHP y también se especifica un conjunto de transformaciones (a través de reglas de transformación llevadas a cabo por ACCELEO [9]) para generar el código fuente de las aplicaciones modeladas teniendo en cuenta la arquitectura MVC de *CodeIgniter*.

Al igual que la propuesta [13] está validada mediante su uso en la generación de aplicaciones CRUD (Crear, Leer, Actualizar y Eliminar), en [14] se aplica también el enfoque MDA para generar, a partir del diagramas de clase UML, el código relativo a operaciones CRUD, en este caso de aplicaciones de *Windows Phone*. Esta contribución [14] aplica el enfoque por plantilla y usa también Acceleo [9] como un lenguaje de transformación. Las reglas de transformación definidas en [14] exploran el diagrama de clases instancia del modelo de origen, y generan archivos que contienen los códigos C# y XAML de acuerdo con el modelo objetivo.

En [15] se presenta la aplicación de MDA para generar, a partir un modelo (diagramas de clases UML), el código que sigue el patrón de modelado web MVC2, utilizando el estándar MOF 2.0 QVT (*Meta-Object Facility 2.0 Query-View-Transformation*) como lenguaje de transformación. Las reglas de transformación definidas en [15], aunque no generan como en el caso de este trabajo el código fuente final ejecutable en un entorno web, pueden generar, a partir del diagrama de clases, un archivo XML (que contiene las Acciones, los Formularios y las páginas JSP para el manejo de todas las operaciones CRUD), que se podría utilizar para generar el código necesario de una aplicación web.

El artículo está organizado en las siguientes secciones. En la sección II se presenta el problema y el contexto en el que se valida la investigación. En la sección III se describe la solución propuesta: una arquitectura MDA y el proceso de las transformaciones necesarias para la generación automática de código. Finalmente, se presentan los resultados, las conclusiones y el trabajo futuro.

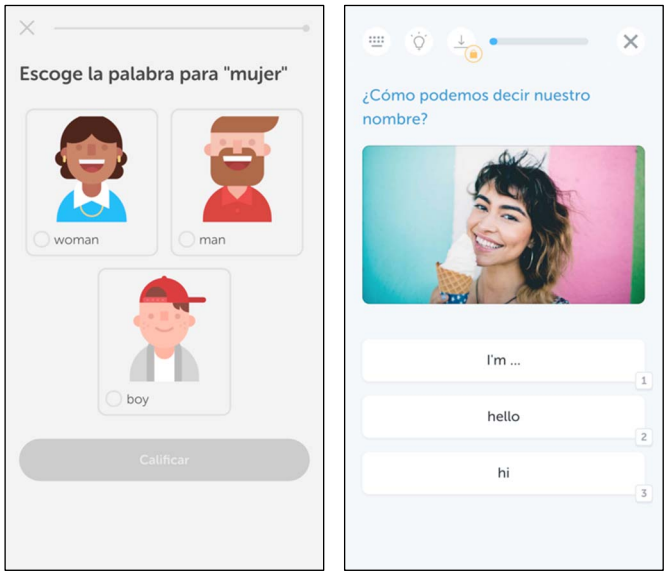
II. Planteamiento del Problema

A. Analizando las Metodologías de Enseñanza de Idiomas

El objetivo de esta subsección es presentar el subdominio de actividades de aprendizaje de idiomas que se implementan en las metodologías de aprendizaje de idiomas en entornos web, y que se ha utilizado para validar nuestra arquitectura MDA y el proceso de transformaciones hasta llegar a la generación de código fuente. Aunque la implementación de la mayoría de estas metodologías tiene un aspecto y comportamientos diferentes, comparten características comunes como los *Conceptos a Aprender* (CtL), la organización estructural de los contenidos (es decir, niveles, bloques, lecciones, actividades, etc.), los recursos multimedia (es decir, audios, textos, imágenes, etc.), los *Mecanismos de Actividades de Aprendizaje* (LAMs), etc.

Por ejemplo, la Fig. 1 representa el LAM de opción múltiple en dos metodologías de aprendizaje diferentes. La Fig. 1 (a) representa la versión *Duolingo* del mecanismo, que utiliza botones de opción asociados a las imágenes para mostrar las opciones disponibles, y un botón para confirmar la selección de la opción. La Fig. 1 (b) representa la respectiva versión del mecanismo en *Bussu*, que emplea un conjunto de botones

etiquetados con las opciones disponibles en lugar de los botones de opción. Además, ambas versiones también comparten características comunes en la presentación (por ejemplo, el proporcionar el progreso de la lección del alumno en la parte superior de la interfaz de usuario de la actividad de aprendizaje).



