Arquitectura Dirigida por Modelos Aplicada al Desarrollo de Metodologías de Aprendizaje de Idiomas

Gabriel Sebastián Rivera

Dirigida por Dr. Ricardo Tesoriero y Dr. José A. Gallud Lázaro

Departamento de Sistemas Informáticos

Universidad de Castilla-La Mancha

Esta disertación se presenta para el grado de

Doctor en Informática

Marzo 2020

Quiero dedicar este trabajo especialmente a mi padre, Víctor, doctor en Derecho Canónico y doctor en Historia de la Farmacia, pues –como gran profesor universitario que fue– le habría hecho gran ilusión el verlo finalizado.

Declaración

Este trabajo es el resultado de mi propio trabajo y no incluye ningún resultado de trabajos hechos en colaboración, excepto donde ha sido específicamente indicado en el texto. Este trabajo no ha sido previamente enviado, ni parcial ni totalmente, a ninguna universidad o institución de ningún grado u otra titulación. Además, por la presente declaro ser uno de los principales autores de los trabajos utilizados en esta tesis por compendio de publicaciones, incluyendo los siguientes trabajos que han sido publicados en revistas con índice de impacto:

- Gabriel Sebastián, Ricardo Tesoriero, José A. Gallud, "**Modeling Language-Learning Applications**", IEEE Latin America Transactions, doi: 10.1109/TLA.2017.8015084, 15(9): 1771-1776, 2017 (IF: 0.804, Q4)

- Gabriel Sebastián, Ricardo Tesoriero, José A. Gallud, "**Model-based approach to develop learning exercises in language-learning applications**", IET Software, doi: 10.1049/iet-sen.2017.0085, 12(3): 206-214, 2018 (IF: 0.695, Q4)

- Gabriel Sebastián, Ricardo Tesoriero, José A. Gallud, "**Automatic Code Generation for Language-Learning Applications**", IEEE Latin America Transactions, 2019 (IF: 0.804, Q4)

- Gabriel Sebastián, José A. Gallud, Ricardo Tesoriero, "**Code Generation Using Model Driven Architecture: A Systematic Mapping Study**", Journal of Computer Languages, doi: 10.1016/j.cola.2019.100935, 2019 (IF: 1.714, Q2)[1]

Gabriel Sebastián Rivera

Firmado digitalmente por Gabriel Sebastián Rivera
Fecha: 2020.03.13 13:21:38 +01'00'

Gabriel Sebastián Rivera
Marzo 2020

[1]Este artículo se envió originalmente a *Computer Languages, Systems & Structures* (IF: 1.714, Q2), sin embargo, durante el proceso de revisión, la revista experimentó un cambio de nombre y ahora se titula *Journal of Computer Languages*

Agradecimientos

En primer lugar, me gustaría agradecer a mis directores, Ricardo y José Antonio, su constante ayuda y desvelo, que me ha permitido seguir siempre adelante en este trabajo de investigación. Gracias Ricardo por tu paciencia y disponibilidad en todos los momentos en los que hemos tenido que enfrentarnos a los distintos escollos que han ido apareciendo en el camino de esta investigación. Gracias José Antonio por tu optimismo y visión positiva a lo largo de estos años de trabajo.

Quiero agradecer también a todos los miembros del Grupo ISE (Interactive Systems Engineering) su apoyo, ánimo y amistad durante estos años.

Finalmente, quiero destacar que esta tesis ha sido parcialmente financiada por el proyecto nacional –concedido por el Ministerio de Ciencia, Innovación y Universidades– con referencia RTI2018-099942-B-I00, y por el proyecto TecnoCRA (ref: SBPLY/17/180501/000495) concedido por la Junta de Comunidades de Castilla-La Mancha (JCCM) y el Fondo Europeo de Desarrollo Regional (FEDER).

Resumen

Abstract

Today, more and more people are interested in learning a second –and even a third– foreign language. This is due to the phenomenon of globalization, and is being facilitated by the extensive use of the Internet.

The process of learning a foreign language is defined by methodologies and increasingly supported by technology. The development of such applications is complex (for the variety of execution environments, and for having a lot of content often difficult to validate), so this thesis proposes a model-driven approach to software development to support language teaching.

The study and analysis of different methodologies for language learning has allowed to obtain, through an abstraction process, the elements common to all of them: First (1) a set/hierarchy of concepts that are taught (contents), and (2) a set of resources (means) to present the concepts. On the other hand, all methodologies define (3) different types of learning exercises (activities) and (4) a sequence or set of sequences that relate them (control/work flow). Finally, they define a series of (5) templates or layouts for viewing content and activities (presentation).

This thesis presents a Model-Driven Architecture (MDA) proposal that support the development of language learning applications from models. From a Computational Independent Model (CIM) layer to the Implementation Specific Model (ISM) layer, meta-models and graphical editors are presented for the various domain-specific languages, which allow you to model everything you need for the development of almost any language learning methodology. On the other hand, this thesis also defines the necessary transformations, and the associated transformation process, for the automatic generation of source code (in HTML and JavaScript) of language learning applications.

The proposal has been validated through the modeling and generation of source code of the most important functionality elements of the Lexiway methodology, as well as various types of learning activities present in methodologies such as Duolingo and Busuu.

Resumen

Hoy en día, cada vez más gente está interesada en el aprendizaje de una segunda –e incluso de una tercera– lengua extranjera. Esto es debido al fenómeno de la globalización, y está siendo facilitado por el uso extensivo de Internet.

El proceso de aprendizaje de una lengua extranjera está definido por metodologías y –cada vez más– apoyado por la tecnología. El desarrollo de este tipo de aplicaciones es complejo (por la variedad de los entornos de ejecución, y por tener una gran cantidad de contenidos con frecuencia difíciles de validar), por lo que esta tesis propone un enfoque dirigido por modelos para desarrollo de software para dar soporte a la enseñanza de idiomas.

El estudio y análisis de diferentes metodologías para el aprendizaje de idiomas ha permitido obtener, mediante un proceso de abstracción, los elementos comunes a todas ellas: En primer lugar (1) un conjunto/jerarquía de conceptos que se enseñan (contenidos), y (2) un conjunto de recursos (medios) para presentar los conceptos. Por otra parte, todas las metodologías definen (3) diversos tipos de ejercicios de aprendizaje (actividades) y (4) una secuencia o conjunto de secuencias que las relacionan (flujo de control/trabajo). Finalmente, definen una serie de (5) plantillas o disposiciones para la visualización de los contenidos y de las actividades (presentación).

Esta tesis presenta una solución *Model-Driven Architecture* (MDA) que permite el desarrollo de aplicaciones para el aprendizaje de idiomas a partir de modelos. Así pues, desde una capa *Computational Independent Model* (CIM) hasta la capa *Implementation Specific Model* (ISM), se presentan los meta-modelos y los editores gráficos para los distintos lenguajes específicos de dominio, que permiten modelar todo lo necesario para el desarrollo de casi cualquier metodología de aprendizaje de idiomas. Por otra parte, en esta tesis también se definen las transformaciones necesarias, y el proceso de transformación asociado, para la generación automática del código fuente (en HTML y JavaScript) de aplicaciones de aprendizaje de idiomas.

La propuesta ha sido validada a través del modelado y la generación del código fuente de los elementos de funcionalidad más importantes de la metodología Lexiway, así como de varios tipos de actividades de aprendizaje muy utilizados –y de forma muy parecida– en metodologías como Duolingo y Busuu.

Tabla de Contenidos

Lista de Figuras

Lista de Tablas

Nomenclatura

Acrónimos / Abreviaturas

CIM Modelos Independientes de la Computación (del inglés: Computational Independent Model)

CWM Common Warehouse Metamodel

DSL Lenguaje Específico de Dominio (del inglés: Domain Specific Language)

GMF Eclipse Graphical Framework

GMP Eclipse Modeling Project

GUI Interfaz Gráfica de Usuario (del inglés: Graphical User Interface)

IoT Internet of Things

ISM Modelos Específicos de la Implementación (del inglés: Implementation Specific Model)

LAM Mecanismo de Actividades de Aprendizaje (del inglés: Learning Activity Mechanisms)

$MB-UID$ Model-Based UI Development

MDA Arquitectura Dirigida por Modelos (del inglés: Model-Driven Architecture)

MDD Desarrollo Dirigido por Modelos (del inglés: Model-Driven Development)

MOF Meta Object Facility

OCL Object Constraint Language

OMG Object Management Group

PIM Modelos Independientes de Plataforma (del inglés: Platform Independent Model)

$PIM - DL$ PIM con la Lógica de Dominio

$PIM - UI$ PIM con la Interfaz de Usuario

PSM Modelos Específicos de Plataforma (del inglés: Platform Specific Model)

UML Unified Modeling Language

XMI XML Metadata Interchange

Capítulo 1

Introducción

En este capítulo se presentan los diferentes temas abordados en esta tesis. La sección 1.1 presenta la justificación del problema abordado en este trabajo, así como la motivación. La sección 1.2 presenta una breve descripción del estado del arte, así como las cuestiones de investigación propuestas en esta tesis. La sección 1.3 presenta los objetivos –objetivo principal y objetivos específicos– planteados y alcanzados en esta tesis. Finalmente, la sección 1.4 presenta la estructura de esta tesis.

1.1 Justificación y motivación

El interés por estudiar un idioma extranjero está aumentando en los últimos tiempos debido al fenómeno de la globalización y al uso generalizado de Internet como plataforma para la proyección tanto personal como de las empresas.

No hay más que ver cómo crece el número de estudiantes en escuelas y academias de enseñanza de idiomas, o el aumento de la oferta bilingüe en centros educativos a todos los niveles (primaria, secundaria y universidad). También es interesante señalar la numerosa oferta de diferentes métodos para aprender idiomas.

La tecnología no ha sido ajena a este interés de la gente por el aprendizaje de idiomas, y se constata que son numerosas las aplicaciones y herramientas relacionadas con este campo, que se ofrecen al usuario. Con el rápido crecimiento del uso de *smartphones*, las aplicaciones móviles se están convirtiendo, junto a la Web, en uno de los ámbitos preferidos por los usuarios para el aprendizaje de un idioma extranjero. El desarrollo de este tipo de aplicaciones es complejo, por la diversidad de metodologías para el aprendizaje de idiomas que existen, por la diversidad de plataformas de ejecución posibles (Web, móvil y escritorio) y por la diversidad de tecnologías que se pueden utilizar para desarrollarlas. Por ello, resulta muy oportuno aplicar la solución que proporciona el Desarrollo Dirigido por Modelos (*MDD*)

y la Arquitectura Dirigida por Modelos (*MDA*), por el ahorro de costes que pueden aportar en el desarrollo de aplicaciones complejas multi-plataforma.

Se ha elegido el campo de los aplicativos de enseñanza de idiomas, debido a la trayectoria profesional y de investigación del autor de esta tesis en estos dos ámbitos:

1. La generación de código a partir de modelos creados con editores visuales; así como
2. El desarrollo de aplicativos (multimedia interactivos) para la enseñanza de idiomas.

En relación a la experiencia en la generación de código a partir de modelos creados con editores visuales, podemos destacar que:

- En 1996/97 desarrolló, defendió y publicó su Proyecto Fin de Carrera "Sistema traductor de código HTML para la creación y diseño de páginas web de instituciones" [19], que fue un innovador editor gráfico y WYSIWYG (*What You See Is What You Get*) de páginas web (desarrollado con Visual Basic), que a partir de modelos creados con dicho editor gráfico, permitía la generación automática de código HTML de páginas web.

- En los años que van de 2009 a 2012 participó –como miembro del Grupo ISE– en el proyecto de investigación liderado por Telefónica I+d "Tecnologías para prestar servicios en movilidad en el futuro universo inteligente", financiado por el CDTI (CENIT-2008-2010, programa INGENIO 2010). En particular, desarrolló y publicó diversos prototipos de creación de servicios a partir de modelos generados con editores y en movilidad [20][25]. Estos trabajos los defendió en su tesis de master con título: "Diseño de Interfaces de Creación de Servicios en Movilidad".

En relación a la experiencia en el desarrollo de aplicativos de enseñanza de idiomas, podemos destacar que:

- En el curso académico 1998-1999, con el *Museo Internacional de Electrografía* (MIDE) y el *Grupo de Redes Neuronales* de la Universidad de Castilla-La Mancha, participó en el proyecto I+D+i "Creación de una aplicación en multimedia interactivo para una maqueta de un diccionario de Richmond con Orientación On Line", financiado por la empresa *Richmond Publishing* (del *Grupo Santillana de Ediciones* S.A.), desarrollando la aplicación informática de un revolucionario diccionario de inglés que generaba una novedosa experiencia global del lenguaje que sustituía al tradicional diccionario.

- Desde 2000 hasta 2006, en la empresa Ideando Experiencias S.L. desarrolló para las empresas *Richmond Publishing* y *Santillana francés*, diversos aplicativos multimedia interactivos de aprendizaje de inglés y francés: "Richmond Picture Dictionary", "Galaxy français", "Wizard english", entre otros.

- Desde 2013 hasta 2018, ha desarrollado, mantenido y actualizado los aplicativos y el aula virtual que dan soporte a la metodología de aprendizaje de inglés "Lexiway". Como se verá más adelante en esta memoria, la experiencia en estos desarrollos relativos a una metodología –real y en uso comercial– de enseñanza del inglés, ha facilitado todo lo necesario para empezar a construir el meta-modelo propuesto en esta tesis, así como para validar los resultados de esta tesis con casos de estudios reales y llevados hasta el final (es decir, hasta la generación automática de código) [22][23].

1.2 Estado del arte y cuestiones de investigación

Esta sección presenta una visión general del contexto de la investigación, dividido en dos puntos de vista importantes: en primer lugar, los conceptos fundamentales relacionados con los métodos para aprender idiomas (características generales y algunos ejemplos de métodos); y en segundo lugar, algunos trabajos relacionados con un enfoque de desarrollo dirigido por modelos. En esta sección se presentan también las cuestiones de investigación propuestas.

1.2.1 Conceptos fundamentales relacionados con los métodos de aprendizaje de idiomas

El aprendizaje de idiomas es un ámbito en el que proliferan prometedores métodos, técnicas y herramientas. El presente estudio, se ha centrado en aquellas metodologías que ofrecen alguna clase de soporte tecnológico (sitio Web, app móvil o similar).

Entre las metodologías que han sido objeto de estudio se encuentran Lexiway[1], Busuu[2], Duolingo[3] y Babbel[4]. Lexiway es una metodología novedosa que pone un mayor énfasis en la pronunciación, en el vocabulario y en la composición de frases (y no tanto en la gramática). Lexiway cuenta con un buen apoyo de herramientas: un aula virtual y una aplicación iOS. Duolingo es un método para aprender inglés que ha surgido con la profusión de aplicaciones móviles. Este método da una mayor importancia a la motivación de los usuarios, para lo cual recurre a la gamificación [26]. Babbel es una metodología que da mucha importancia a que el alumno amplíe su vocabulario; con cada ejercicio, detecta sus puntos fuertes y débiles, y va personalizando cada lección. Babbel tiene la virtud de conseguir que la gramática

[1]Lexiway (El método Lexiway). URL = http://academialexiway.com/academia-ingles/metodo-academia-ingles.html (último acceso 28/01/20)

[2]Busuu (página principal). URL = https://www.busuu.com/ (último acceso 28/01/20)

[3]Duolingo (página principal). URL = https://www.duolingo.com/ (lúltimo acceso 28/01/20)

[4]Babbel (página principal). URL = https://www.babbel.com/ (último acceso 28/01/20)

se aprenda intuitivamente, casi sin que el alumno se de cuenta. Busuu es un método que mejora la comprensión lectora, el diálogo y la escritura de los alumnos con una gran variedad de ejercicios; en algunos aspectos se parece a Duolingo. El entorno de aprendizaje de Busuu, permite además a sus alumnos interactuar con su propia comunidad internacional de hablantes nativos.

Los enfoques que ofrecen las distintas metodologías para aprender idiomas son muy diversos, aunque es posible identificar elementos comunes en todas ellas: un conjunto de contenidos que se enseñan, un conjunto de recursos multimedia para presentar dichos contenidos, diversos tipos de ejercicios de aprendizaje, etc. El enfoque del desarrollo basado en modelos se encargará de abstraer los elementos presentes en los diferentes métodos.

1.2.2 Trabajos relacionados con un enfoque de desarrollo dirigido por modelos

El concepto de *MDA* fue propuesto por el Object Management Group (*OMG*) en 2001. En MDA Guide[5], *MDA* se define como una solución para utilizar modelos en el desarrollo software. Esta solución proporciona un mayor protagonismo a los modelos en los sistemas que se van a desarrollar. Como el propio nombre indica, se dice dirigido por modelos porque cubre las etapas de concepción, diseño, construcción, desarrollo, mantenimiento y modificación. En el núcleo de *MDA* hay una serie de estándares importantes de *OMG*: Unified Modeling Language (*UML*), la Meta Object Facility (*MOF*), XML Metadata Interchange (*XMI*) y la Common Warehouse Metamodel (*CWM*). Estos estándares definen la infraestructura central de MDA, que se ha utilizado con éxito en el modelado y desarrollo de muchos sistemas modernos.

En el ámbito de la Interacción Persona-Ordenador se puede encontrar un enfoque que utiliza modelos para obtener la interfaz de usuario (Desarrollo Basado en Modelos de Interfaces de Usuario $MB-UID$) [18]. En este sentido, como en este trabajo de investigación se propone el desarrollo de sistemas interactivos, éste comparte objetivos y enfoque con aquellos que utilizan $MB-UID$.

Son numerosos los trabajos relacionados que utilizan un enfoque *MDD* o *MDA* en el desarrollo de aplicaciones Web [14][5]. En [15] se presenta un ejemplo de aplicación de una MDA para desarrollar una aplicación para el aprendizaje de música. Una metodología para el modelado de aplicaciones de e-Learning se puede ver en [7][8]. Se trata de una metodología generalista para el desarrollo de aplicaciones de e-learning, y por lo tanto no está enfocada al desarrollo de aplicaciones para el aprendizaje de idiomas.

[5]Object Management Group. Model driven Architectures. URL = http://www.omg.org/mda/

En [1] se recogen las experiencias de diferentes grupos de investigación internacionales ilustrando el estado del arte del diseño del aprendizaje de idiomas usando tecnología móvil en educación formal y no formal. Otros trabajos de metodologías de desarrollo de aplicaciones de e-Learning se pueden ver en [10][11][17][16].

Muy pocos dudan del protagonismo que tendrá el *MDD* en el futuro cercano del desarrollo software, especialmente en la medida en que los desarrolladores vayan contando con herramientas cada vez más sofisticadas [6][2]. Quizá por eso no se ha encontrado en la bibliografía trabajos previos que formalicen mediante un meta-modelo una metodología para el aprendizaje de idiomas.

1.2.3 Cuestiones de investigación propuestas

Después de realizar una rigurosa revisión del estado del arte, se constata que no hay apenas bibliografía que aborde el desarrollo de aplicaciones de enseñanza de idiomas con un enfoque *MDA*.

Teniendo en cuenta las deficiencias y necesidades apreciadas, se han establecido las siguientes Cuestiones de Investigación (CIs) para esta tesis:

- **CI1**: ¿El enfoque MDA sigue siendo un área de interés en la investigación en ingeniería del software? ¿Cuáles son los principales campos donde se aplica con éxito el enfoque MDA? ¿Cuántos artículos que utilizan MDA incluyen generación de código? ¿Existen trabajos relacionados con el desarrollo de aplicaciones dirigido por modelos para el aprendizaje de idiomas?

- **CI2**: ¿Cuáles son las principales características de las aplicaciones de aprendizaje de idiomas? ¿Es posible definir una arquitectura dirigida por modelos para el desarrollo de este tipo aplicaciones, que incluyan dichas características?

- **CI3**: ¿Es posible el desarrollo de herramientas capaces de manipular los modelos definidos por la arquitectura alcanzada en los resultados de la CI2? ¿Se pueden proponer notaciones gráficas que mejoren la experiencia del usuario, en relación a la tradicional notación del árbol reflexivo, usada por muchos *frameworks* de desarrollo basado en modelos?

- **CI4**: ¿Se pueden desarrollar las transformaciones *modelo-a-modelo* y *modelo-a-texto* necesarias para generar el código fuente final de este tipo de aplicaciones?

- **CI5**: ¿Se puede –mediante el desarrollo de aplicaciones dirigido por modelos– modelar y generar el código fuente de aplicaciones como Lexiway o Duolingo? ¿Se puede

validar el meta-modelo propuesto y las transformaciones desarrolladas, mediante casos de estudio de actividades reales de metodologías como Busuu o Duolingo?

1.3 Objetivos

El **objetivo principal** de esta tesis doctoral es la definición de una arquitectura dirigida por modelos para desarrollar aplicaciones de aprendizaje de idiomas, y los **objetivos específicos** de este trabajo que se derivan de dicho objetivo son los siguientes:

- Realización de una revisión exhaustiva de las características principales de las aplicaciones utilizadas para el aprendizaje de idiomas, así como de los trabajos relacionados con el desarrollo de éstas con un enfoque dirigido por modelos.

- Definir una arquitectura dirigida por modelos para el desarrollo de aplicaciones de aprendizaje de idiomas.

- Desarrollar un conjunto de lenguajes específicos de dominio y los editores asociados que los implementen, para crear y manipular modelos definidos por los metamodelos contenidos en dicha arquitectura.

- Desarrollar un conjunto de transformaciones (modelo-a-modelo y modelo-a-texto) conjuntamente con el proceso de transformación necesario, para generar los modelos y el código fuente de aplicaciones para el aprendizaje de idiomas.

- Aplicar la arquitectura definida para desarrollar casos de estudio que la validen adecuadamente.

1.4 Estructura de la tesis

En el Capítulo 1 se ha presentado: en la Sección 1.1 la justificación y motivación de esta tesis; en la Sección 1.2 el estado del arte y las cuestiones de investigación propuestas; y en la Sección 1.3 el objetivo principal y los objetivos específicos de este trabajo. A continuación, el Capítulo 2 muestra los resultados obtenidos con respecto a cada pregunta de investigación en las Secciones 2.1, 2.2, 2.3, 2.4 y 2.5, así como un resumen de las publicaciones que forman parte de esta tesis por compendio de publicaciones, en la Sección 2.6.

Después, del Capítulo 3 al Capítulo 9 se presentan en detalle los trabajos más importantes con su contenido completo; tanto los artículos que ya han sido publicados o aceptados, como aquellos otros enviados que están en proceso de revisión. Finalmente, el Capítulo 10

concluye este trabajo con la enumeración de las principales conclusiones, los trabajos en marcha, así como de los trabajos futuros derivados de éstas.

Capítulo 2

Resultados

En este capítulo, se presentan los resultados obtenidos con respecto a cada pregunta de investigación descrita anteriormente, en las Secciones 2.1, 2.2, 2.3, 2.4 y 2.5 respectivamente. Finalmente, en la Sección 2.6 se muestra un resumen de todos los artículos publicados o enviados.

Téngase en cuenta que las cuatro publicaciones que respaldan esta tesis [22][23][21][24], junto con los otros artículos relevantes presentados en detalle desde el Capítulo 3 hasta el Capítulo 9, están relacionadas con las cinco preguntas de investigación definidas anteriormente en la Sección 1.2. Con todo, **los resultados obtenidos son**:

1. **Revisión de las características principales de las aplicaciones más utilizadas por las metodologías de aprendizaje de idiomas**. Entre las aplicaciones más popularmente utilizadas encontramos: Lexiway, Busuu, Duolingo y Babbel, entre otras. El estudio ha permitido obtener, mediante un proceso de abstracción, los elementos comunes a todas ellas. Entre los factores más relevantes podemos encontrar: (1) un conjunto/jerarquía de conceptos/contenidos que se enseñan (contenidos); (2) un conjunto de recursos multimedia (medios) para presentar los conceptos de aprendizaje o las actividades; (3) los diversos tipos de ejercicios de aprendizaje (actividades); (4) una secuencia –o conjunto de secuencias– que las relacionan (flujo de control); y (5) un conjunto de plantillas para la visualización de los contenidos y de las actividades (presentación).

2. **Revisión de los trabajos relacionados con el desarrollo de este tipo de aplicaciones con un enfoque dirigido por modelos**. Se ha realizado un estado del arte (trabajos relacionados) para cada uno de los aspectos del enfoque MDA para este tipo de aplicaciones, en lo relativo a: meta-modelos, editores de modelos y generación de código fuente.

3. **Estudio sistemático de la generación de código utilizando el enfoque de arquitecturas dirigidas por modelos**. Este estudio sistemático (*Systematic Mapping Study*), proporciona una comprensión estructurada del estado del arte –dentro de la ingeniería del software– de la investigación de la generación de código utilizando el enfoque MDA, mediante la identificación de 50 estudios primarios a partir de 2.145 artículos relacionados con MDA publicados durante un periodo de 10 años (2008-2018).

4. **Definición de una arquitectura dirigida por modelos para desarrollar aplicaciones de aprendizaje de idiomas**. Se ha desarrollado una propuesta de una completa arquitectura MDA –desde una capa *Computational Independent Model* (CIM), hasta la capa *Implementation Specific Model* (ISM)– que desacopla perfectamente los diferentes aspectos de este tipo de aplicaciones.

5. **Evaluación de usuarios para analizar la calidad en uso** (la efectividad y eficiencia del proceso de desarrollo), **así como la calidad del producto** (corrección, robustez, extensibilidad y reutilización del producto) de la propuesta. Los resultados de dicha evaluación, concluyen que el código generado usando el enfoque propuesto supera –en ciertas métricas seleccionadas– el código generado usando un enfoque tradicional.

6. **Desarrollo de los editores de modelos necesarios para desarrollar aplicaciones de aprendizaje de idiomas siguiendo la arquitectura definida**. Se ha documentado la consecuente notación gráfica para un Lenguaje Específico de Dominio (DSL) que permite a los desarrolladores representar todas las características de las metodologías de aprendizaje de idiomas. Además, se ha evaluado esta propuesta con un *framework* que utiliza las dimensiones cognitivas de las notaciones para sistemas de notación.

7. **Desarrollo de las transformaciones modelo-a-modelo y modelo-a-texto necesarias para desarrollar aplicaciones de aprendizaje de idiomas siguiendo la arquitectura definida**. El adecuado proceso de dichas transformaciones conduce a la generación automática del código fuente (en HTML y JavaScript) de aplicaciones de aprendizaje de idiomas. 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 y Busuu.

8. **Desarrollo de casos de estudio siguiendo las arquitectura definida**. Se ha llevado hasta el final el modelado y la completa generación automática del código fuente de las principales funcionalidades del método Lexiway, así como de algunos tipos de actividades utilizadas en la metodología Busuu. De forma parcial (modelado) se

han realizado casos de estudio con los principales tipos de actividades de Duolingo y algunos de Babbel y Busuu.

2.1 Resultados obtenidos relativos a la Cuestión de Investigación 1 (CI1)

CI1 - ¿El enfoque MDA sigue siendo un área de interés en la investigación en ingeniería del software? ¿Cuáles son los principales campos donde se aplica con éxito el enfoque MDA? ¿Cuántos artículos que utilizan MDA incluyen generación de código? ¿Existen trabajos relacionados con el desarrollo de aplicaciones dirigido por modelos para el aprendizaje de idiomas?

En las cuatro publicaciones que respaldan esta tesis [22][23][21][24], junto con los otros artículos relevantes presentados en detalle desde el Capítulo 3 al Capítulo 9, se presenta –según corresponde en cada caso, en los apartados de introducción y de *background*– el estado del arte y los trabajos relacionados con esta tesis.

Por otra parte, en [21] se presenta un estudio sistemático (*Systematic Mapping Study*) relativo a la generación de código utilizando el enfoque de arquitecturas dirigidas por modelos (*MDA*). Este estudio sistemático, proporciona una comprensión estructurada del estado del arte –dentro de la ingeniería del software– de la investigación de la generación de código utilizando el enfoque *MDA*. Este estudio se logró –como puede verse en la la Fig. 2.1– mediante la identificación de 50 estudios primarios a partir de 2.145 artículos relacionados con *MDA* publicados durante un periodo de 10 años (2008-2018). Estos 50 trabajos (ver el anexo A *Primary studies* en [21]) han sido analizados con respecto a (i) su frecuencia de publicación por año, (ii) sus canales de publicación, (iii) sus áreas de aplicación, (iv) capas *MDA* involucradas, (v) lenguajes de modelado utilizados, y (vi) *frameworks* utilizados.

En el citado trabajo, se han analizado las amenazas a la validez del estudio y se han descrito las estrategias utilizadas para mitigar sus efectos, así como las limitaciones del estudio.

Para la realización del estudio sistemático, han sido utilizadas las siguientes bases de datos: ACM digital library, IEEE Xplore, ISI Web of Science y Scopus, ya que se sabe que estas fuentes contienen la mayoría de las publicaciones (ver la Tabla 2.1).

Desde las diferentes etapas del mencionado estudio, está claro que:

- Hay un número considerable de trabajos que utilizan *MDA*, publicados en una gran variedad de revistas, congresos y *workshops*, refiridos a una amplia variedad de campos de aplicación, y –además– este número de publicaciones parece estar aumentando.

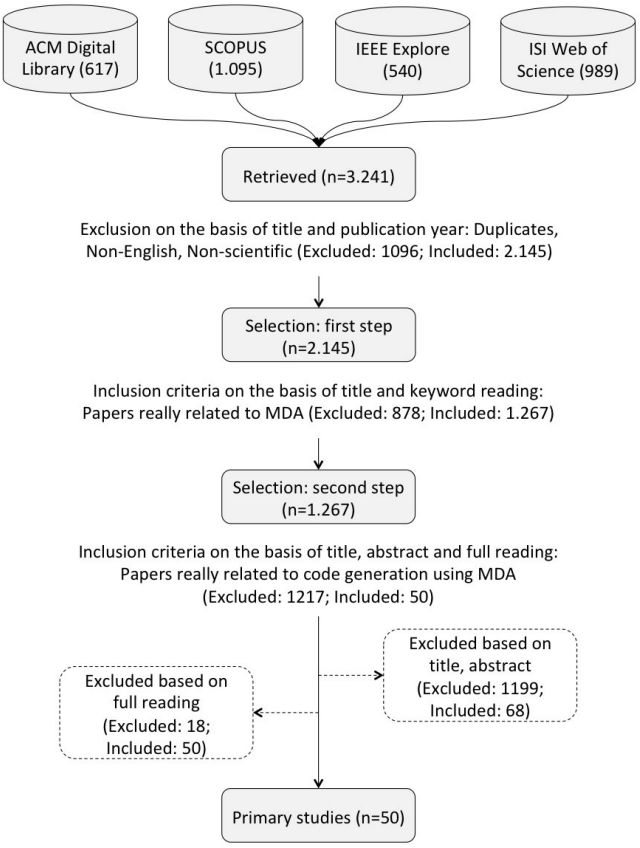


Figura 2.1 Selección de artículos para su inclusión o exclusión (documentos relevantes)

- La mayoría de trabajos con enfoque *MDA* obtenidos del primer paso de selección, están dirigidos a estos campos de aplicación: web en general, seguridad, testeo, simulación, *e-learning*, *IoT*, negocios, diseño *GUI*, sistemas de fabricación, juegos serios, aplicaciones ubicuas, y sistemas multiagente. Debido a esto, se puede decir que *MDA* se usa actualmente en una amplia gama de áreas.

- De los 50 artículos primarios, 35 han sido publicados en conferencias internacionales, y los otros 15 fueron publicados en revistas.

Tabla 2.1 Número de artículos encontrados en cada base de datos desde 2008

Base de datos	Filtro	Número de artículos
ACM Digital Library	Comunicaciones en congresos y artículos de revistas	617
IEEE Xplore	Comunicaciones en congresos y artículos de revistas	540
ISI Web of Science	Artículos de Informática	989
Scopus	Comunicaciones en congresos y artículos de revistas	1095
		(899 duplicados)

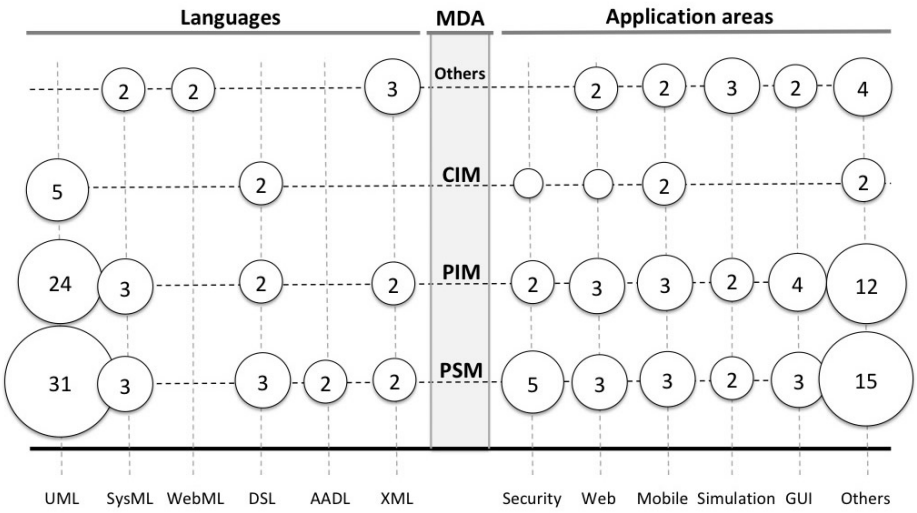


Figura 2.2 Visualización del *mapa sistemático* resultante en forma de *diagrama de burbujas*

En particular, el resultado del *mapa sistemático* reflejado en la Fig. 2.2, muestra que: (a) el lenguaje claramente más utilizado para el modelado de los diferentes meta-modelos de las arquitecturas *MDA*, es *UML* (seguido por *DSL* y SysML); (b) pocos trabajos utilizan en su arquitectura *MDA* una capa *CIM* con modelos independientes de la computación; (c) pocos trabajos utilizan capas de arquitecturas *MDA* diferentes a las clásicas *CIM*, *PIM*, *PSM*; (d) en cuanto a la faceta del área de aplicación, los trabajos dedicados al desarrollo web, al desarrollo móvil, a las *GUI* y a la seguridad, continúan destacando.

Todavía hay un largo camino por recorrer en el campo de la *MDA* y, en muchos casos, la generación automática de código a partir de modelos, sigue siendo el sueño de los ingenieros de software. Además, el desarrollo y posterior publicación de trabajos de investigación que utilizan *MDA* para generar código, sigue siendo complicado.

2.2 Resultados obtenidos relativos a la Cuestión de Investigación 2 (CI2)

CI 2 - ¿Cuáles son las principales características de las aplicaciones de aprendizaje de idiomas? ¿Es posible definir una arquitectura dirigida por modelos para el desarrollo de este tipo aplicaciones que incluyan dichas características?

El estudio y análisis de las diferentes metodologías para el aprendizaje de idiomas, ha permitido obtener –mediante un proceso de abstracción– los elementos comunes a todas ellas [22][23]. Como resultado de la CI2, se presentan en esta sección dichos elementos, así como

su formalización utilizando un meta-modelo ECORE con restricciones en Object Constraint Language (*OCL*).

Veamos en primer lugar dichos elementos comunes. Las metodologías analizadas tienen siempre (1) un conjunto/jerarquía de conceptos/contenidos que se enseñan (**contenidos**); (2) un conjunto de recursos multimedia (**medios**) para presentar los conceptos de aprendizaje o para ilustrar las actividades; tienen además (3) diversos tipos de ejercicios de aprendizaje (**actividades**); (4) una secuencia –o conjunto de secuencias– que las relacionan (**flujo de trabajo**); y (5) un conjunto de plantillas o formas de distribuir los diversos elementos para la visualización de los contenidos y de las actividades (**presentación**).

A partir de estos elementos comunes, se definen un conjunto de meta-modelos, relacionados a partir de una architectura que puede verse en el diagrama de paquetes mostrado en la Figura 2.3. Por otra parte, la Figura 2.4 muestra el diagrama de meta-clases que son parte de cada uno de los paquetes definidos en la arquitectura.

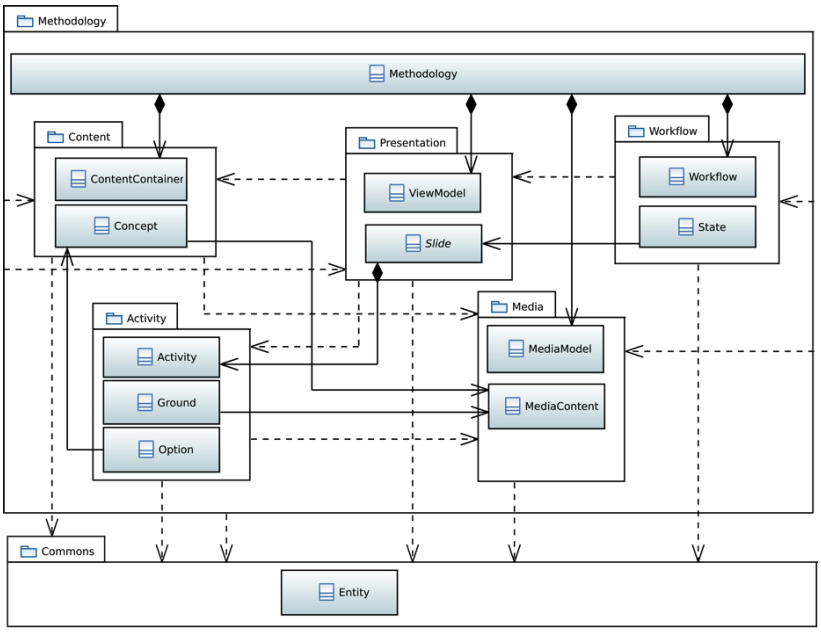


Figura 2.3 Modelo de paquetes del meta-modelo propuesto

El paquete central de esta arquitectura de modelos se denomina *methodology* y se divide en 5 paquetes: *workflow*, *content*, *media*, *activity* y *presentation*.