(a) (b)

Fig. 1. Actividad de aprendizaje de *elección-múltiple* en *Duolingo* (a) y *Busuu* (b).

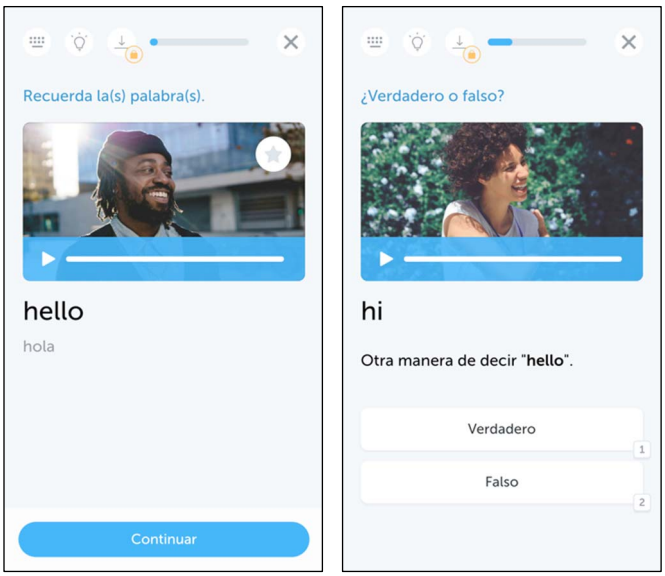
Los siguientes párrafos se centran en algunas características de la metodología de aprendizaje de *Busuu* como base para el caso de estudio. Esta descripción asume que el español es el idioma nativo del estudiante (SNL) y que el inglés es el idioma a aprender (LtL).

La metodología *Busuu* organiza, o estructura las actividades de aprendizaje de acuerdo con los siguientes 4 niveles de habilidad: Principiante (A1), Elemental (A2), Intermedio (B1) e Intermedio alto (B2). Cada nivel está compuesto por un conjunto de lecciones para lograr los objetivos de aprendizaje. Los objetivos se logran por medio de actividades de aprendizaje que se agrupan de acuerdo con *Conceptos a Aprender* (CtL). Los grupos de actividades de aprendizaje están compuestos por actividades de aprendizaje que emplean diferentes *Mecanismos de Actividades de Aprendizaje* (LAM).

El siguiente epígrafe describe 2 sencillas LAM utilizadas en *Busuu*, y que se utilizarán para validar la propuesta:

B. Mecanismos de Actividades de Aprendizaje en Busuu

LAM Imagen-Audio-Texto: El objetivo de este LAM es asociar un *concepto para aprender* (CtL) con su forma textual (es decir, una sola palabra, expresión del lenguaje u oración), con su forma visual (es decir, una ilustración, una fotografía o un gráfico) y –finalmente– con su locución. Para llevar a cabo esta tarea, la aplicación muestra a los alumnos la Interfaz de Usuario (IU) que se muestra en la Fig. 2 (a), donde pueden ver la asignación en SNL, es decir, "*Recuerda la(s) palabra(s)*", una imagen asociada al CtL, un reproductor de audio para escuchar este concepto en el LtL, un texto también asociado a este concepto en LtL (es decir, "*Hello*") y su traducción al SNL. Tan pronto como se muestra la interfaz de usuario, reproduce la locución asociada al CtL en SNL. Los estudiantes pueden repetir la locución haciendo *clic* en el botón de reproducción del reproductor de audio o pueden navegar a la siguiente actividad de aprendizaje haciendo *clic* en el botón "*Continuar*".



(a) (b)

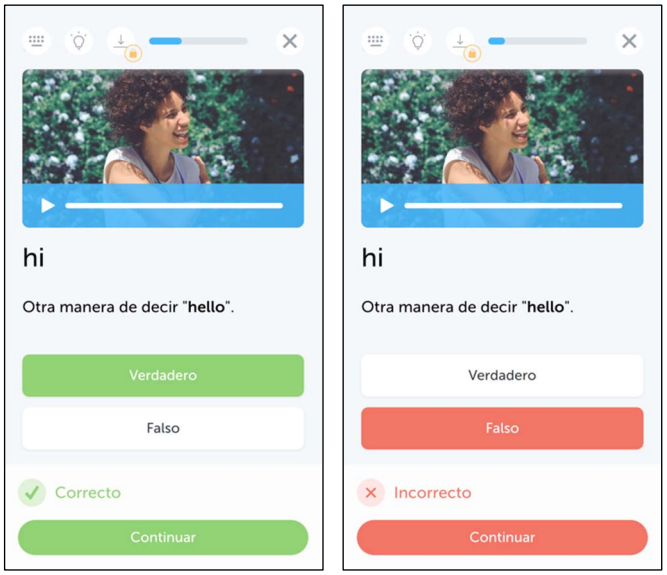
Fig. 2. Dos Mecanismos de Actividades de Aprendizajes en *Busuu*.

LAM Opciones de Audio-Texto: Este LAM es una variante del *LAM Imagen-Audio-Texto*, que solicita a los estudiantes que seleccionen una opción para responder una pregunta en SNL relacionada con un CtL. La interfaz de usuario de la aplicación para realizar esta asignación, se muestra en la Fig. 2 (b). Muestra la declaración de la tarea en SNL (es decir, "*¿Verdadero o falso?*"), una imagen asociada a la CtL, el texto asociado a la CtL en LtL (es decir, "*hi*"), la pregunta a resolver en SNL (es decir, "*Otra manera de decir Hola*"), y un panel que contiene un conjunto de botones cuyas etiquetas son las opciones para responder la pregunta en SNL (es decir, "*Verdadero*" y "*Falso*"). Los estudiantes hacen *clic* en una de estas opciones para elegir la respuesta a la pregunta. Si los estudiantes eligen la respuesta correcta, la aplicación muestra la interfaz de usuario que se muestra en la Fig. 3 (a), y en el caso contrario la que se muestra en la Fig. 3 (b). En ambos casos, navegan a la siguiente actividad de aprendizaje haciendo clic en el botón "*Continuar*".

III. Solución Propuesta

El estudio y análisis de las diferentes metodologías para el aprendizaje de idiomas permite obtener, mediante un proceso de abstracción, los elementos comunes a todas ellas [1][2]. Las metodologías analizadas manejan un conjunto de conceptos de aprendizaje estructurados (meta-modelo de contenidos). Los contenidos tienen una representación gracias a diversos recursos (meta-modelo de medios), y pueden estar agrupados y comportarse de diferentes maneras (meta-modelo

presentación). Finalmente, las metodologías estudiadas definen un conjunto de actividades (meta-modelo actividad) y una secuencia o flujo de control que las relacionan (meta-modelo flujo de control).



(a) (b)

Fig. 3. LAM Opciones de Audio-Texto.

Se presenta ahora la arquitectura MDA y el proceso de las transformaciones necesarias para la generación automática de código:

A. Arquitectura Basada en Modelos Propuesta

Como ya se ha comentado, el desarrollo de aplicaciones basado en modelos, captura la información de la aplicación a desarrollar en uno o más modelos, para generar el código fuente de la aplicación a partir de ellos. Las arquitecturas basadas en modelos promueven la separación de intereses mediante el uso de modelos definidos en diferentes niveles de abstracción.

En el caso del desarrollo basado en modelos de las aplicaciones de aprendizaje de idiomas, la arquitectura basada en el meta-modelo que propuesto en [1][2], se divide en 5 capas (véase la Fig. 4): (1) la capa del *Modelo Independiente de la Computación* (CIM); la capa del *Modelo Independiente de la Plataforma* (PIM) que se divide en 2 subcapas: (2) la capa PIM con la *Lógica del Dominio* (PIM DL) y (3) la capa PIM con la *Interfaz de Usuario* (PIM UI); (4) la capa del *Modelo Específico de la Plataforma* (PSM) y (5) la capa del *Modelo Específico de la Implementación* (ISM).

La capa del **Modelo Independiente de la Computación (CIM)** está definida por un modelo de contenido (*ContentModel*) creado a partir del meta-modelo definido en [1], y se utiliza para representar la estructura de los contenidos de una metodología de aprendizaje de idiomas.

La capa de los **Modelos Independientes de la Plataforma (PIM)** de nuestra arquitectura, tiene 2 subcapas. Por una parte, la subcapa **PIM con la Lógica de Dominio (PIM-DL)** está definida por 2 modelos: un modelo de actividades (*ActivityModel*) y un modelo de flujo de trabajo (*WorkflowModel*). Ambos modelos han sido creados a partir del meta-modelo definido en [1], y se utilizan respectivamente para representar las actividades de una metodología de aprendizaje de idiomas y el flujo de trabajo que lleva de unas actividades a otras. Por otra parte, la subcapa **PIM con la Interfaz de Usuario (PIM-UI)** está definida por otros 2 modelos: un modelo de presentación (*PresentationModel*) y un modelo de recursos (*MediaModel*). Ambos modelos también han sido creados a partir del meta-modelo definido en [1], y se utilizan –respectivamente– para representar la presentación de las actividades de una metodología de aprendizaje de idiomas, y la estructura de recursos multimedia disponibles en la misma.