El paquete *Content* (**CONTENT MODEL** en la Figura 2.4), se utiliza para representar la organización del contenido conceptual que se impartirá al alumno (por ej., nivel, unidad, lección, etc.). Este contenido está estructurado en forma de árbol (*Content*, *ContentContainer*, *Concept*) ya que permite relacionar los conceptos en función de la estrategia metodológica.

El paquete *workflow* (**WORKFLOW MODEL** en la Figura 2.4), permite representar el flujo de la presentación de la aplicación. Para representar este flujo de información se ha adoptado la clásica representación en estructura de grafo, donde los nodos representan los estados (*State*) asociados a una presentación (*Slide* del paquete *presentation*) que puede tomar la aplicación, y donde las aristas representan las diferentes transiciones entre los estados o presentaciones de la aplicación.

El paquete *presentation* (**PRESENTATION MODEL** en la Figura 2.4), se utiliza para representar la interfaz gráfica de los diferentes tipos de ejercicios que se desarrollan en la aplicación durante el proceso de aprendizaje. Este paquete define el concepto de dispositiva (*Slide*) que está asociado a una actividad concreta (*Activity* del paquete *activity*). Esto permite, por ejemplo, la reutilización de la vista de una diapositiva (*Slide*) con diferentes actividades (*Activity*) y viceversa.

El paquete *activity* (**ACTIVITY MODEL** en la Figura 2.4), provee la capacidad de parametrizar el funcionamiento de las distintas actividades que se llevan a cabo durante el proceso de aprendizaje (por ejemplo, una actividad podría consistir en un ejercicio de elección múltiple, otra podría consistir en un ejercicio de rellenar los huecos, etc.). Toda actividad (*Activity*) define información para el usuario (*Ground*) enunciando un ejercicio o actividad; y define la información que introduce el usuario (*Gap*) para llevar a cabo el ejercicio. La introducción de información por parte del usuario es contrastada con información asociada (*Option*) que representa tanto respuestas correctas como opciones candidatas a ser la respuesta o las respuestas correctas. Toda información lleva asociada una o varias representaciones multimedia que permitirán parametrizar la presentación de la actividad.

El paquete *media* (**MEDIA MODEL** en la Figura 2.4), se utiliza para representar los recursos multimedia (medios) que se utilizarán en la presentación, de acuerdo con la actividad que se lleva a cabo. Básicamente, define 4 tipos de recursos multimedia: audio, texto, video e imagen (*Audio*, *Text*, *Video* e *Image*) que pueden ser combinados para representar recursos multimedia complejos (por ej. texto y audio).

Finalmente, el paquete *methodology*, define la meta-clase *Methodology* que representa a una metodología en su conjunto y está definida por el conjunto de modelos definidos por los meta-modelos de contenido, flujo de control, medios y presentación.

Todas las meta-clases de este meta-modelo son descendientes de la meta-clase *Entity*, la cual está definida en un paquete denominado *commons*. Este paquete provee a las demás meta-clases del meta-modelo la capacidad de identificarse (*Entity*) y de extenderse (*Property*).

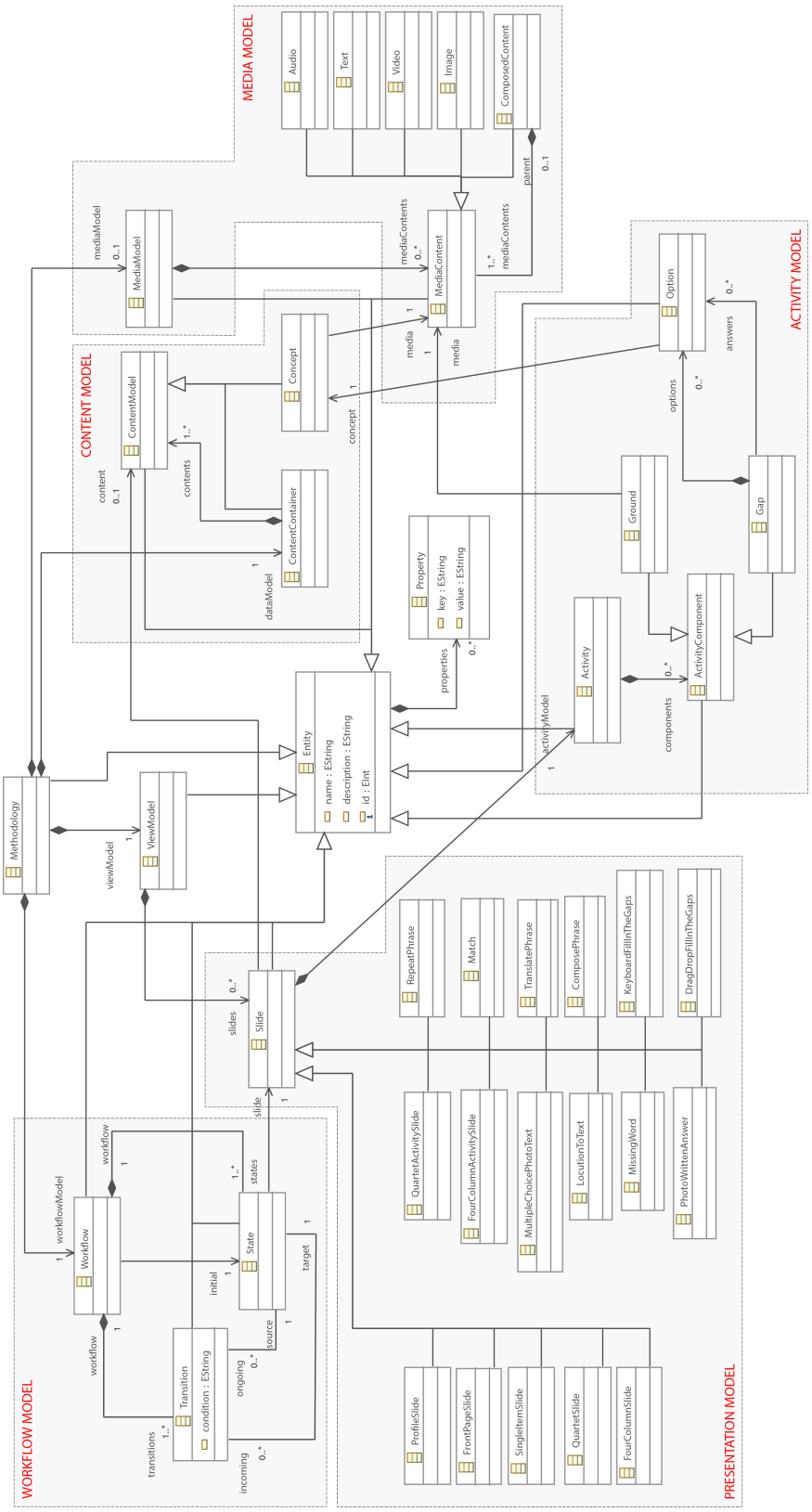


Figura 2.4 Meta-modelo que modela metodologías para el aprendizaje de un idioma

2.3 Resultados obtenidos relativos a la Cuestión de Investigación 3 (CI3)

CI3 - ¿Es posible el desarrollo herramientas capaces de manipular los modelos definidos por la arquitectura alcanzada en la CI2? ¿Se pueden proponer notaciones gráficas que mejoren la experiencia del usuario en relación a la tradicional notación del árbol reflexivo usada por muchos *frameworks* de desarrollo basado en modelos?

En el Capítulo 9, se presenta una notación gráfica de un Lenguaje Específico de Dominio (*DSL*) concebido para representar actividades de aprendizaje de idiomas. Se ha descrito cómo esta notación permite a los desarrolladores representar las características de aplicaciones de aprendizaje de idiomas, usando modelos de flujo de trabajo, de presentación, de contenidos, de medios y de actividades, conforme al meta-modelo que define la sintaxis abstracta del Lenguaje Específico de Dominio (ver la Figura 2.4). Esta notación está implementada como parte de un entorno de desarrollo integrado para construir aplicaciones basadas en modelos. Además, se ha evaluado esta propuesta con un *framework* que utiliza las *dimensiones cognitivas de las notaciones* para sistemas de notación (ver el Apartado 4 de la publicación presentada en el Capítulo 9).

Se puede concluir, que el editor gráfico de diagramas propuesto supera la experiencia que el usuario tiene con el editor de modelos reflexivo. Si bien no hay gran diferencia en lo referente al modelado de modelos de contenidos y al modelado de modelos de recursos de metodologías, sin embargo en lo referente a la creación y edición de modelos de flujo de trabajo y de modelos de presentación/actividades, la solución aportada –a diferencia de la que ofrece el editor reflexivo– es intuitiva y fácil de mantener visualmente. Algunos resultados del análisis llevado a cabo son los siguientes:

1. El mantenimiendo de los submodelos de flujo de trabajo (y también el de actividades) es muy tedioso con el editor reflexivo, entre otras cosas porque muchos elementos y propiedades permanecen ocultos. Con el editor gráfico propuesto se pueden crear y mantener fácil y visualmente los estados, las transiciones entre ellos y los *sub-workflows*.

2. En el editor reflexivo, las operaciones de edición mediante *copy & paste*, son delicadas, y en ocasiones, para encontrar y solucionar errores hay que acudir al editor de texto, con toda la dificultad que eso entraña.

3. El editor propuesto, genera correctamente los modelos que después se utilizan como entrada en el proceso de transformaciones que generan el código fuente (totalmenete funcional) en HTML y JavaScript.

La notación gráfica para la sintaxis abstracta definida en el meta-modelo presentado en [22] y en [23], ha sido implemetada como un plugin Eclipse IDE usando el Eclipse Graphical Framework (GMF)[1] que es parte del Eclipse Modeling Project (GMP)[2].

El objetivo de la sintaxis concreta de un *DSL* es proporcionar a los usuarios una notación intuitiva y cercana a otras que normalmente usan. En el caso del trabajo antes citado, dicha notación facilita a los usuarios la especificación de los modelos de una metodología de aprendizaje de idiomas. La sintaxis concreta normalmente se define como un mapeo entre los conceptos del meta-modelo y su representación textual o gráfica. Así pues, para definir dichas representaciones visuales, es necesario establecer vínculos entre estos conceptos y los símbolos visuales que los representan. Por todo esto, se han elegido para el editor gráfico desarrollado, símbolos visuales de forma que sean tan intuitivos como sea posible. Un ejemplo completo que ilustra la sintaxis concreta adoptada, puede observarse en la Figura 2.6.

La descripción de la notación gráfica se ha realizado siguiendo las tablas de descripción presentadas en [9], y se muestra y describe en las tablas 2.2 y 2.3.

La Interfaz Gráfica de Usuario (*GUI*) del editor desarrollado se compone de los siguientes componentes: una barra de título, un panel de propiedades, una paleta de herramientas y el espacio de trabajo. La Figura 2.5 muestra la distribución de estos elementos en la *GUI*.

Barra de título. En esta barra, se muestran tanto el nombre de la aplicación como el nombre del archivo de trabajo actual.

Panel de propiedades. En este panel, el usuario puede ver y editar las propiedades de los elementos seleccionados en el espacio de trabajo.

Paleta de herramientas. Los elementos de la paleta de herramientas, están organizados en cinco categorías: Contenidos, Recursos, Presentación, Actividades y Flujo de trabajo.

El **Espacio de trabajo**, es el lienzo en el que se sitúan todos los elementos que definirán la metodología que se quiera modelar con el editor gráfico. En la Figura 2.6 se puede ver un ejemplo de contenido del "Espacio de trabajo".

Como ya se ha mencionado, el diagrama de toda metodología consta de 4 sub-diagramas dentro de la misma: el sub-diagrama de los **Contenidos**, el sub-diagrama de los **Recursos**, el sub-diagrama de las **Diapositivas**, y el sub-diagrama del **Flujo de trabajo**. En los 4 casos se utilizan respectivamente 4 contenedores de elementos gráficos, cada uno de los cuales se distingue por un icono identificativo, una descripción en su barra de cabecera, y –en general– por un color asociado a dicho sub-diagrama. Para ilustrar esto, en la Figura 2.6, se

[1] GMF. Eclipse Graphical Framework. URL=https://www.eclipse.org/gmf-tooling/ (último acceso 09/23/19)

[2] GMP. Graphical Modeling Project. URL=https://www.eclipse.org/modeling/gmp/ (último acceso 09/23/19)

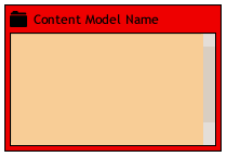
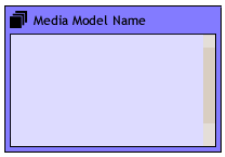
Concepto y **Representación gráfica**	**Base lógica** (forma/icono & color)
Modelo de contenidos 	**Forma**: Ventana rectangular que contendrá cada uno de los niveles y subniveles de contenidos en forma de árbol, así como los conceptos o palabras (hojas del árbol de contenidos). En la cabecera de la ventana mostramos un icono representativo y el nombre del modelo (normalmente se corresponde a la metodología o nivel de la metodología que se esté modelando; por ejemplo, Nivel 11 de Duolingo). Véase un ejemplo en la Figura 2.6 (a). En la Figura 2.7 (b) se muestra visualmente la asociación de contenidos del ejemplo anterior, y en la Figura 2.7 (a) otro ejemplo (en este caso de Busuu methodology). **Color**: La cabecera de esta ventana se pinta de rojo (y el lienzo de la misma de un rojo muy tenue).
Niveles de contenidos 	**Forma**: Se representan con la metáfora de una carpeta, pues contienen conceptos (o subniveles de contenidos). **Color**: Icono de color rojo
Conceptos o palabras 	**Forma**: Se representan con un icono de un concepto, pues representan los conceptos o palabras nuevas que se estudian en una lección. **Color**: Icono de color rojo
Modelo de medios 	**Forma**: Ventana rectangular que contendrá cada uno de los niveles y subniveles de recursos media. En la cabecera se muestra un icono representativo y el nombre del modelo. Véase un ejemplo en la Figura 2.6 (b). En la parte derecha de la Figura 2.8 se muestra visualmente la asociación de recursos del ejemplo anterior. **Color**: Esta ventana se pinta de violeta (y el lienzo de la misma de un violeta muy tenue).
Niveles de media	**Forma**: Se representan con la metáfora de una carpeta, pues contienen recursos compuestos de varios recursos (o subniveles de recursos). En la de la Figura 2.8 se muestran los recursos media de una actividad de tipo *multiple choice* en la metodología Duolingo, para ilustrar lo que se quiere decir con "recursos compuestos". **Color**: Icono de color violeta.
Media 	**Forma**: Se representan con un icono representativo del tipo de recurso de que se trate: texto, audio, video o imagen. **Color**: Icono de color violeta.

Tabla 2.2 Descripción de la notación gráfica (Modelo de contenidos y modelo de recursos)

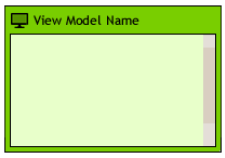
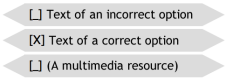
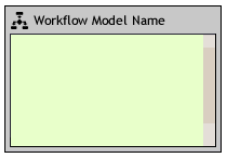
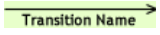
Concepto & **Representación gráfica**	**Base lógica** (forma/icono & color)
Modelo de presentación y de actividades 	**Forma**: Ventana rectangular que contendrá cada uno de las slides y de las actividades. En la cabecera se muestra un icono representativo y el nombre del modelo. Véase un ejemplo en la Figura 2.6 (c). En la Figura 2.10 se ilustra el modelado de una actividad de tipo *Fill-in the gaps* de Busuu. **Color**: Esta ventana se pinta de verde (y el lienzo de la misma de un verde muy tenue).
Enunciados y contenidos pasivos 	**Forma**: Icono de un recurso media genérico. **Color**: Icono de color verde.
Hueco 	**Forma**: Se representa con un icono de un hueco. **Color**: Icono de color verde.
Opción 	**Forma**: Se representan con un rectángulo romboide con un *checkbox* (marcado si se trata de la opción correcta), y con el texto de la opción. Si la opción es de tipo texto, se muestra el principio del texto correspondiente; y si la opción es un recurso multimedia, se muestra entre paréntesis el nombre del recurso multimedia en cuestión. **Color**: Rectángulo romboide de color gris.
Modelo de Flujo de trabajo 	**Forma**: Ventana rectangular que contendrá los estados, las transiciones y los sub-workflows. En su cabecera se muestra un icono representativo y el nombre especificado para el modelo. Véase un ejemplo en la Figura 2.6 (d). En la Figura 2.9 se ilustra otro flujo de trabajo que tiene además un sub-workflow. **Color**: Esta ventana la pintamos de color gris con fondo verde tenue.
Estado 	**Forma**: Rectángulo con un icono representativo, el nombre del estado y la descripción del estado. **Color**: Rectángulo de color verde. Los estados (y el fondo del Modelo de flujo de trabajo) se pintan de verde porque están muy relacionados con las slides y las actividades del Modelo de Actividades.
Transición 	**Forma**: Se representan con una flecha (que conecta el estado origen con el estado destino) junto con una label descriptiva. **Color**: Flecha de color negro.

Tabla 2.3 Descripción de la notación gráfica (Modelo de presentación y flujo de trabajo)

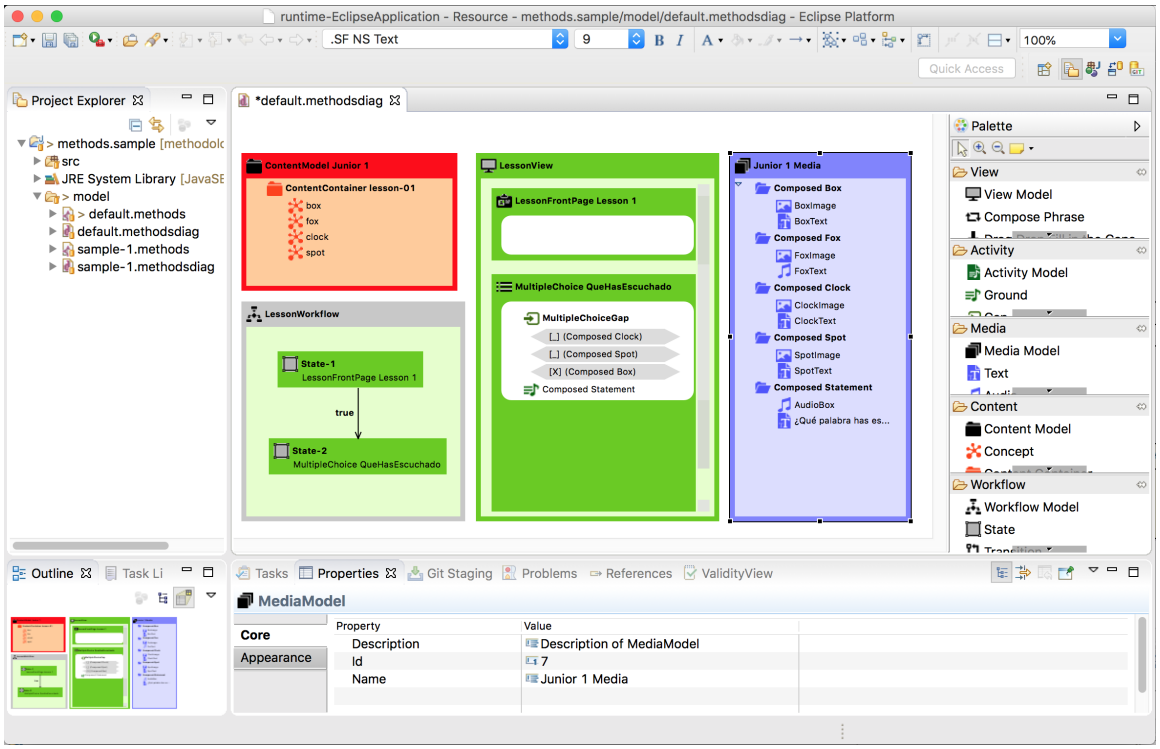


Figura 2.5 Interfaz Gráfica de Usuario del editor gráfico desarrollado

muestran los 4 sub-diagramas que definen una pequeña parte de la metodología y actividades de Duolingo.

Para los Contenidos se utiliza el paradigma de una estructura de tipo árbol, para expresar los distintos niveles y subniveles de los contenidos de un nivel del método de aprendizaje de idiomas que se esté definiendo. Por ejemplo, la metodología Duolingo, está estructurada en Niveles, que a su vez están configurados por Unidades, las cuales constan de Lecciones, y en las cuales se trabajan una serie de Conceptos o palabras. Ver esto ilustrado en Figura 2.7 (b).

Para los Recursos, se ha utilizado también el paradigma de una estructura de tipo árbol para definir la librería de recursos multimedia que serán utilizados en las actividades interactivas. Básicamente, se utilizan carpetas o contenedores de recursos, que a su vez pueden contener sub-carpetas. Dichos recursos o medios, pueden ser de 4 tipos (audio, texto, video o imagen), que pueden combinarse para representar medios “complejos” o “compuestos” (por ej. texto y locución o por ej. imagen y texto, etc.). Ver esto ilustrado en la Figura 2.8.

Para modelar el flujo de trabajo con el editor gráfico, se define un diagrama de estados. Ver esto ilustrado en la Figura 2.9.

Finalmente, para modelar las actividades, como todas ellas –sean del tipo que sean– se modelan según el paradigma de los ejercicios de rellenar huecos (ver en el Apartado 4 de la publicación mostrada en el Capítulo 6), siempre se modelan con recursos para el enunciado

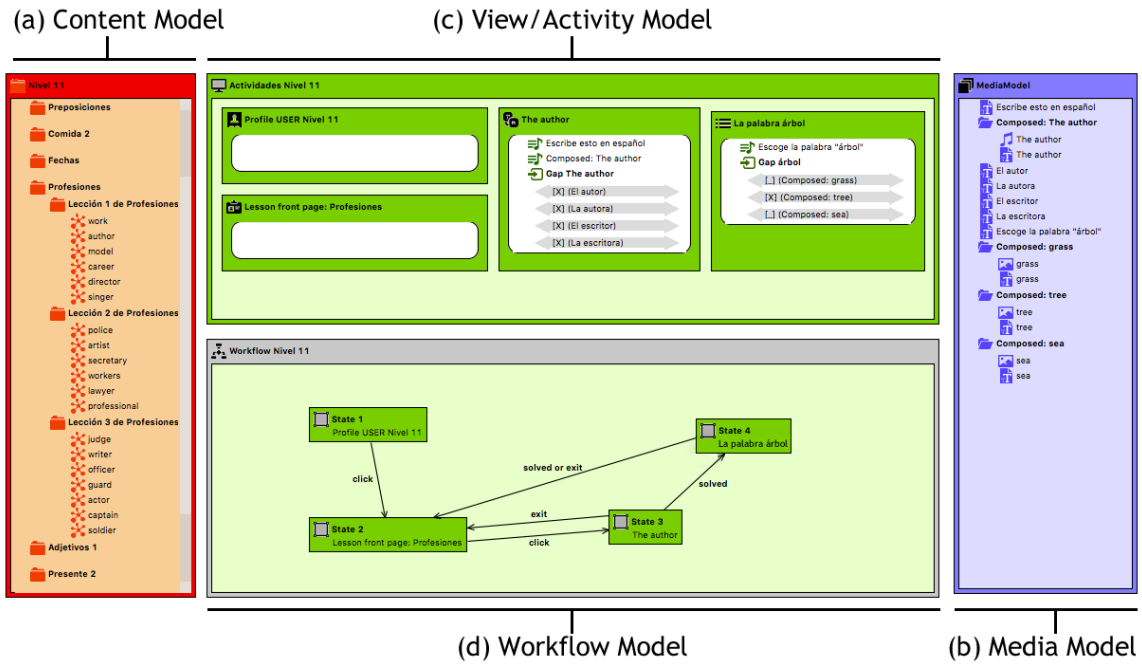
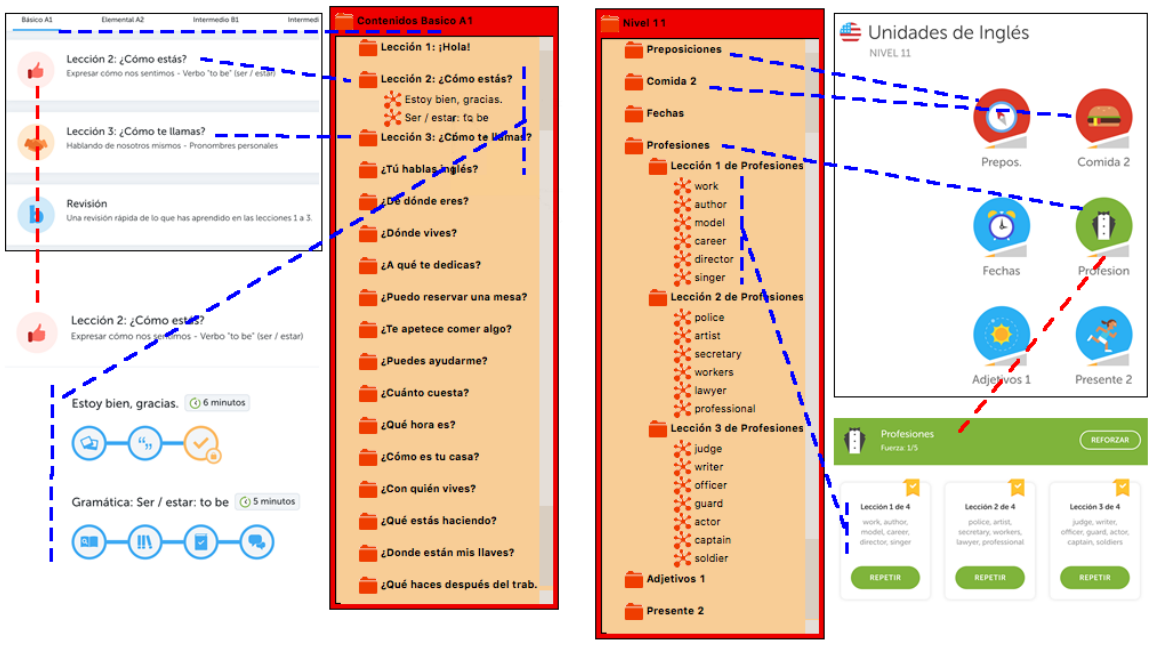


Figura 2.6 Los 4 sub-diagramas que componen el diagrama de una metodología (como Duolingo)



(a) (b)

Figura 2.7 Estructura de árbol para representar contenidos en Busuu (a) y en Duolingo (b)

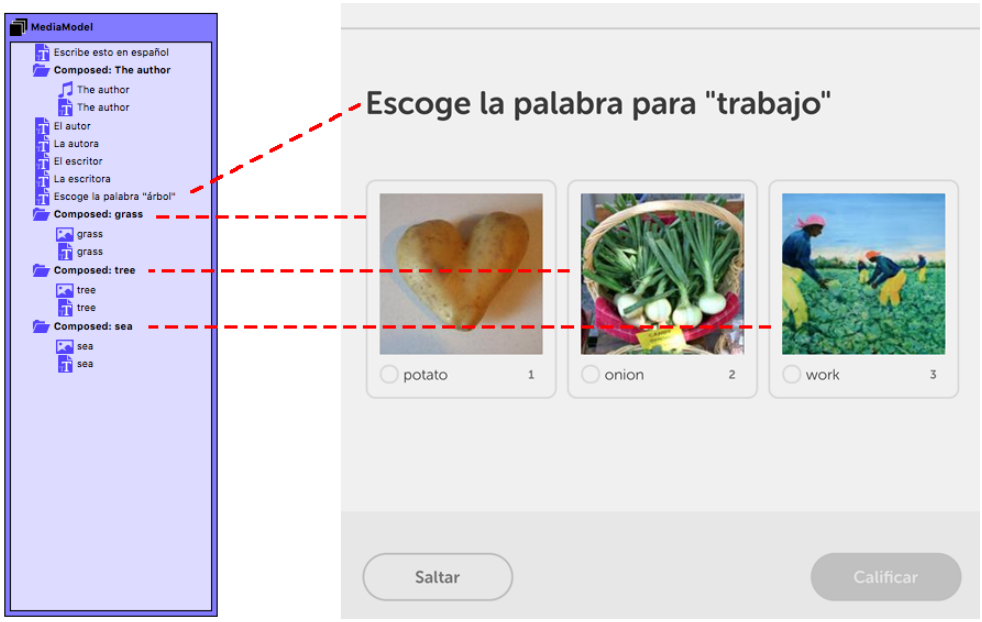


Figura 2.8 Los recursos media de una actividad de elección múltiple en la metodología Duolingo

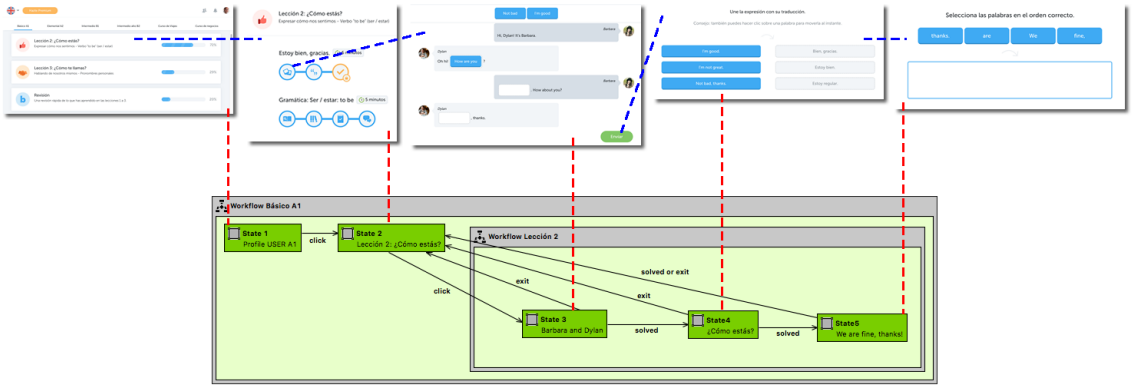


Figura 2.9 Flujo de trabajo de 5 diapositivas en la metodología Busuu

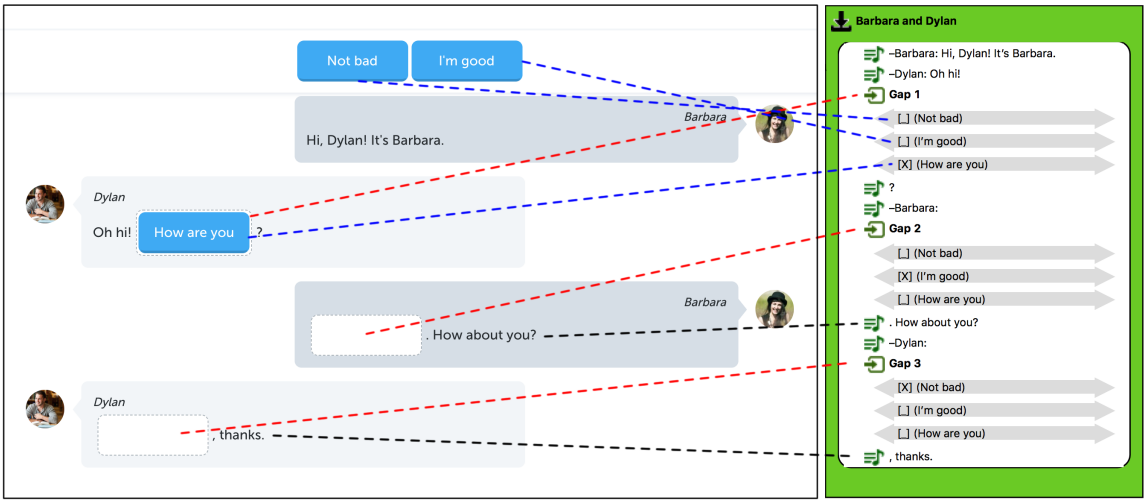


Figura 2.10 Modelando gráficamente una actividad de tipo *Fill-in the gaps* de Busuu

(*Ground*) y con huecos (*Gap*) con las posibles respuestas (válidas o no). Ver esto ilustrado en la Figura 2.10.

2.4 Resultados obtenidos relativos a la Cuestión de Investigación 4 (CI4)

CI4 - ¿Se pueden desarrollar las transformaciones *modelo-a-modelo* y las transformaciones *modelo-a-texto* necesarias para generar el código fuente final de aplicaciones de aprendizaje de idiomas?

En dos de las cuatro publicaciones que respaldan esta tesis ([22][23]), se presenta el meta-modelo del que se ha hablado en el Capítulo 2.2, y más concretamente en el artículo [23] se valida el meta-modelo propuesto, presentando las transformaciones necesarias para generar las principales diapositivas y actividades de la metodología Lexiway. Si bien el meta-modelo quedó validado (y además con un completo editor presentado en el Capítulo 9 y del que se han resumido los resultados en el Capítulo 2.3), las transformaciones estaban muy orientadas hacia la metodología Lexiway, y no respondía a una arquitectura *MDA* en la que se desacoplara la funcionalidad de las aplicaciones de aprendizaje de idiomas. Esto dificultaba el modelar todo lo necesario para el desarrollo (las transformaciones para la generación de código correspondiente) de otras metodologías de aprendizaje de idiomas.

Así pues, en [24] se presenta la propuesta de una completa arquitectura *MDA*, desde una capa *CIM* hasta la capa *ISM* (ver la Figura 2.11), y por otra parte el proceso de las

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

Tabla 2.4 Transformaciones Modelo a Modelo

transformaciones necesarias, para la generación automática del código fuente (en HTML y JavaScript) de aplicaciones de aprendizaje de idiomas (ver la Figura 2.12).

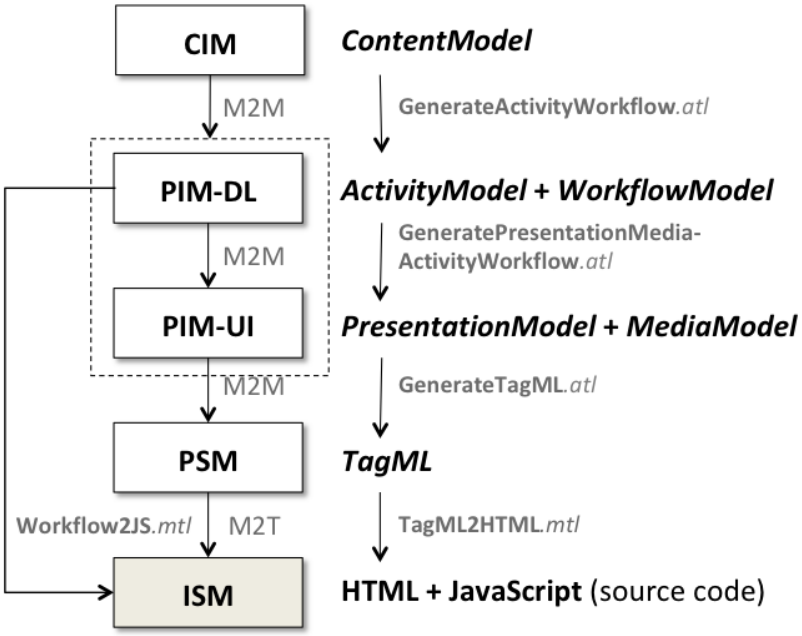


Figura 2.11 Capas y transformaciones de la arquitectura basada en modelos propuesta

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. Para el proceso de transformaciones, se han desarrollado en primer lugar las 3 transformaciones *modelo-a-modelo* necesarias, enumeradas en la Tabla 2.4. Así mismo, para generar los códigos HTML a partir de los modelos TagML pertinentes (ver el Capítulo 7), 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 2.5.