A continuación, la capa del **Modelo Específico de la Plataforma (PSM)** está definida por un único modelo de etiquetas (*TagMLModel*) creado a partir del meta-modelo definido en [16], que captura –en este caso– todas características de una metodología de aprendizaje de idiomas, en un lenguaje basados en etiquetas.

Finalmente, la capa del **Modelo Específico de la Implementación (ISM)** completa la arquitectura, y es propiamente la capa con el código fuente de una aplicación web de aprendizaje de idiomas –en este caso– en HTML y JavaScript. Como se verá más adelante, para llegar a esta capa desde la capa PSM, se han definido 2 *transformaciones modelo a texto* (M2T): una capaz de generar una representación textual en HTML de los modelos *TagML* (ya definida en [16]), y otra para generar en *JavaScript* la navegación definida en los modelos *Workflow*.

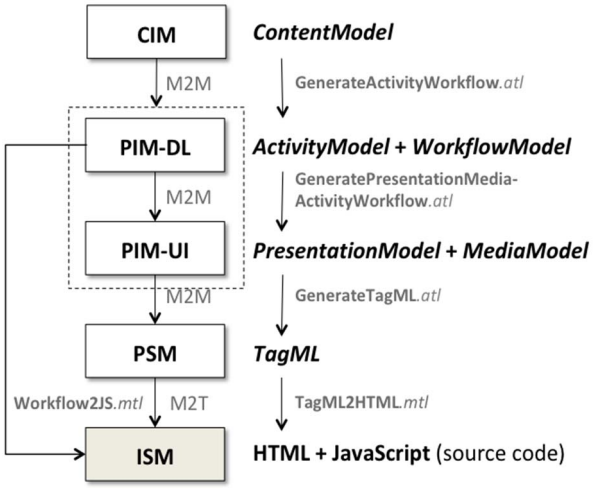


Fig. 4. Capas y transformaciones de la arquitectura basada en modelos de la propuesta.

Luego, se propone el meta-modelo *commons* [1], que es común a todos, y que –básicamente– provee a las demás meta-clases del meta-modelo la capacidad de identificarse y de extenderse.

Finalmente, se propone además un nuevo meta-modelo, que tiene solamente dos meta-clases, y que tiene por objetivo establecer relaciones entre las instancias de dos modelos con la intención de establecer modelos de marcado (por ejemplo, un flujo de trabajo o *workflow* con una *actividad*).

B. Proceso General de las Transformaciones de la Propuesta

El proceso general de la creación de modelos (a partir del meta-modelo propuesto en [1][2]) que se muestra en la Fig. 5, y de las transformaciones que conducirán a la generación del código fuente de una aplicación de aprendizaje de idiomas, puede resumirse en los pasos que se describen a continuación. Los editores gráficos y las transformaciones que se describen en este proceso general, han sido desarrollados hasta el final para poder así validar toda la propuesta.