Otro resultado de esta tesis, que se ha utilizado en el proceso de transformaciones antes mencionado (ver la Figura 2.12), es la definición y la descripción de TagML (ver el Capítulo

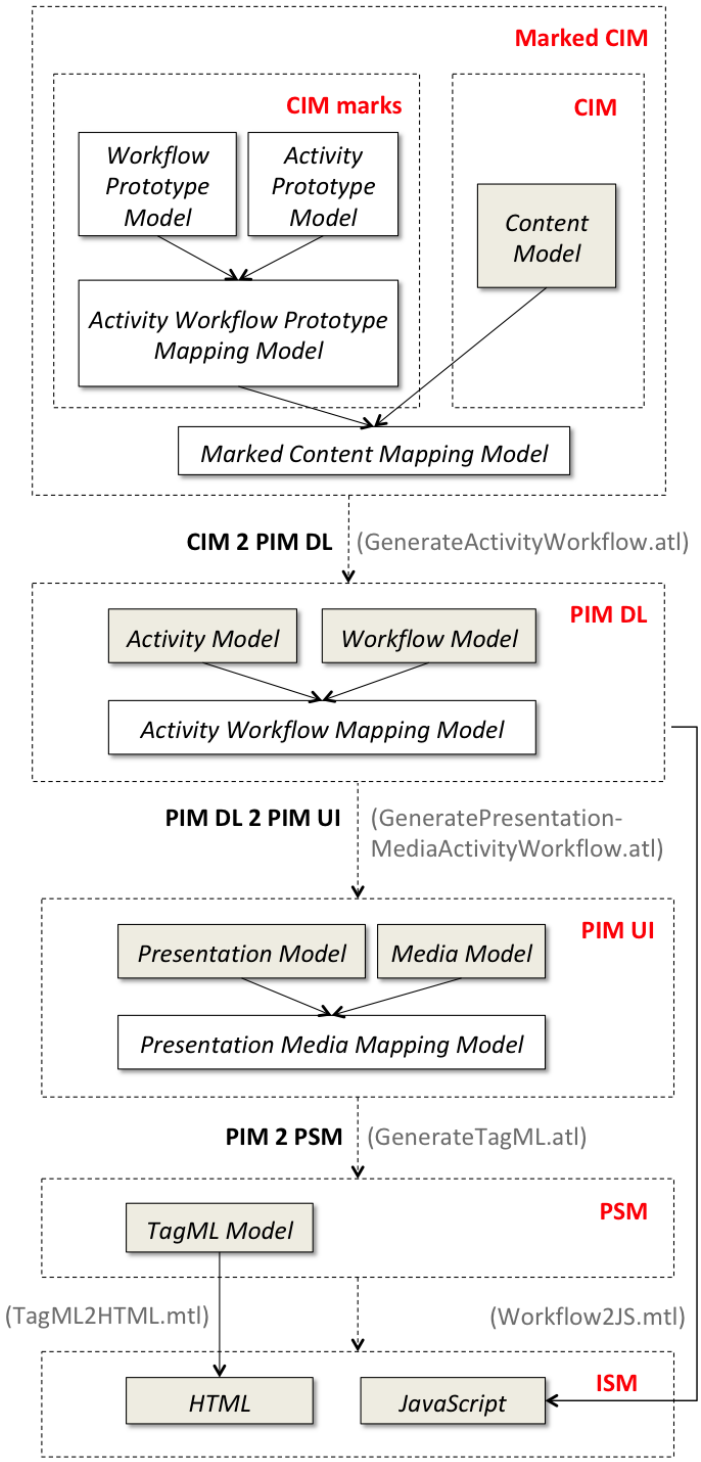


Figura 2.12 Proceso de desarrollo basado en modelos de aplicaciones de aprendizaje de idiomas

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

Tabla 2.5 Transformaciones Modelo a Texto

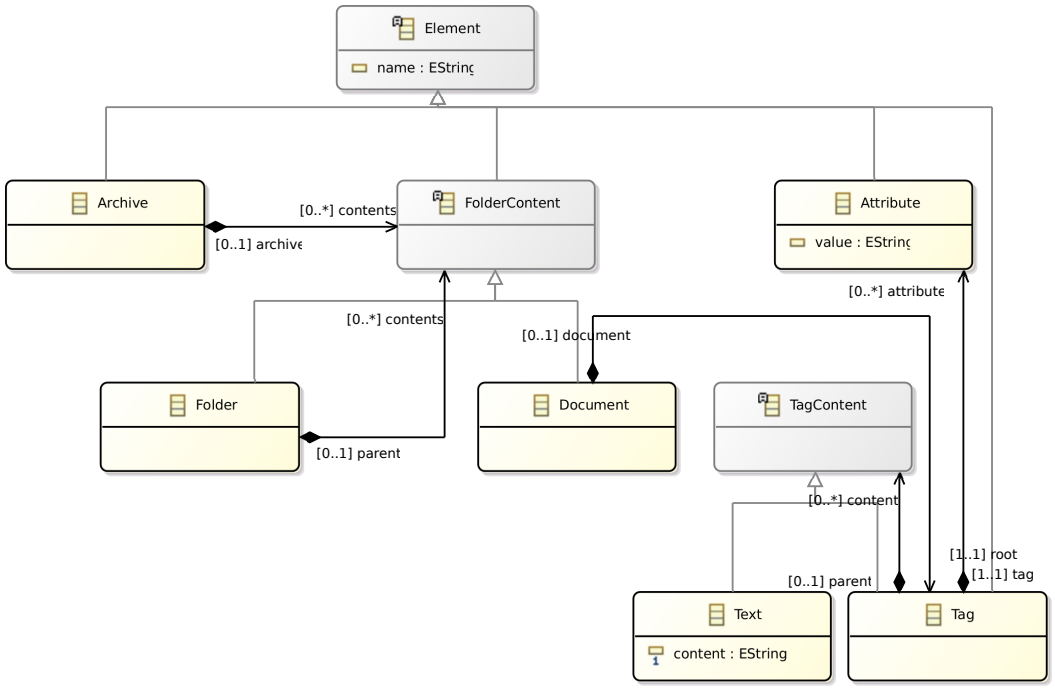


Figura 2.13 El meta-modelo de TagML

7), que es un método para generar documentos XML utilizando transformaciones *modelo-a-modelo* (M2M). Para lograr este objetivo, se ha definido el meta-modelo TagML (ver la Figura 2.13) y la transformación *modelo-a-texto* TagML-to-XML. Mientras que los modelos TagML representan las características esenciales de los documentos XML, la transformación TagML-to-XML genera la representación textual de documentos XML a partir de modelos TagML. Esta aproximación permite a los desarrolladores definir transformaciones *modelo-a-modelo* para generar modelos TagML. Estos modelos se convierten en texto aplicando la transformación TagML-to-XML. En consecuencia, los desarrolladores pueden utilizar lenguajes declarativos para definir transformaciones *modelo-a-texto* que generan documentos XML, en lugar de lenguajes tradicionales basados en arquetipos para definir transformaciones *modelo-a-texto* que generan documentos XML.

La aplicación de TagML, a diferencia de otras propuestas anteriores, implica el uso de transformaciones *modelo-a-modelo* para generar documentos XML en lugar de transformaciones *modelo-a-texto*, lo cual supone una mejora de la legibilidad y la fiabilidad de la

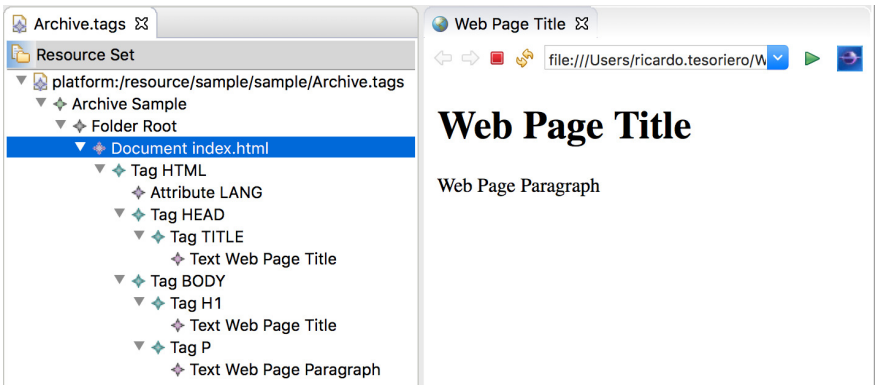


Figura 2.14 Ejemplo del modelo TagML de un documento Web

transformación, así como una reducción del coste de mantenimiento de la transformación. El enfoque propuesto ayuda a los desarrolladores a definir transformaciones menos propensas a errores que con el uso del enfoque tradicional. Además, la simplicidad del enfoque propuesto permite la generación de documentos XML sin la necesidad de ninguna configuración de la transformación, lo que no penaliza la reutilización del modelo.

Para llevar a cabo las transformaciones *modelo-a-modelo* y *modelo-a-texto*, se han utilizando los lenguajes de transformación ATL[3] y ACCELEO[4] respectivamente.

2.5 Resultados obtenidos relativos a la Cuestión de Investigación 5 (CI5)

CI5 - ¿Se puede –mediante el desarrollo de aplicaciones dirigido por modelos– modelar y generar el código fuente de aplicaciones como Lexiway o Duolingo? ¿Es posible la validación del meta-modelo propuesto y de las transformaciones desarrolladas, mediante casos de estudio de actividades reales de metodologías como Busuu o Duolingo?

2.5.1 Caso de estudio 1

En [22], publicación que respalda esta tesis, se presenta un caso de estudio que ilustra la versatilidad del meta-modelo propuesto para modelar distintos métodos de enseñanza de idiomas (ver el Capítulo IV del artículo antes referenciado). Para ello, se muestra un ejemplo de modelo que incluye una actividad que da soporte a un ejercicio de tipo elección múltiple

[3]Eclipse Foundation 3, "ATL: a model transformation technology", Eclipse Foundation, 2019. URL = https://eclipse.org/atl/ (último acceso 12/02/2019)

[4]Obeo, "Acceleo", Obeo, 2019. URL = https://www.eclipse.org/acceleo/ (último acceso 12/02/2019)

en el que el usuario escucha una palabra y debe elegir entre 4 opciones de pares imagen-texto. Señalando los elementos comunes de dicha actividad en dos metodologías distintas (como son Duolingo y Lexiway Junior), y su reflejo en el modelo común, se constata y valida la versatilidad del meta-modelo propuesto (ver la Figura 2.15).

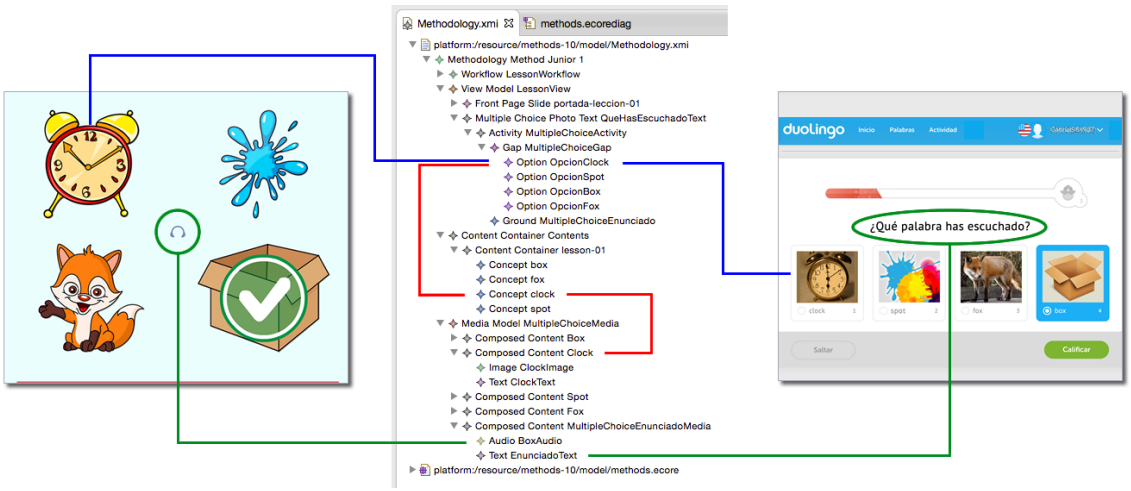


Figura 2.15 Presentación en Lexiway y en Duolingo de una actividad tipo elección múltiple con los mismos elementos

En dicho caso de estudio, se muestra también un ejemplo (ver la Figura 2.16) que ilustra cómo se modela la presentación de dos diapositivas en Lexiway, así como el modelo del flujo de control o *workflow* necesario para pasar de una diapositiva a la otra.

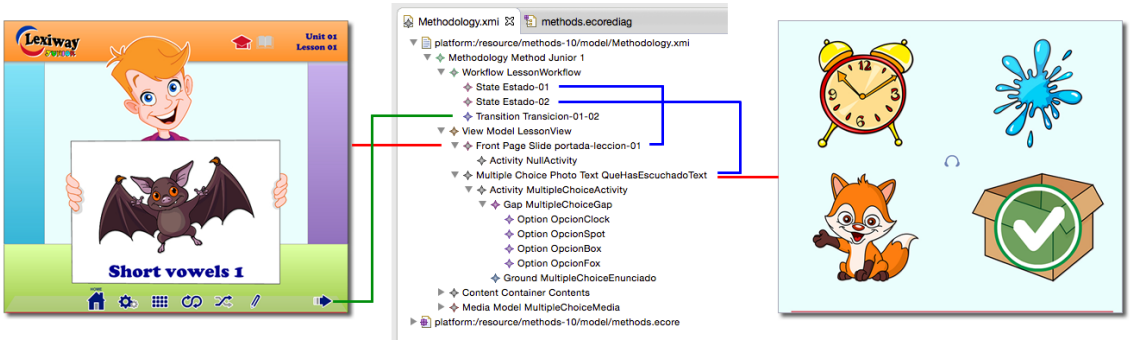


Figura 2.16 Presentación en Lexiway de dos *slides* y su flujo de control (*workflow*)

2.5.2 Caso de estudio 2

En [23], publicación que respalda esta tesis, se presenta una validación de la propuesta (ver el Capítulo 6 de la mencionada publicación) que presenta cómo se mapean los modelos del meta-modelo de metodologías, en el código fuente generado de las respectivas aplicaciones.