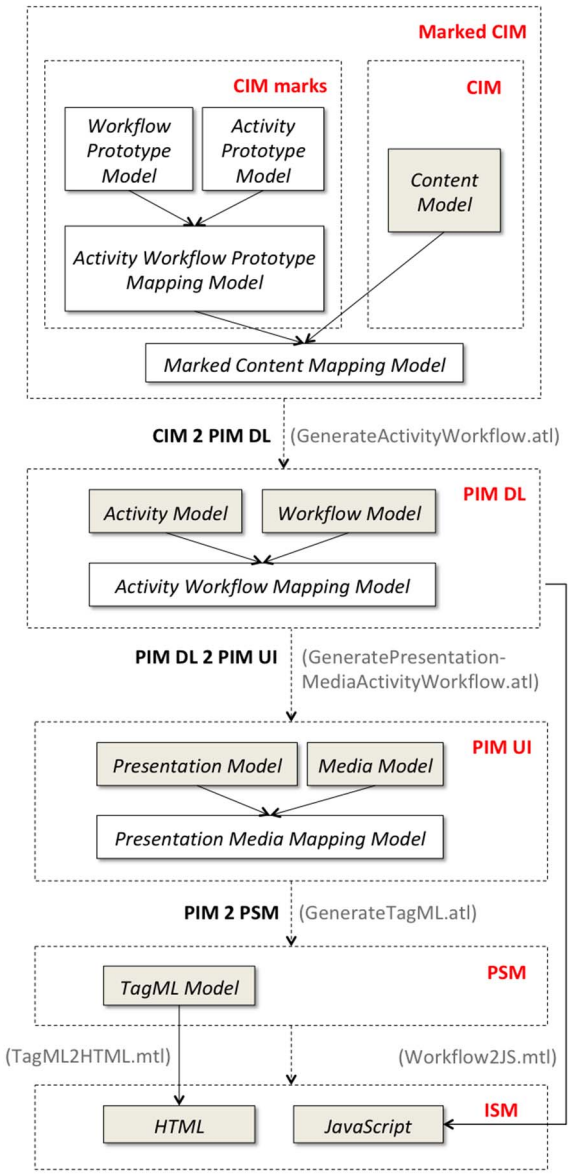


Fig. 5. Proceso de desarrollo basado en modelos de aplicaciones de aprendizaje de idiomas.

En primer lugar, se crea un *ContentModel* que representa la estructura de los contenidos de la metodología de aprendizaje de idiomas que se quiera modelar; por ej. una estructura de contenidos con una secuencia de unidades didácticas conteniendo cada una conjunto de lecciones. Esto constituirá el CIM de la metodología en cuestión que es independiente de su computación.

En segundo lugar, cada lección está compuesta por una secuencia de actividades de aprendizaje que comparten ciertas características modeladas a partir de prototipos. (Ejemplo de estas actividades prototipo, son las LAM *Imagen-Audio-Texto* y la LAM *Opciones de Audio-Texto* descritas en el apartado II, y que se han utilizado para validar esta propuesta). Un prototipo consta de 3 modelos: un modelo de *flujo de trabajo* (que define el comportamiento de la actividad), un modelo de *actividad* (que define la estructura de la actividad), y un modelo de *asociación* (que relaciona un modelo estructura de actividad de aprendizaje con uno de comportamiento).

En tercer lugar, se crea otro modelo de asociación, el *Marked Content Mapping Model*, que representa el CIM marcado. Este modelo relaciona las asociaciones de prototipos de *actividad* y de *flujos de trabajo* (que definen el prototipo de una actividad de aprendizaje) con las lecciones que contienen esas actividades. Así el CIM marcado está compuesto por 5 modelos: el *Content Model*, el *Workflow Prototype Model*, el *Activity Prototype Model*, el *Activity Workflow Prototype Mapping Model*, y el *Marked Content Mapping Model*.

Luego, a partir de estos modelos, se aplica una transformación para construir el PIM-DL de la metodología de aprendizaje a partir del CIM marcado. Esta transformación básicamente clona los prototipos de las actividades de aprendizaje en función del CIM marcado, generando los modelos de actividad (*ActivityModel*), el flujo de trabajo (*WorkflowModel*) y el modelo que los relaciona (*ActivityWorkflowMappingModel*), definiendo el modelo de lógica de dominio de las actividades de aprendizaje de la metodología (PIM-DL).

A continuación, se refinan los modelos generados; por ej. estableciendo el orden de las actividades dentro de las lecciones, caracterizando los tipos de actividades, personalizando los flujos de trabajo, etc.

Una vez refinados, se aplica una segunda transformación que toma el PIM-DL refinado para generar un PIM-UI definido por un *MediaModel*, un *PresentationModel* y un modelo de asociación que relaciona los elementos de interacción con los recursos multimedia del sistema (*PresentationMediaMappingModel*).

Este proceso comienza con una transformación que genera un modelo de presentación (*PresentationModel*) donde por cada *flujo de trabajo asociado* a una actividad se genera una presentación (*presentation*); por ej. una ventana, una página Web, etc. Además, por cada estado del flujo de trabajo, que no este asociado a una actividad, se genera un contenedor (*container*); por ej. un panel, un grupo de opciones, etc., que representa el modelo de interacción de la actividad en cada estado.; por ej. enunciados, opciones, etc. Finalmente, dentro de cada panel se generan los controles a partir del modelo de actividad que representa su estructura (*controls*); por ej. botones, etiquetas, etc. Luego, se genera un modelo de medios (*MediaModel*) donde se genera un contenedor de recursos por cada actividad que contiene los recursos; por ej. videos, texto, imágenes y audios, asociados a la misma. Finalmente, se crea un modelo que asocia los recursos del modelo de medios a los controles del modelo de presentación (*PresentationMediaMappingModel*).