Para dicha validación se ha seleccionado el modelo de la metodología Lexiway presentado en las Figuras 2.16 y 2.15, y el lenguaje de marcado de hipertexto (HTML) como el leguaje destino para mostrar cómo se mapean los modelos de una metodología en el código fuente generado.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <title>practice-01-04</title>
5    <meta charset='utf-8'>
6    <meta name='viewport' content='width=device-width, initial-scale=1'
     >
7    <link rel='stylesheet' href='lexiway.css' type='text/css'>
8    <script src='https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/
     jquery.min.js'>
9    </script>
10   <script src='http://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/
     bootstrap.min.js'>
11   </script>
12   <script src='script.js'>
13   </script>
14 </head>
15 <body onload='playSound(this,"/media/MultipleChoiceMedia/Box/
     BoxAudio.mp3");'>
16   <div id='container'>
17     <div class='quartet-item quartet-item-1'>
18       <img src='/media/MultipleChoiceMedia/Clock/ClockImage.png'
         class='quartet-item quartet-item-' onclick='checkAnswer(this,4,
           1);' alt='clock'> </img>
19       <div id='practice-check-1'> </div>
20     </div>
21     <div class='quartet-item quartet-item-2'>
22       <img src='/media/MultipleChoiceMedia/Spot/SpotImage.png' class=
         'quartet-item quartet-item-' onclick='checkAnswer(this,4,2);'
         alt='spot'> </img>
23       <div id='practice-check-2'> </div>
24     </div>
25     <div class='quartet-item quartet-item-3'>
26       <img src='/media/MultipleChoiceMedia/Fox/FoxImage.png' class='
         quartet-item quartet-item-' onclick='checkAnswer(this,4,3);'
         alt='fox'> </img>
27       <div id='practice-check-3'> </div>
28     </div>
29     <div class='quartet-item quartet-item-4'>
30       <img src='/media/MultipleChoiceMedia/Box/BoxImage.png' class='
         quartet-item quartet-item-' onclick='checkAnswer(this,4,4);'
         alt='box'> </img>
31       <div id='practice-check-4'> </div>
32     </div>
33     <div id='practice-enunciado' onclick='playSound(this,"/media/
         MultipleChoiceMedia/Box/BoxAudio.mp3")'> </div>
34   </div>
35   <div id='actions-container' onmouseover='showActions()' onmouseout=
     'hideActions()'> </div>
36 </body>
37 </html>
```

Figura 2.17 Código fuente generado a partir de la instancia de *MultipleChoicePhotoText* "QueHasEscuchadoText".

La Figura 2.17 muestra el código fuente generado a partir de la instancia de *MultipleChoicePhotoText* "QueHasEscuchadoText". Mientras que las líneas 1–14 presentan el código para cargar los *frameworks* web tales como Bootstrap[5] y jQuery[6], las lineas 15–36 presentan

[5]Bootstrap (página principal). URL = https://getbootstrap.com/ (último acceso 14/02/20)
[6]jQuery (página principal). URL = https://jquery.com/ (último acceso 14/02/20)

la composición de la diapositiva mostrada en la parte izquierda de la Figura 2.15, la cual está organizada en una cuadrícula consistente en 2 filas por 2 columnas.

2.5.3 Caso de estudio 3

En el Apartado 4 de la publicación mostrada en el Capítulo 6, se proponen algunos ejemplos de cómo utilizar el paradigma de actividades de tipo *Fill-in the Gaps* con el paquete *activity* (ACTIVITY MODEL en la Figura 2.4), como una abstracción universal para representar los diferentes tipos de actividades de aprendizaje más frecuentemente usados en metodologías como Babbel, Bussu o Duolingo.

Así pues, en el artículo antes referenciado, se ilustra cómo modelar actividades de *Fill-in the gaps*, *Word ordering* y *Match-up* en Busuu (Figuras 2.18, 2.19 y 2.20 respectivamente), así como actividades de *Locution to text*, *Multiple choice* y *Translate phrase* en Duolingo, y actividades de *multiple choice* en Babbel, Bussu y Duolingo.

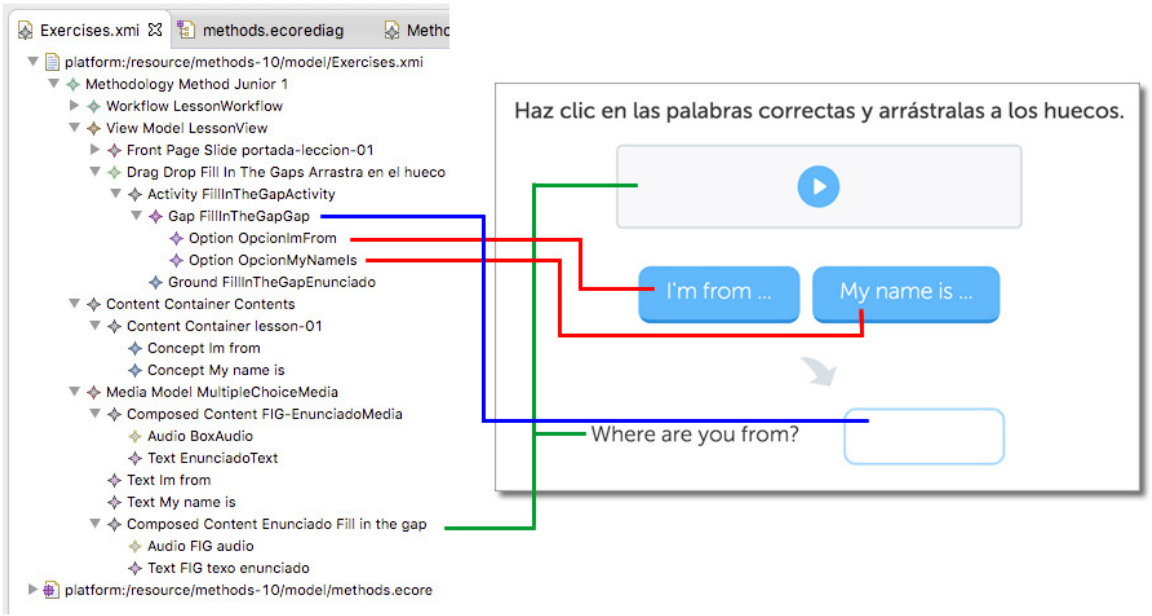


Figura 2.18 Modelado de una actividad de *Fill-in the gaps* en Busuu.

2.5.4 Caso de estudio 4

También en la publicación mostrada en el Capítulo 6 (en su Apartado 5), se presenta el proceso de preparación y los resultados de una evaluación que se ha llevado a cabo. Se trata de una **evaluación de usuarios** para analizar la **calidad en uso** (la efectividad y la eficiencia

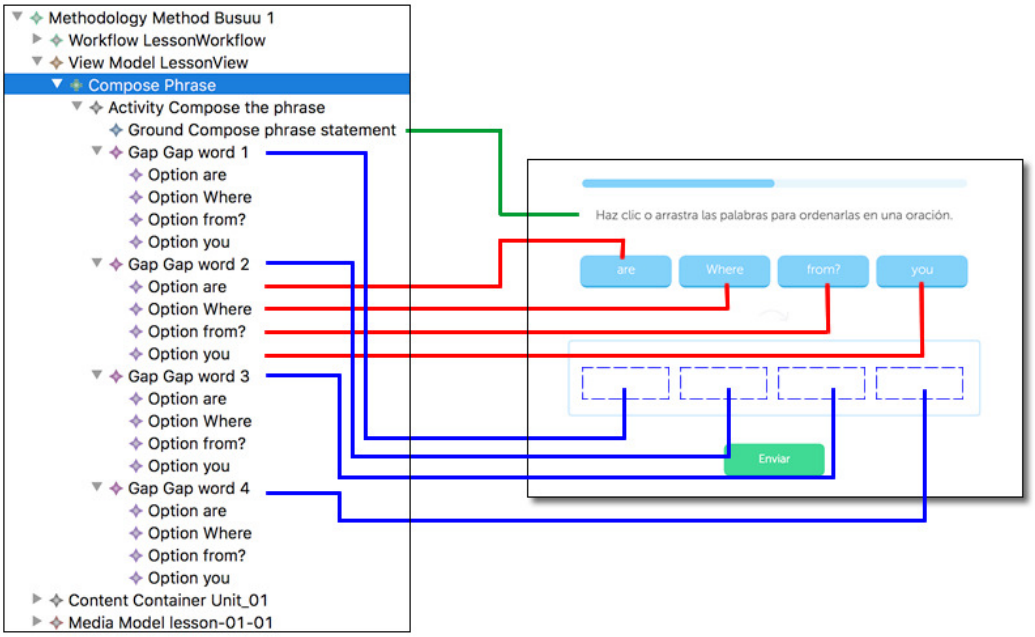


Figura 2.19 Modelado de una actividad de *Word ordering* en Busuu

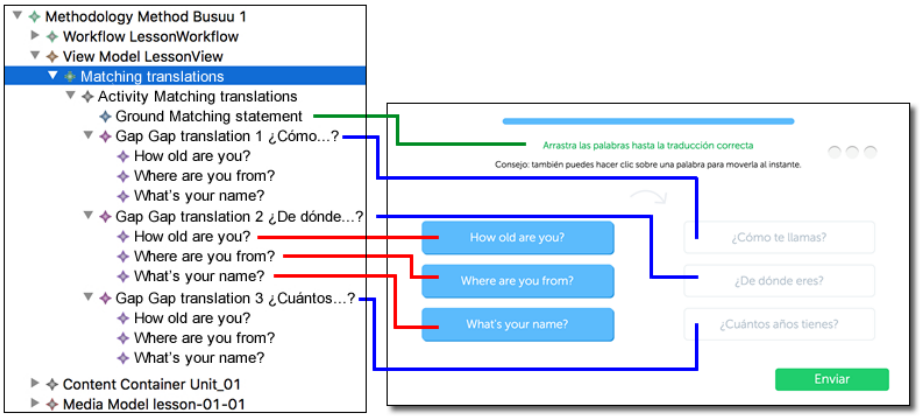
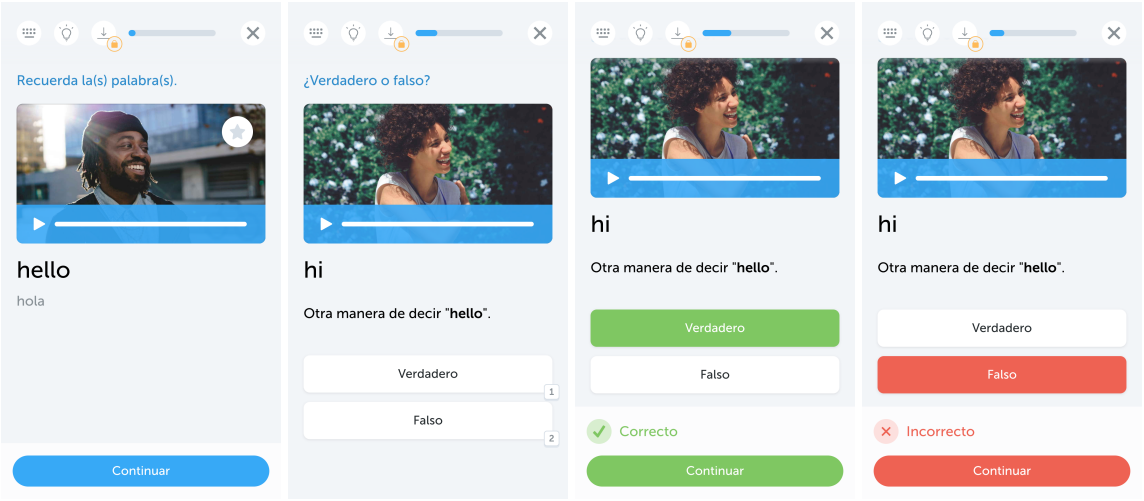


Figura 2.20 Modelado de una actividad de *Match-up* en Busuu

del proceso de desarrollo), así como para analizar la **calidad del producto** (corrección, robustez, extensibilidad y reutilización del código fuente generado con el enfoque *MDA* de esta tesis, en relación al obtenido mediante una aproximación o enfoque tradicional).

Por un lado, para la evaluación de la calidad en uso se ha seguido el estándar ISO/IEC 25062 [12], que compara la efectividad y la eficiencia de los procesos de desarrollo propuestos y tradicionales. Por otro lado, para la evaluación de la calidad del producto, que compara el código fuente obtenido utilizando los enfoques tradicionales con los generados con el enfoque propuesto, se ha seguido el estándar ISO/IEC 25010:2011(E) [13].



(a) (b) (c) (d)

Figura 2.21 Mecanismos de actividades de aprendizaje *Imagen-Audio-Texto* y *Opciones de Audio-Texto* en Busuu

Los resultados de dicha evaluación, concluyen que el código generado usando el enfoque propuesto supera el código generado usando un enfoque tradicional en 4 métricas para evaluar software: *corrección*, *robustez*, *extensibilidad* y *reutilización*.

2.5.5 Caso de estudio 5

En [24], publicación que respalda esta tesis, se presenta –también para validar la propuesta– el proceso de transformaciones y la completa generación automática del código fuente de dos tipos de actividades utilizadas de forma muy parecida en metodologías como Duolingo y Busuu: el Mecanismo de Actividades de Aprendizaje (LAM) Imagen-Audio-Texto (ver la Figura 2.21 (a)) y el *LAM* Opciones de Audio-Texto (ver la Figura 2.21 (b)(c)(d)), descritas en el Apartado II del artículo antes referenciado.

2.5.6 Caso de estudio 6

En el Apartado 4 de la publicación presentada en el Capítulo 9, se presenta la evaluación de la notación descrita en dicho trabajo. Para llevar a cabo esta evaluación, se ha utilizado un *framework* que utiliza las “Dimensiones Cognitivas de las Notaciones” para sistemas de notación, presentadas en [3] y –de forma más extendida– en [4]. Este *framework* proporciona a los diseñadores de interfaces de usuario, un vocabulario común, de cara a realizar evaluaciones de interfaces de usuario, de forma comprensible. Un resumen de dichas “Dimensiones Cognitivas de las Notaciones” se presenta en la Tabla 2.6.

#	Dimension	Aspect
1	Viscosity	Resistencia al cambio
2	Visibility	Posibilidad de ver componentes fácilmente
3	Premature commitment	Restricciones en el orden de hacer las cosas
4	Hidden dependencies	Los enlaces importantes entre entidades no están visibles
5	Role-expressiveness	El propósito de una entidad se deduce fácilmente
6	Error-proneness	La notación invita a errores y el sistema ofrece poca protección
7	Abstraction (redefinitions)	Tipos y disponibilidad de mecanismos de abstracción
8	Secondary notation	Información adicional en medios distintos a la sintaxis formal
9	Closeness of mapping	La cercanía de la representación al dominio
10	Consistency	Semánticas similares se expresan en formas sintácticas similares
11	Diffuseness	Verbosidad del lenguaje
12	Hard mental operations	Alta demanda de recursos cognitivos
13	Provisionality	Grado de compromiso con acciones o marcas
14	Progressive evaluation	El trabajo hasta la fecha se puede verificar en cualquier momento

Tabla 2.6 Resumen de las "Dimensiones Cognitivas de las Notaciones" para sistemas de notación

Los resultados de esta evaluación son:

1. Existen una serie de aspectos más destacables del editor desarrollado: Una adecuada gestión y resistencia al cambio (*Viscosity*), una completa oferta de mecanismos no formales para plasmar información extra (*Secondary notation*), y la ausencia de verbosidad de lenguaje (*Diffuseness*).

2. En segundo lugar, el editor propuesto también destaca por las acertadas soluciones (gráficas y de interacción) alcanzadas en lo relativo a otros 8 importantes *notational dimensions*: *Visibility*, *Premature commitment*, *Hidden dependencies*, *Error-proneness*, *Closeness of mapping*, *Consistency*, *Provisionality* y *Progressive evaluation*. Como es lógico, algunos de estos últimos aspectos están relacionados entre sí: Por ejemplo, la buena valoración en *Premature commitment* y *Error-proneness* se debe a que efectivamente el editor ofrece restricciones en el orden de hacer las cosas, lo cual proporciona a su vez al usuario una adecuada protección ante posibles errores. Así mismo, la buena valoración en los aspectos de la *Provisionality* y la *Progressive evaluation*, muestra de que el editor tiene un adecuado compromiso con las acciones, permitiendo a los usuarios realizar acciones provisionales, y facilitándoles a la vez el validar –en cualquier momento– lo que están haciendo.

3. Finalmente, los aspectos *Role-expressiveness*, *Abstraction* y *Hard mental operations*, que aún siendo adecuados, tienen una peor valoración, también están relacionados entre sí. La valoración adoptada responde a que: (1) en los diagramas generados con el

Publicación	CI	Cap.
* **Modeling Language-Learning Applications** (Gabriel Sebastián, Ricardo Tesoriero, José A. Gallud) (2017) (IEEE LAT, **Q4**)	1, 2, 5	Todos
* **Model-based approach to develop learning exercises in language-learning applications** (Gabriel Sebastián, Ricardo Tesoriero, José A. Gallud) (2018) (IET Software, **Q4**)	1, 2, 4, 5	3-8
A Unified Abstract Mechanism to Model Language Learning Activities (Gabriel Sebastián, Ricardo Tesoriero, José A. Gallud) (2020) (CAI, **Q4**)	1, 2, 4, 5	3.1, 4, 5
* **Automatic Code Generation for Language-Learning Applications** (Gabriel Sebastián, Ricardo Tesoriero, José A. Gallud) (2019) (IEEE LAT, **Q4**)	1, 2, 4, 5	3, 4
TagML: An Implementation Specific Model to Generate Tag-based Documents (Ricardo Tesoriero, Gabriel Sebastián, José A. Gallud) (2019) (ACM Transactions on the Web, **Q2**)	4, 5	Todos
* **Code Generation Using Model Driven Architecture: A Systematic Mapping Study** (Gabriel Sebastián, José A. Gallud, Ricardo Tesoriero) (2019) (COLA, **Q2**)	1, 4	Todos
A Visual DSL for automatic generation of Language-learning Applications (Gabriel Sebastián, Ricardo Tesoriero, José A. Gallud) (2019) (PLOS-ONE, **Q2**)	1, 2, 3, 5	2-5

Tabla 2.7 Principales publicaciones llevadas a cabo durante la tesis

editor, el propósito de algunas entidades no siempre se deducen fácilmente, (2) el editor no permite la definición de nuevos mecanismos de abstracción, y (3) en ocasiones al usuario puede no resultarle evidente lo que otro autor ha modelado.

2.6 Resumen de los artículos publicados y enviados

La Tabla 2.7 muestra la lista de artículos publicados y enviados (en proceso de revisión), como resultado de esta tesis. Nótese que 4 de esas publicaciones que avalan esta disertación por compendio de publicaciones, están resaltadas con un asterisco antes del título. A lo largo de cada una de las publicaciones, las preguntas de investigación están –en mayor o menor medida– expuestas y representadas. Finalmente, la tercera columna muestra el capítulo donde el trabajo en cuestión presenta o resuelve la pregunta de investigación.

Capítulo 3

Code Generation Using Model Driven Architecture: A Systematic Mapping Study

Citación: G. Sebastian, R. Tesoriero, J. A. Gallud, Code Generation Using Model Driven Architecture: A Systematic Mapping Study, Journal of Computer Languages (2019). doi:10.1016/j.cola.2019.100935.

Tipo de publicación: Revista Internacional (IF: 1.714, Q2)

Esta publicación avala esta disertación por compendio de publicaciones. Este artículo se envió originalmente a *Computer Languages, Systems & Structures* (IF: 1.714, Q2), sin embargo, durante el proceso de revisión, la revista experimentó un cambio de nombre y ahora se titula *Journal of Computer Languages*.

Journal of Computer Languages 56 (2020) 100935

Contents lists available at ScienceDirect

Journal of Computer Languages

journal homepage: www.editorialmanager.com/cola/default.aspx

Code generation using model driven architecture: A systematic mapping study☆

Gabriel Sebastián[a], Jose A. Gallud[*,b], Ricardo Tesoriero[b]

[a] Albacete Research Institute of Informatics, University of Castilla-La Mancha, Albacete (02071), Spain
[b] Computing Systems Department, Faculty of Computer Science Engineering, University of Castilla-La Mancha, Infante Don Juan Manuel Building, Albacete (02071), Spain

ARTICLE INFO

Keywords:
Model-driven architecture
Software engineering
Systematic mapping study

ABSTRACT

Model Driven Architecture (MDA) has earned a prominent place in Software Engineering. However, while the application of MDA in software industry has grown, its impact in industry is not being as fast and broad as predicted. Moreover, the emerging use of agile methods have had an effect on the interest of industry and academics on developing software using metamodels. As a result, it is difficult to establish the true interest of practicioners about code generation using model driven architectures. This study investigates the scientific evidence of model driven architecture by conducting a systematic mapping of MDA literature in software engineering between 2008 and 2018. The search strategy resulted in 2.145 studies, of which 50 were identified as primary articles relevant to this study. We have verified that there is a considerable number of research articles that make use of MDA, published in a variety of journals, congresses and workshops, referred to very varied fields of application, and this number of publications seems to be increasing. The study also shows that, while there are few relevant publications that specifically deal with code generation using MDA, most of them show how this technology can be applied successfully to enhance software development.

1. Introduction

One of the software engineers' dream is the automatic generation of code from models, in the same way that the civil architects begin the designing of a house by drawing plans. The recent history of software engineering is full of promises of automatic generation of code from models, the days of writing code were about to end. Meanwhile, the Agile Age is cornering the models, the modeling languages and, as a consequence, also the metamodels and MDA.

However, there are some software engineers that are insisting in generating code from models. Some of them are using model driven development, or even model driven architecture. The reasons that motivated the beginning of Model Driven Architecture are there, in the same place, with the same challenges.

The development of software usually becomes a repetitive and tedious process. The process involves repeating the same tasks many times having duplicated code, which is difficult to maintain. Moreover, the problem of duplicated code and task repetition is multiplied by the number of different target platforms, making the overall process more complex and prone to errors. This approach, aka the traditional approach, can be improved using a Model-driven Architecture (MDA) to capture common features in Platform Independent Models (PIMs).

There are two conceptual tools that software engineering has traditionally employed to accomplish these challenges. The first tool is increasing the level of abstraction which is proportional to the amount of code to be generated. The second one is encouraging the reuse of models through the definition of metamodels. Therefore, there are many reasons that justify the use of MDA technology.

However, the implantation of this technology in the industry is not wide enough even if it has been around for more that 20 years. There seems to be many factors that may lead to the under-utilization of this approach in production environments, such as the high cost of qualified professionals in the area or the lack of reliable tools to support this technology in production environments. Having these issues in mind, the code generation is a key faction to take into account in production environments; thus, this study is focused on MDA approaches that end up in the source code generation.

To address this knowledge gap, the goal of this study is to identify

☆ This work has been partially supported by the project TecnoCRA (ref: SBPLY/17/180501/000495) granted by the regional government (JCCM) and the European Regional Development Funds (FEDER). This article was originally submitted to Computer Languages, Systems & Structures, however during the process of review the journal underwent a name change and is now titled Journal of Computer Languages.

* Corresponding author.
E-mail addresses: gabriel.sebastian@uclm.es (G. Sebastián), jose.gallud@uclm.es (J.A. Gallud), ricardo.tesoriero@uclm.es (R. Tesoriero).

https://doi.org/10.1016/j.cola.2019.100935
Received 4 August 2019; Received in revised form 9 October 2019; Accepted 13 November 2019
Available online 26 November 2019
2590-1184/

the state-of-the-art of MDA in Software Engineering by conducting a systematic mapping study. In particular, this study is focused in investigating if MDA remain a topic of interest in research (Research Question 1), and detecting the main fields where MDA is applied successfully (Research Question 2).

We focus our study in the MDA software technology, excluding other MDE (Model Driven Engineering) approaches that also generate source code, which are out of scope of this study because we are only interested in the architectural aspect of the model driven engineering.

This paper is organized as follows. Section 2 describes the related work. Section 3 describe the methodology applied in this study. Section 4 presents the answers to the research questions. Section 5 includes a brief discussion on the study. Section 6 analyzes the validity of the study. Finally, we present conclusions.

2. Background and related work

The Model-driven Architecture (MDA)[1] approach proposed by the Object Management Group (OMG) in 2001 presents a set of tools to abstract common elements of an application to improve the software development. This solution gives a leading role to models in the software development during all phases (i.e. inception, design, building, development, and maintenance).

The main reason behind this approach is the constant evolution of the software technologies. Following a traditional development approach, the functionality code and the implementation technology code are interleaved. Consequently, when the technology is enhanced, the functionality is rewritten using the new technology.

Under these scenarios, MDAs introduce abstraction levels to promote the software reuse by emphasizing the design-time interoperability [1]. This kind of interoperability is possible due to the specification of Platform Independent Models (PIMs) that enable developers to separate the specification of the application functionality from the technology that implements it.

Thus, it is possible to reuse the specification of the application functionality for different implementation technologies. Moreover, this functionality can be executed on different hardware and software platforms only with minor changes.