Una vez generado el PIM-UI, se refina para conseguir un PIM marcado con el objetivo de generar el PSM. Por ej. se especifican los lenguajes de implementación (HTML y *JavaScript*), los *frameworks* de desarrollo (*Bootstrap* y *JQuery*), las hojas de estilos, la presentación de los controles y contenedores, etc.

Una vez marcado el PIM-UI, como se generará código HTML, se procede a generar el PSM en TagML [16] utilizando una transformación modelo-a-modelo.

A partir del modelo en TagML se aplica una transformación modelo-a-texto para generar HTML [16] y a partir del modelo de flujo de trabajo (*WorkflowModel*) se genera la implementación en *JavaScript* del comportamiento de la aplicación; por ej. la navegación, el manejo de eventos, etc., llegando al ISM de una aplicación totalmente funcional.

IV. Resultados y Conclusiones

Los resultados de este trabajo son: por una parte la propuesta de una completa arquitectura MDA (desde una capa CIM hasta la capa ISM), y por otra parte el proceso de las transformaciones necesarias, para la generación automática del código fuente (en HTML y *JavaScript*) de aplicaciones de aprendizaje de idiomas.

La validación de la arquitectura MDA propuesta confirma que ésta desacopla perfectamente la funcionalidad de las aplicaciones de aprendizaje de idiomas (gracias a las capas CIM, PIM-DL, PIM-UI, PSM definidas en la misma), permitiendo modelar todo lo necesario para el desarrollo de casi cualquier metodología de aprendizaje de idiomas. Además, aunque no es el principal resultado de este trabajo, se han desarrollado los editores de modelado gráfico pertinentes para permitir y facilitar a los desarrolladores el crear, editar y verificar los modelos de contenidos estructurados, los modelos de recursos media, los modelos de actividades, los modelos de flujo de trabajo y los modelos de presentación, necesarios para modelar este tipo de aplicaciones.

Para el proceso de transformaciones, se han desarrollado en primer lugar las 3 transformaciones modelo a modelo necesarias enumeradas en la Tabla 1. Estas transformaciones tienen 34 *matched rules*, 37 *lazy rules* y 67 *helpers*. A modo de ejemplo, en la Fig. 6 se ilustra parte de la generación del PSM en TagML utilizando la regla *Presentation_Rule*, que es una de las *matched rules* de la transformación modelo-a-modelo denominada *GenerateTagML,* que tiene como parte de su entrada un modelo de presentación (*PresentationModel*).

TABLA I
Transformaciones Modelo a Modelo

Capas	Transformación
CIM 2 PIM-DL	GenerateActivityWorkflow.atl
PIM 2 PIM-UI	GeneratePresentationMediaActivityWorkflow.atl
PIM 2 PSM	GenerateTagML.atl

Así mismo, para generar los códigos HTML a partir de los modelos TagML pertinentes, y para generar las implementaciones en *JavaScript* de los comportamientos de este tipo de aplicaciones a partir de los modelos de flujo de trabajo (*WorkflowModel*) que se definan en cada caso, se han desarrollado las 2 transformaciones modelo a texto enumeradas en la Tabla 2.