The source code of applications is automatically derived from models using model transformations [2].

In summary, the use of the MDA technology enables the generation of multi-platform applications from PIMs. This fact leads to several advantages; for example, let us assume we want to develop an application for different platforms (e.g. iOS, Web and Android). Following a traditional approach, we should develop 3 different and independent source code projects. Following an MDA approach, we specify only one PIM to generate the source code for the 3 platforms.

The core of the MDA infrastructure is defined in terms of the following OMG standards: the Unified Modeling Language (UML)[2], the Meta Object Facility (MOF)[3], XML Metadata Interchange (XMI)[4] and the Common Warehouse Metamodel (CWM)[5] which were successfully used in the modelling and development of modern systems.

From the Human-Computer Interaction perspective, we can find different approaches that makes use of models to generate user interfaces.

For works focuses on the development of interactive systems, the Model-based User Interface Development (MbUID) provides useful elements to analyse based on the CAMELEON Reference Framework (CRF) [3].

In recent years, other approaches such as [4] have also encouraged the use of models to develop multi-modal user interfaces.

The use of Model-driven Development (MDD) for learning applications was applied in different works, such as those exposed in [5–7]. However, none of them formalizes the definition of language learning activities using OMG compliant metamodels. Nevertheless, there are several works that use MDD techniques based on MDAs to develop Web applications [8,9].

An interesting approach that defines a MDA to develop music learning applications is exposed in [10]. Some of the most relevant works regarding the development of e-learning applications we can find those presented in [11,12].With regard to the use of models to build Web sites, some methodologies (e.g. [13]) and models (e.g. RMM [14], WebML [15] have a direct impact on this research because they focus on modelling applications at the software level.

In [16], the authors propose an interesting approach where they conceive a language that enables engineers to define a family of similar metamodel-based languages (DSM2L) to develop Software product lines (SPLs) as implementations of a MDA.

The standard Interaction Flow Modeling Language (IFML)[6] is designed for expressing the content, user interaction and control behaviour of the front-end of software applications in general. In [17], authors describe how to apply model-driven techniques to the problem of designing the front end of software applications (i.e. the user interaction).

2.1. Related works

As it has been noted in Section 1, model-driven code generation is being increasingly applied to enhance software development from perspectives of maintainability, extensibility and reusability. However, aspect-oriented code generation from models is an area that is currently underdeveloped. In [18] the authors have provided a survey of existing research on aspect-oriented model-driven code generation to discover current work and identify needs for future research. However, the authors have only searched the works in a journal, two conferences and a workshop (between 2002 and 2011), and also, these searches are not focused on MDA works, but more generically to jobs ”model-driven”.

The application of the model driven paradigm to the domain of web software development, is particularly helpful because of the continuous evolution of Web technologies and platforms. In [19] the authors have prepared a survey of primary studies on Model Driven Web Engineering (MDWE) to explore current work and identify needs for future research. Unlike our work, this only focuses on Web Engineering and does not focus on solutions, but also covers other types of less practical jobs (philosophical papers, validation proposals, ...).

Model-driven architecture based testing (MDABT) adopts architectural models of a system under test and/or its environment to derive test artifacts. In [20] the authors have identified the published concerns for applying MDABT, have identified the proposed solutions, and have described the current research directions for MDABT. Unlike our work, it only focuses on MDABT.

Finally, in [21] the authors have undertaken a systematic mapping study of the existing scientific literature in Software & Systems Process Engineering Meta-Model (SPEM), that exploits the benefits of the Model-Driven Architecture paradigm applied to software process models, to discover evidence clusters and evidence deserts in the use and application of SPEM from a business process management point of view. Unlike our work, this one only focuses on SPEM. However, we have made a systematic map focused on MDA, its languages and its areas of application.

[1] http://www.omg.org/mda/
[2] http://www.omg.org/spec/UML/2.5/PDF
[3] http://www.omg.org/spec/MOF/2.5.1/PDF
[4] http://www.omg.org/spec/XMI/2.5.1/PDF
[5] http://www.omg.org/spec/CWM/1.1
[6] http://www.omg.org/spec/IFML/1.0/

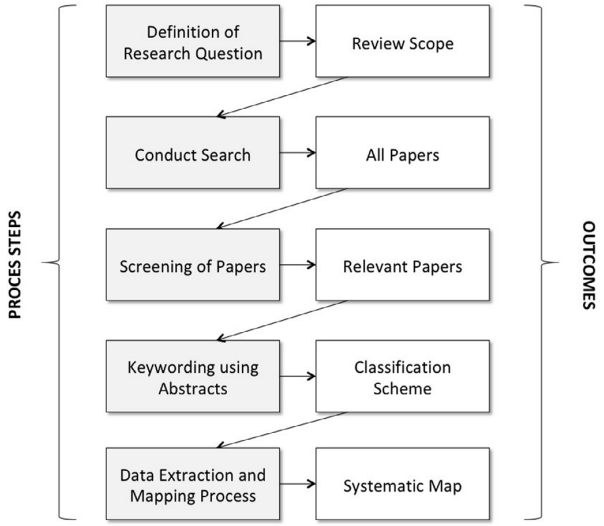


Fig. 1. Steps and outcomes as defined by Petersen et al. [23].

3. Research methodology

The section describes the systematic mapping process applied in this study, which follows the steps defined in [22], which are based on [23]. According to the latter, the systematic mapping process followed in our study is illustrated in Fig. 1 and consists of 5 steps:

- the definition of research questions,
- the search for relevant articles,
- the selection of articles,
- the use of keywords from the abstracts, and finally,
- the extraction of data and the mapping.

Each step of the process has a result, and the final result of the process is the systematic map. After defining the systematic mapping study, the methodology includes 2 final steps: Analysis of the obtained results (presenting the answers to the research questions) (Section 4) and Discussion (a brief discussion on the study) (Section 5).

3.1. Definition of research questions

The main objectives of this study (Step 1), are to (a) establish the body of knowledge of MDA by identifying and categorizing the available research on the topic, (b) identify the most relevant MDA articles in software engineering, (c) extract the reported benefits and challenges of MDA in software engineering, and (d) identify the opportunities for future MDA research. To achieve these research goals, the research questions in the Table 1 have to be answered. With the Research Question 1 we try check whether or not there is a considerable number of works in journals and conferences that apply MDA in various fields of research. With the Research Question 2 we study what the main fields of application of MDA-based works are, focusing on those that generate code. The Research Question 3 is used as a final filtering stage, to stay with those research works that really take the use of MDA to its final objective, which is the generation of code. Finally, with the Research Question 4 we try to identify some research lines or research trends of study offered by MDA.

Table 1
Research questions.

ID	Research question
RQ1	Does MDA remain a topic of interest in research?
RQ2	What are the main fields where MDA is applied successfully?
RQ3	How many MDA articles include code generation?
RQ4	What are the identified research opportunities in MDA?

3.2. Conduct search for primary studies

The next step is to define search strings that have to be developed based on the goal of this study. The main search string in our case will have the following terms: “model driven architecture”. The search will be restricted to the period 2008–2018.

This step also includes the selection of data sources. We are interested in all results from relevant databases for computer science and software engineering. Table 2 presents the specific search strings used in the selected databases.

Table 3 shows the number of articles found in each database. In total, we found 3.241 papers: 617 indexed by ACM, 540 indexed by IEEE, 1.095 indexed by Scopus, and 989 papers indexed by Web of Science.

The retrieved information from each database was stored in a different spreadsheet. Each spreadsheet file contains the relevant information (metadata) from each found article. Some databases allow users to download a CSV file with the search results.

3.3. Screening of papers for inclusion and exclusion (relevant papers)

This step allows us to exclude studies or papers that are not relevant to answer the research questions. The inclusion criteria used was:

- The study should be written in English
- The study should be published between 2008 and December 2018
- The study directly answers one or more of the research questions of this study
- The study should clearly state its focus on MDA in the software engineering domain
- The study should describe the elements and the approach used to implement MDA
- If the study has been published in more than one journal or conference, the most recent version of the study is included.

Articles or papers were excluded when their focus was not MDA or they did not follow a rigorous method. Among the exclusion criteria, we define the following:

- Short papers
- Duplicated articles
- Articles not written in English
- Articles not focused on MDA in the software engineering domain
- Not peer-reviewed articles as book chapters or technical reports

The process of identifying the primary studies that constitute a mapping study is critical to the success of the study. As we saw in the Section 3.2 the search chain was defined in three analogous terms, know, “MDA”, “model driven architecture” and “model-driven architecture”. However, there is still the danger of losing relevant articles. The use of different terminologies in the search string can bias the identification of the primary documents. The search string was used to search keywords, titles and summaries; therefore, the search strategy was recover as many documents related to MDA as possible and closely related contexts (MDE, MDD, etc.).

The results of the searches of articles supposedly related to MDA, released a list of 3241 papers (617 indexed in ACM, 540 indexed in IEEE, 1095 indexed in Scopus, and 989 papers indexed in WOS). In that list of papers, logically there were many duplicates (899 papers) and they were the first items to be discarded. The titles and keywords of the items that are not repeated, and any title that clearly indicated that it

Table 2
Search string used in each database.

Database	Search string
ACM Digital Library	acmdlTitle:(+model +driven +architecture) AND recordAbstract:(+model +driven +architecture)
IEEE Xplore	("model driven architecture") Filters Applied: 2008 - 2019
ISI Web of Science	TS = "model driven architecture" OR TI = "model driven architecture" (SCIENCE TECHNOLOGY) 2008–2018
Scopus	TITLE-ABS-KEY("model driven architecture") AND PUBYEAR > 2008 AND (LIMIT-TO(SUBJAREA, "COMP"))

Table 3
Number of articles found in each database since 2008.

Database	Filter	Number of papers
ACM Digital Library	Conference papers and journal articles	617
IEEE Xplore	Conference papers and journal articles	540
ISI Web of Science	Articles in computer science	989
Scopus	Conference papers and journal articles	1095 (899 duplicates)

was outside the focus of this study was excluded at this stage. For example, we observe that some articles had keywords like: "event-driven architecture", "Event-driven communication", "model-driven security", "event driven architecture", etc. Two articles not written in English and 5 non-scientific articles were discarded as well. Finally, taking into account all criteria for inclusion and Exclusion listed above-, we are left with a ratio of 2145 papers.

We would also like to point out that, if a title did not clearly reveal the application domain of our study, it was included for review in the subsequent steps (this is the case mainly of the articles that talked about Model Driven Engineering or Model-driven Development, because these fields of software engineering encompass MDA, and it may be that such articles do not expressly speak of MDA). At the end of this activity 792 articles that clearly did not speak or used MDA were discarded. However, there were 102 articles recorded as doubtful and pending for further analysis. So, then, two authors read the titles, keywords and summaries of the 102 dubious articles to elucidate their relevance in this mapping study. Finally, they discarded another 86 articles. There were 1267 articles left (see Fig. 2).

Analyzing said 1267 MDA papers, we have observed that they mostly address these fields of application: Web in general (and to these sub-areas in particular: web services, web design, web applications, semantic web), security, testing, simulation, e-learning, IoT, business, GUI design, manufacturing systems (industrial scenarios), serious games, ubiquitous applications, multiagent systems, etc.

In this stage, we have 1267 MDA papers, which are too many to perform a "keywording". With the aim of trying to stay with those that make use of a complete MDA architecture, and are not merely philosophical or theoretical, we are left with the 215 articles that claim to generate source code. In some way, we wanted to make sure that they used MDA "until the end", until the last consequences, so we added a new criterion of exclusion: Discard the papers that in their title, keywords, or abstract do not mention specifically the "generation of code" (executable or not). Based on the title and summary, 1199 papers were excluded. Then, based on a total reading of the 68 papers not excluded so far, 18 other papers were excluded because although these works mention code generation, after checking them, we found out that such proposals did not reach the code generation phase, that is, they did not show code generated, so they can not be considered as eligible. In this way, we are left with 50 papers for the keywording phase (see Fig. 2). The ratio of the 50 primary studies that has been reached after the screening of papers for inclusion and exclusion, is shown in the Appendix A.

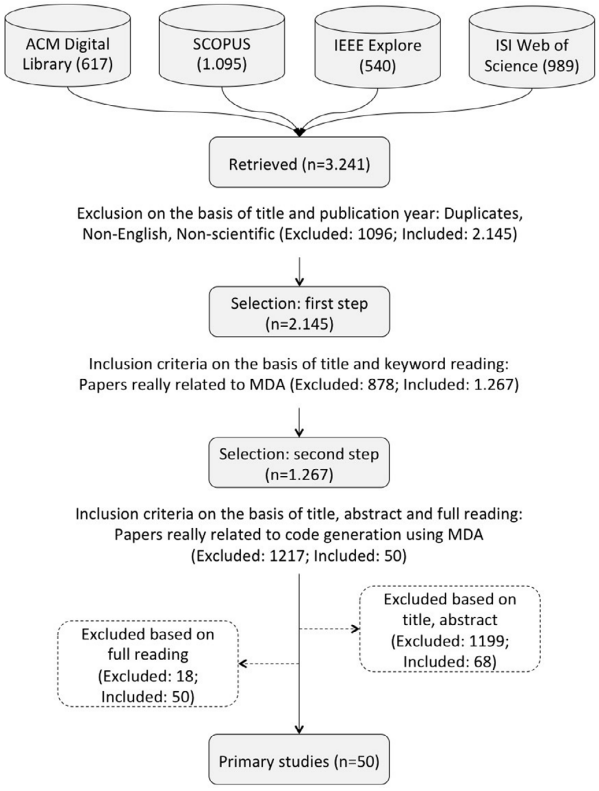


Fig. 2. Screening of Papers for Inclusion and Exclusion (Relevant Papers).

3.4. Characteristics of the primary studies

It is interesting to identify the main channels where the MDA studies (that reach the code generation) are disseminated. Thus, Table 4 shows that 35 of the primary articles were published in international conferences, and that the other 15 were published in journals. Likewise, if we analyze the years of publication of the primary articles, it is possible to point out a slight peak in 2016 and 2017 with 8 articles (see Table 5).

3.5. Keywording of abstracts (classification scheme)

In our study, we have followed the systematic process shown in

Table 4
MDA papers by target conference and journal.

Channel	Title	Primary study
Confer.	International Conference on Model-Driven Engineering and Software Development	[P02], [P12], [P30], [P41]
Confer.	IEEE International Conference on Computer Supported Cooperative Work in Design	[P18], [P22]
Confer.	(another international conferences or workshops)	[P01], [P03], [P04], [P05], [P06], [P07], [P08], [P09], [P10], [P11], [P13], [P14], [P15], [P16], [P17], [P19], [P20], [P21], [P25], [P29], [P32], [P33], [P36], [P40], [P43], [P45], [P46], [P49], [P50]
Journal	International Journal of Electrical and Computer Engineering	[P23]
Journal	International Journal on Software and Systems Modeling	[P24], [P26]
Journal	Intelligent Automation and Soft Computing	[P27]
Journal	Journal of Theoretical and Applied Information Technology	[P28]
Journal	Simulation Modelling Practice and Theory	[P31]
Journal	International Journal of Software Engineering and Its Applications	[P34], [P37], [P38]
Journal	International Review on Computers and Software	[P35]
Journal	SIMULATION: Transactions of The Society for Modeling and Simulation International	[P39]
Journal	Computing and Informatics	[P42]
Journal	The Electronic Library	[P44]
Journal	Journal of Robotics, Networking and Artificial Life	[P47], [P48]

Fig. 3. The elaboration of the keywording is *a way to reduce the time necessary to develop the classification scheme, and also ensures that it takes into account existing studies* [23]. The keywording has been made in two steps:

- First, reviewers read summaries and look for key words and concepts that reflect the contribution of the document. While they are doing it, the reviewers also identify the context of the investigation.
- Second, the set of keywords from different articles combine to develop a high level of understanding about nature and the contribution of the investigation. This helps the reviewers to create a set of categories that is representative of the underlying population. When the summaries are of a quality too low to allow significant keywords to be chosen, reviewers can choose to study also the sections of introduction or conclusion of the document. When a final set of keywords has been chosen, they can be grouped and used to form the categories for the map.

In our study, initially four main facets were created:

- One facet structured the MDA in software engineering (for example, in terms of the MDA architecture, the MDA languages, etc.).
- In another facet, the languages (or types of languages) used to model the different meta-models of the MDA architecture were considered.
- In another third facet we have considered the different fields of application of the research works studied: security, web, decision support, mobile development, simulation, GUI, etc.
- Finally, we considered a fourth facet of research under the title "frameworks." In this facet we have grouped the keywords related to the broad concept frameworks: MDArte framework, AndroMDA framework, toolkit, CASE, methodology, eclipse, etc.

We find these categories easy to interpret and use in the classification, without evaluating each article in detail (as it is done for a systematic review). However, there are keywords generated in the keywording that escaped from these 4 facets (such as "templates", "compile", "MVC", "reengineering"), and that we have left then, outside of any classification. (See Fig. 4).

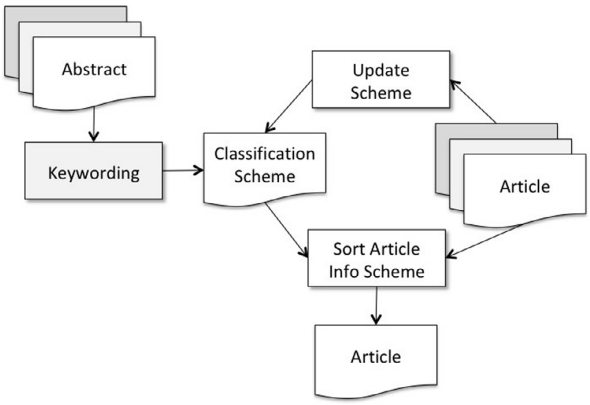


Fig. 3. Building the Classification Scheme.

3.6. Data extraction and mapping of studies (systematic map)

Once we have the classification scheme, the relevant articles are classified in the scheme, that is, the real data extraction is carried out. As shown in Fig. 3, the classification scheme evolves by performing data extraction, such as adding new categories or merging and dividing existing categories. In this step, we created an Excel spreadsheet to document the data extraction process. The spreadsheet shows the number of articles for each category defined in the classification scheme.

From the final table, the frequencies of the publications in each category can be calculated. Finally, for the realization of the Systematic Map, of the four facets or initial categories, we have stayed with three: (1) MDA, (2) Languages and (3) Application areas. The final criteria we

Table 5
Publication by year.