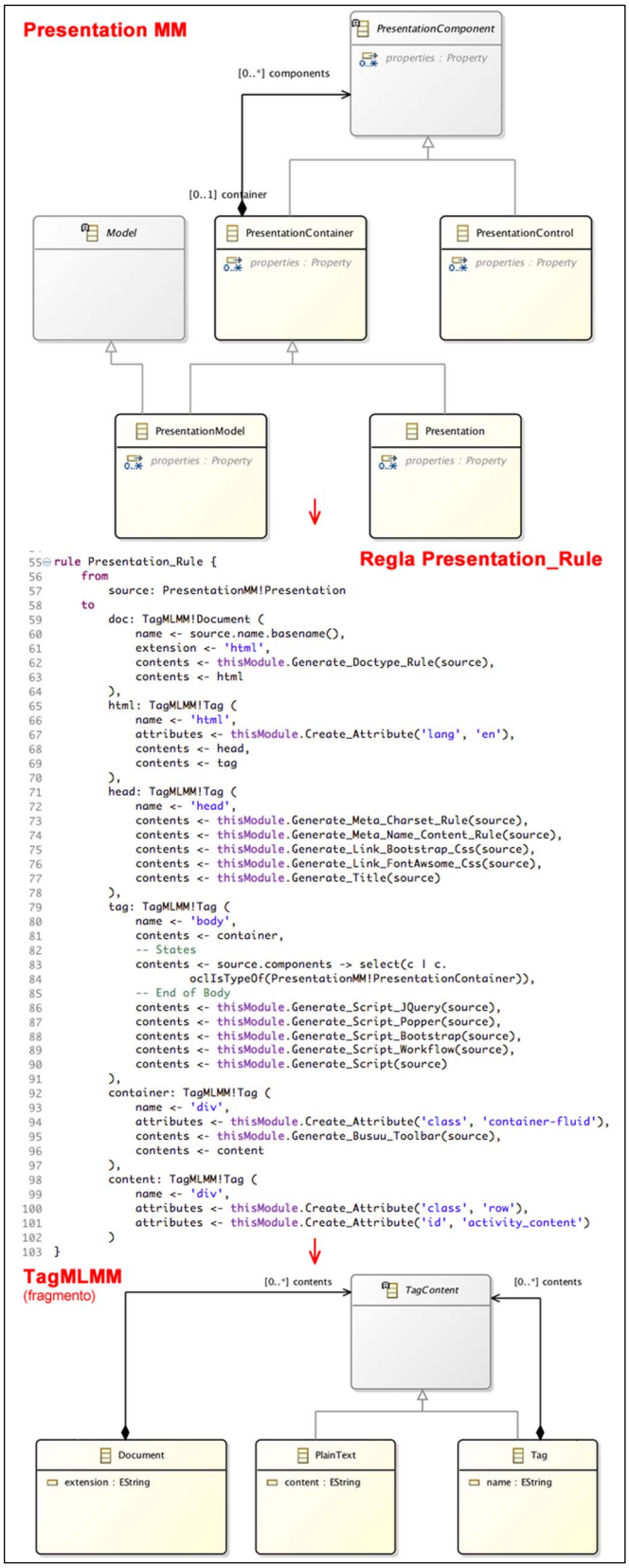


Fig. 6. Ilustración de parte de la generación del PSM en TagML (regla Presentation_Rule contenida en GenerateTagML.atl).

TABLA II
TRANSFORMACIONES MODELO A TEXTO

Capas	Transformación
PSM 2 HTML	TagML2HTML.mtl
PSM 2 JS	Workflow2JS.mtl

Para llevar a cabo las transformaciones modelo a modelo y modelo a texto, se han utilizando los lenguajes de transformación ATL [8] y ACCELEO [9] respectivamente.

Para validar la propuesta, se ha llevado hasta el final el modelado y la completa generación automática del código fuente de dos tipos de actividades utilizadas de forma muy parecida en metodologías como *Duolingo* [3] y *Busuu* [5]: las LAM *Imagen-Audio-Texto* y la LAM *Opciones de Audio-Texto*, descritas en el apartado II.

El trabajo futuro de este trabajo, incluye el desarrollo más avanzado de los editores de modelado gráfico que se utilizan para crear, editar y verificar los modelos de metodologías de aprendizaje generados con los meta-modelos que configuran la arquitectura MDA propuesta. Se está trabajando también en validar la arquitectura MDA y el proceso de transformaciones propuesto, con nuevas LAM utilizadas de forma semejante en varias metodologías como *Busuu*, *Babbel* y *Duolingo*. Finalmente, se está trabajando en nuevas extensiones de los meta-modelos propuestos, que permiten mejorar la validación de los modelos mediante la definición de expresiones OCL personalizadas que se cargan dinámicamente, utilizando el *plugin Complete OCL* [17]. Estas extensiones se definen en función de la sub-clasificación de las meta-clases, que permiten introducir nuevas características de una manera flexible y robusta.

REFERENCIAS

[1] G. Sebastián, R. Tesoriero, and J. A. Gallud, “Modeling language-learning applications”, IEEE Latin America Transactions, vol. 15(9), pp. 1771-1776, 2017.

[2] G. Sebastián, R. Tesoriero, and J. A. Gallud, “Model-based approach to develop learning exercises in language-learning applications”, IET Software, vol. 18(3), pp. 206-2014, 2018.

[3] DUOLINGO. [Online]. Available: https://es.duolingo.com/

[4] BABBEL. [Online]. Available: https://es.babbel.com/

[5] BUSUU. [Online]. Available: https://www.busuu.com/es/

[6] Eclipse Foundation, “Eclipse modeling framework (EMF),” Eclipse Foundation, 2019, [Online]. Available: http://www.eclipse.org/modeling/emf/.

[7] Eclipse Foundation 2, “The oclinecore language,” Eclipse Foundation, 2013, https://wiki.eclipse.org/OCL/OCLinEcore.

[8] Eclipse Foundation 3, “ATL: a model transformation technology,” Eclipse Foundation, 2019, https://eclipse.org/atl/.

[9] Obeo, “Acceleo,” Obeo, 2019, https://www.eclipse.org/acceleo/.

[10] A. García-Holgado and F. J. García-Peñalvo, “Validation of the learning ecosystem metamodel using transformation rules”, Future Generation Computer Systems, vol. 91, pp. 300-310, 2019.

[11] Y. Rhazali, Y. Hadi, and A. Mouloudi, “Model Transformation with ATL into MDA from CIM to PIM Structured through MVC”, Procedia Computer Science, vol. 83, pp. 1096-1101, 2016.

[12] A. P. Fontes Magalhaes, A. M. Santos Andrade, and R. S. Pitangueira Maciel, “Model driven transformation development (MDTD): An approach for developing model to model transformation”, Information and Software Technology, vol. 114, pp. 55-76, 2019.

[13] K. Arrhioui, S. Mbarki, O. Betari, S. Roubi, and M. Erramdani, "A Model Driven Approach for Modeling and Generating PHP CodeIgniter based Applications", 1st International Conference on Affective computing, Machine Learning and Intelligent Systems, Transactions on Machine Learning and Artificial Intelligence, vol. 5, no. 4, 2017.

[14] H. Benouda, M. Azizi, M. Moussaoui and R. Esbai, "Automatic code generation within MDA approach for cross-platform mobiles apps", First International Conference on Embedded & Distributed Systems (EDiS), pp. 1-5, 2017.

[15] R. Esbai, M. Erramdani, S. Mbarki, I. Arrassen, A. Meziane, and M. Moussaoui, "Transformation by Modeling MOF 2.0 QVT: From UML to MVC2 Web model", INFOCOMP Journal of Computer Science, vol. 10, no. 3, 2011.

[16] H. M. Fardoun, R. Tesoriero, G. Sebastián, and N. Safa, “A Simplified MbUID Process to Generate Web Form-based UIs”, in *Proc. ICSOFT*, pp. 835-842, 2018.

[17] Complete OCL. [Online]. Available: https://marketplace.eclipse.org/content/eclipse-ocl

Gabriel Sebastián received his BSc degree from Polytechnic University of Valencia (UPV), and his MSc degree from University of Castilla-La Mancha (UCLM) – all the two degrees in Computer Science. His main research interests are Multimedia, Human-Computer Interaction and Software Engineering. He was involved in the development of many projects related to Distributed User Interfaces and Model-driven Development of User Interfaces focused on the Web as the deployment platform. He published more than 20 research articles and book chapters in Journals and International Congresses. Currently, he works as Project Manager in the Interactive Systems Everywhere research group. He is a PhD student of the Computing System Department (Faculty of Computing Science Engineering) in UCLM, and researcher in the Albacete Research Institute of Informatics (I3A) in Albacete, Spain.

Ricardo Tesoriero received his PhD and MSc degrees from the University of Castilla-La Mancha (UCLM), Spain and a BSc degree from National University of La Plata (UNLP), Argentina – all the three degrees in Computer Science. His main research interests are Model-driven development of User Interfaces focused on the Web as deployment platform and Human-Computer Interaction on ubiquitous computing environments. He published more than 70 research articles and book chapters in Journals and International Congresses. He performed a post-doctoral stay in the Universite Catholoque de Louvain (UCL) in Louvain-La Neuve, Belgium where he performed research activities on Model-driven development of User Interfaces. He was committee member in several scientific conferences and workshops, including DUI (Distributed User Interfaces), INTERACCCION, ISEC, IADIS/WWW (La Web), etc. He has been professor in the Computing System Department at the Faculty of Computer Science Engineering of the UCLM teaching the Advanced Methods of Software Development and Human-Computer Interaction subjects since 2008.

Jose A. Gallud received his PhD degree from the University of Murcia and the MSc and BSc degrees from Polytechnic University of Valencia – all the three degrees in Computer Science. Her main research of interest focuses on Human-Computer Interaction, development of Interactive Systems and Distributed User Interfaces. Dr. Jose A. Gallud has published widely in these areas. He has been Guest Editor for several international journals, such as JSS (The International Journal of Software and Systems), IJHCS (International Journal of Human Computer Studies), JUCS (Journal of Universal Computer Sciences). He has some Books and Chapters in the field of Human-Computer Interaction. He is member of different national and international societies (ACM and AIPO). Currently, he works as a professor at the University of Castilla-La Mancha.