Year	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018
Papers	6	4	3	2	5	3	6	2	8	8	3

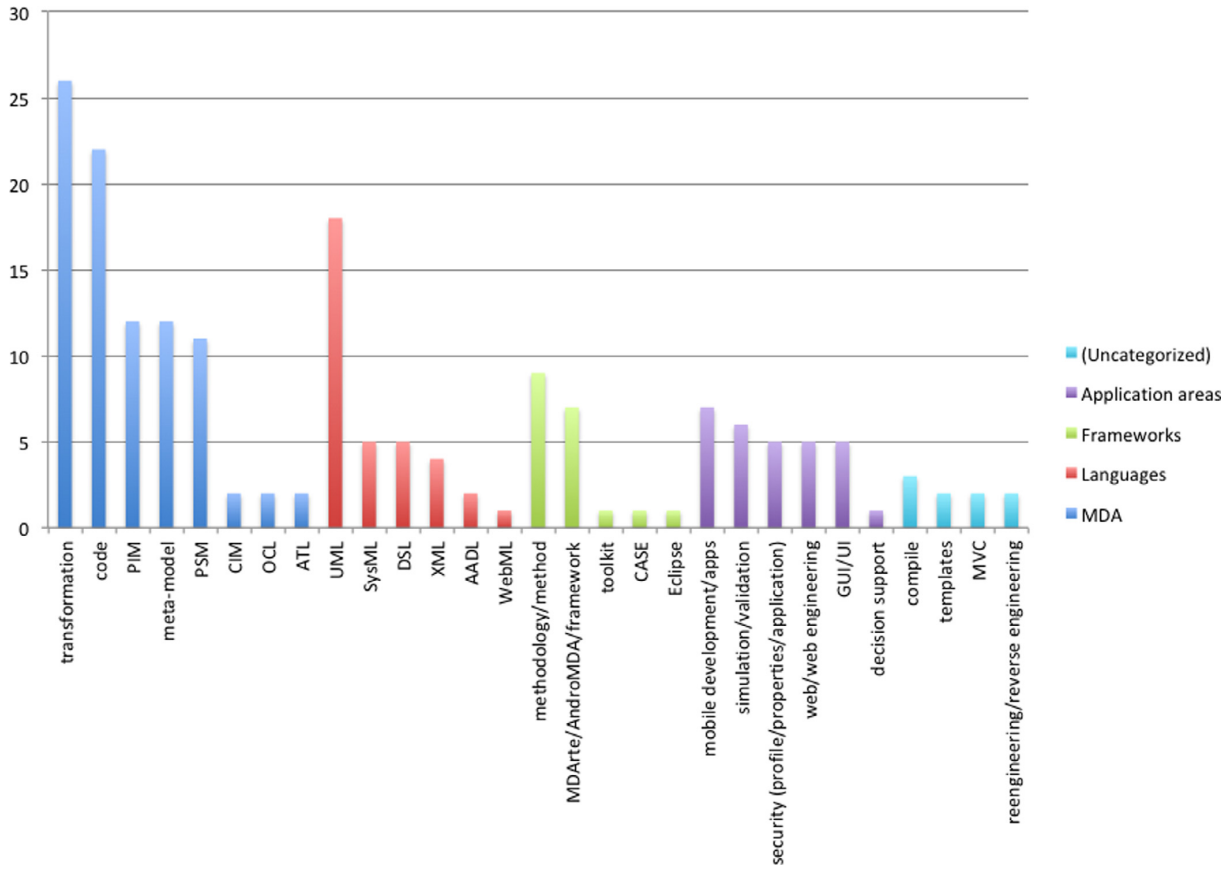


Fig. 4. Keywords grouped by facets.

have adopted in these facets are:

1. **MDA**: Structures the layers of traditional MDA architecture, presented in Table 6.
2. **Languages**: the languages used to model the different meta-models of the layers of the MDA architecture (UML, SysML, WebML, etc.) were considered.
3. **Application areas**: we have considered the different fields of application of the research works studied (security, web, mobile development, simulation, GUI, etc.).

The analysis of the results focuses on presenting the frequencies of the publications for each category. This allows to see which categories have been emphasized in previous research and, therefore, identify gaps and possibilities for future research.

The map aims to illustrate, using statistics summarized in tables, the frequencies of publications in each category. For example, it allows to illustrate which languages (or what type of languages) were used more frequently to model the different meta-models of the different layers of the MDA architecture. For that, the number of articles for each type of language (UML, SysML, XML, WebML, AADL, DSL) were counted, grouping them according to the layers of the MDA architecture (CIM, PIM, PSM) that it modeled. This MDA Architecture facet responds to the definitions we present in Table 6. In our study, we have used a bubble chart to report these frequencies, and it is shown in Fig. 5. It is basically two xy scatter diagrams with bubbles in intersections of categories. The size of a bubble is proportional to the number of items that are in the pair of categories corresponding to the bubble's coordinates. The same idea is used twice, in different quadrants of the same diagram to show the intersection with the third facet. We believe that the bubble chart allows a better analysis than the frequency tables. It is easier to consider different facets simultaneously, and it is also more powerful to give a quick overview of a field and, therefore, to provide a map (see Fig. 5).

Table 6
MDA layer definitions.

Layer	Definition
CIM	Represents independent models of computing (Computationally-Independent Model) that characterize the domain of the problem. This type of models arises above all in processes of business modeling and ideally are conceived before the taking of requirements of a particular application.
PIM	It represents the models that describe a software solution that does not contain details of the concrete platform in which the solution is going to be implemented (Platform-Independent Models). These models arise as a result of analysis and design.
PSM	They are the derived models of the previous category, which contain the details of the platform or technology with which the solution will be implemented (Platform-Specific Models).
Other	(Another MDA layering)

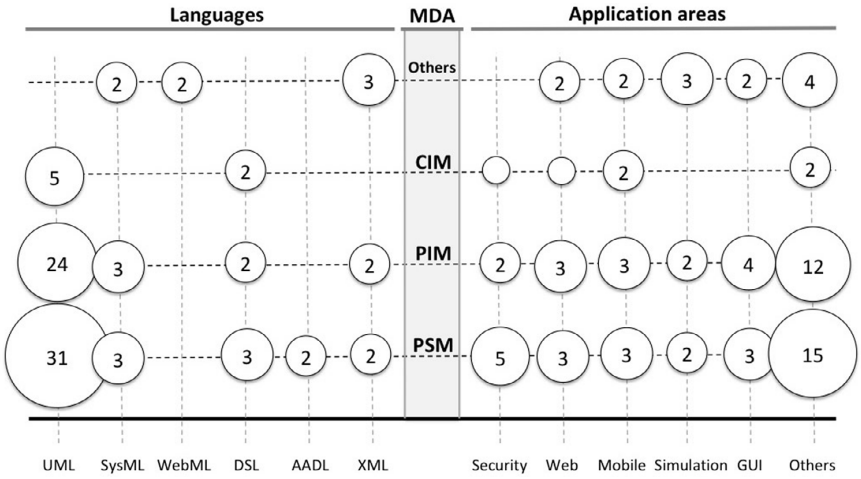


Fig. 5. Visualization of a systematic map in the form of a bubble diagram.

4. Results

This study has allowed us to answer the questions listed in Table 1.

In regard to Research Question 1 *Does MDA remain a topic of interest in research?*, this study shows that (1) there is a considerable number of works (1.267 papers) that claim using MDA, (2) published in a variety of journals, conferences and workshops, (3) referred to many fields of research application, and that (4) this number of publications using MDA seems to be increasing.

As far as Research Question 2 *What are the main fields where MDA is applied successfully?* is concerned, in the Fig. 4 we can see five main areas of interest: mobile application development, simulation and verification, security, Web engineering and GUI development. It can be noted that the number of frameworks based on MDA that are among the primary studies.

With regard to Research Question 3 *How many MDA articles include code generation?* (after following the research methodology described in Section 3), we have reached -in the last few years- at least 50 relevant publications using MDA let us say that " until the end ", until the last consequences, arriving really at the "generation of code" (executable or not) automatically. They are few, but they open a very interesting gap to improve productivity especially in the fields of web development and mobile development.

Regarding to Research Question 4 *What are the identified research opportunities in MDA?* we can conclude that the MDA technology is focused on transformation models, which is not surprising, since MDAs are based on model transformations between models at the same, or different, level of abstraction. Despite of this, we can state that although there is a considerable number of papers regarding PIMs and PSMs; there are few publications regarding CIM showing a research line that may be worth of exploration. Another important point to highlight is the popularity of UML as a modelling language to support MDA technology compared to the use of DSLs. Therefore, the low number of publications related to CIMs can be related to the high popularity of UML which is a software modelling language that may not be the most suitable language to represent models at a high level of abstract such as CIMs. Although there are many toolkits and CASE tools to develop software, this number is relatively low compared to the modelling methods/methodologies. Perhaps, a new research opportunity lies to making the MDA technology (methodologies/methods) more concrete building new development tools. Finally, this study confirms the suitability of MDA to develop complex software (such as mobile applications, simulators, Web application or UIs) that require interoperability of different software technologies (i.e. platforms, protocols, etc.).

5. Discussion

The results of the IEEE and Scopus databases we observe are generally very accurate. However, in the case of ACM it draws attention that not even a third of the results really respond to the required search. In the case of WOS, many more dubious papers appeared than in the other databases, since the keywords do not appear among the results.

Analyzing the 1267 papers of MDA obtained from the first step of selection (Fig. 2), we have observed that they mostly address these fields of application: web in general (these subareas in particular: web services, web design, web applications, semantic web), security, testing, simulation, e-learning, IoT, business, GUI design, manufacturing systems (industrial scenarios), serious games, ubiquitous applications, multiagent systems, etc. Therefore, we can say that MDA is currently used in a wide range of areas.

It is interesting to identify the main channels where the primary studies (MDA papers that arrive at the generation of code) are disseminated. Thus, the Table 4 shows that 35 of the primary articles were published in international conferences, and that the other 15 were published in journals.

From the results of the systematic map (see Fig. 5), it follows that: (a) The language most clearly used for the modeling of the different meta-models of the MDA architecture is UML (followed by DSL and SysML); (b) Few works use in their MDA architecture a CIM layer with an independent models of the computation; (c) Regarding the application area, the papers dedicated to web development, mobile development, GUI and security are the most common.

From the previous analysis, it is possible to identify some research lines or trends of study that can be explored in future research works:

- This study reveals that UML is the preferred modeling language employed in most of the MDA referenced works. However, as exposed in [24], employing a DSL approach could improve the expressiveness of the language to represent models in specific domains. Consequently, there are some open questions in relation to the use of DSLs:
  1. Why do practitioners use UML instead of designing their own DSL?
  2. Are there any advantages of using UML instead of a DSM in terms of code generation?
  3. What are the trade-offs between employing UML profiles and MOF?
- This study also shows that most studied approaches employ PIMs and PSMs only, avoiding the inclusion of CIMs in the model

architecture. As exposed in [1], CIMs leverage the level of abstraction of a MDAs improving model reuse and model expressiveness. Therefore, we could ask the following questions:

1. Why do practitioners avoid the definition of CIM in MDAs generating source code?
2. Are the benefits of employing CIMs in MDAs enough to justify their definition?

- Regarding the Application Area of the selected Primary Studies, it could be interesting to investigate other fields where Model Driven Architectures can be applied successfully. Therefore, we should ask the following questions:
  1. Why MDA approaches are the most popular en these application domains?
  2. Are there any other application domains where MDA can be suitable?
  3. What are there advantages in terms of well-known metrics of using a MDA approach on these application domains respect to traditional approaches?

6. Threats to validity

There are always threats to the validity of a study (compare in [23,25,26]). In this section, we analyze these threats and describe the strategies used to mitigate their effects, as well as the limitations of this study. To assess the validity of this study, the authors have used the validity framework presented by Wohlin et al. [25], which addresses:

- *Construct validity*,
- *External validity*,
- *Internal validity* and
- *Conclusion validity*.

The *validity of construction* is related to the obtaining of the correct measures for the concept that is being studied [23,25,26]. To reduce this threat, a data collection process was designed (Fig. 1) to objectify the selection of items (for example, inclusion and exclusion) and data extraction (Fig. 2) of the 50 main documents to support data recording. To further mitigate this threat, one of the authors of this study acted as an external reviewer to validate the research protocol. Therefore, this threat has been minimized significantly.

The *external validity* is related to the extent to which the results of the study can be generalize [23,25]. To know in what degree the results of a study can be generalized, it is extremely important to describe the context of the research [27,28]. This threat is minimized in this study with a rigorous research methodology that followed the guidelines of [29], and the extraction of data with respect to the methodology (data collection procedures) was carried out following the guidelines of [23] and [30].

The *internal validity* is related to the causal relationships and guarantees that it is not the result of a factor that was not measured or that the researcher did not control. Since the objective of the present study is not to establish a causal statistical relationship on code generation based on MDA, it is not considered a threat for this study.

The *validity of conclusion* is related to the bias of the researchers in the interpretation of these data. Although this risk can not be eliminated, it was reduced by taking the following measures:

- three researchers participated in the analysis of the primary documents,
- a complete “audit” of the recovery of 3241 documents was maintained to identify 50 primary documents,
- as noted above, the 68 relevant articles were read in their entirety by at least two authors, and the conclusions drawn from the analysis of the 50 primary documents involved the three authors.

These four validity threats resonate with the publication bias, which refers to the theme that positive research results are more likely to be published than negative results (compare in [31]). In this case, its effect is minimal because the objective of the study is to present a state of the art of MDA research. However, we recognize that the publication bias could have affected our results with respect to the benefits and challenges of using MDA.

Publication bias can also be affected by the sources of the data in a study and its publication channel. The databases were used: ACM digital library, IEEE Xplore, ISI Web of Science and Scopus, since it is known that these sources return most of the publications and have been used in similar types of literature mapping exercises in software engineering ([20,22,32]).

Although the results of this mapping study are limited by the scientific studies published in these databases, they covered a wide range of MDA publications and closely related contexts (ie, MDE, MDD).

In addition, scientific studies, books, book chapters, short articles, experience reports and assimilation studies not peer-reviewed were excluded. The reason for being to exclude these publications is:

- the data can be anecdotal,
- lack of rigor in the investigation, and
- simulation studies do not reflect the contextual nature in which MDA is used.

Recently, a different approach (systematic mapping study driven by margin of error) has been proposed [33]. The authors introduce statistics with random sampling and a margin of error into the design, reducing the cost of selecting primary studies. The search step can also be improved by including the method proposed by Zhang et al. [34], which they applied in designing Systematic Literature Reviews.

7. Conclusions

This systematic mapping study provides a structured understanding of the state-of-the-art of code generation using MDA research in software engineering. This was achieved by identifying 50 primary studies out of 2.145 related MDA articles over a ten years period (2008–2018) and analysed them with respect to (i) frequency of publication by year, (ii) publication channels, (iii) application area, (iv) MDA layers, (v) modeling languages, and (vi) frameworks used.

In this work we have analyzed the threats to the validity of this study and have described the strategies used to mitigate its effects, as well as the limitations of this study.

For the realization of this mapping study, the following databases have been used: ACM digital library, IEEE Xplore, ISI Web of Science and Scopus, since it is known that these sources return most of the publications.

From the different stages of this study it is clear that:

- There is a considerable number of works that use MDA, published in a great variety of journals, congresses and workshops, referring to a wide variety of fields of application, and this number of publications seems to be increasing.
- The majority of the MDA papers obtained from the first selection step are directed to these fields of application: web in general, security, testing, simulation, e-learning, IoT, business, GUI design, manufacturing systems, serious games, ubiquitous applications, and multiagent systems. Because of this, we can say that MDA is currently used in a wide range of areas
- Of the 50 primary articles, 35 were published in international conferences, and the other 15 were published in journals.

In particular, the results of the systematic map reflected in Figure 5, show that: (a) The language most clearly used for the modeling of the different meta-models of the MDA architecture is UML (followed by DSL and SysML); (b) Few works use in their MDA architecture a CIM

layer with an independent models of the computation; (c) Few jobs use an MDA architecture different from the classic CIM, PIM, PSM; (d) Regarding the facet of the application area, the papers dedicated to web development, mobile development, GUI and security continue to stand out.

There is still a long way to go in the field of MDA, and -in many cases- the automatic generation of code from models is still a software engineers' dream, and the development and subsequent publication of research works that use MDA to generate code is still complicated.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Primary studies

P01. L. Abdellatif, M. Chhiba, O. Mjihil, Deals with integrating of security specifications during software design phase using mda approach, in: Proceedings of the Second International Conference on Internet of Things, Data and Cloud Computing, ICC '17, ACM, New York, NY, USA, 2017, pp. 196:1–196:7. doi:10.1145/3018896.3065835

P02. R. S. Monteiro, G. Zimbrão, J. M. de Souza, Exchanging solutions for information systems development using a model pattern perspective: Diagram templates in the context of the mdarte collaborative evolution process, in: 2016 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD), 2016, pp. 640–647.

P03. A. Puder, S. Haeberling, R. Todtenhoefer, An mda approach to byte code level cross-compilation, in: 2008 Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2008, pp. 251–256. doi:10.1109/SNPD.2008.109.

P04. S. N. Karimi, S. Parsa, Semi-automatic transformation of sequential code to distributed code using model driven architecture approach, in: 2009 IEEE International Symposium on Parallel and Distributed Processing with Applications, 2009, pp. 708–714. doi:10.1109/ISPA.2009.71.

P05. A. Sabraoui, M. E. Koutbi, I. Khriss, Gui code generation for android applications using a mda approach, in: 2012 IEEE International Conference on Complex Systems (ICCS), 2012, pp. 1–6. doi:10.1109/ICoCS.2012.6458567.

P06. M. Lettner, M. Tschernuth, Applied mda for embedded devices: Software design and code generation for a low-cost mobile phone, in: 2010 IEEE 34th Annual Computer Software and Applications Conference Workshops, 2010, pp. 63–68. doi:10.1109/COMPSACW.2010.21.

P07. G. Zhuang, J. Du, Mda-based modeling and implementation of e-commerce web applications in webml, in: 2009 Second International Workshop on Computer Science and Engineering, Vol. 2, 2009, pp. 507–510. doi:10.1109/WCSE.2009.863.

P08. S. Roubi, M. Erramdani, S. Mbarki, Modeling and generating graphical user interface for mvc rich internet application using a model driven approach, in: 2016 International Conference on Information Technology for Organizations Development (IT4OD), 2016, pp. 1–6. doi:10.1109/IT4OD.2016.7479249.

P09. L. Ma, S. Gui, L. Luo, L. Yin, Research of automatic code generating technology based on aadl, in: 2008 International Conference on Embedded Software and Systems Symposia, 2008, pp. 136–141. doi:10.1109/ICESS.Symposia.2008.49.

P10. B. L. Romano, G. Braga e Silva, A. Marques da Cunha, W. I. Mourão, Applying mda development approach to a hydrological project, in: 2010 Seventh International Conference on Information Technology: New Generations, 2010, pp. 1127–1132. doi:10.1109/ITNG.2010.121.

P11. D. Perillo, M. D. Natale, An mda approach for the generation of glue code for a virtual simulation environment, in: Proceedings of the 9th IEEE International Symposium on Industrial Embedded Systems (SIES 2014), 2014, pp. 1–4. doi:10.1109/SIES.2014.7087456.

P12. S. Jäger, R. Maschotta, T. Jungebloud, A. Wichmann, A. Zimmermann, An emf-like uml generator for c++, in: 2016 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD), 2016, pp. 309–316.

P13. Z. Yuyan, H. Guodong, F. Wentang, H. Jinkui, A model-driven development approach for web report system, in: 2008 27th Chinese Control Conference, 2008, pp. 708–712. doi:10.1109/CHICC.2008.4605405.

P14. H. Benouda, M. Azizi, M. Moussaoui, R. Esbai, Automatic code generation within mda approach for cross-platform mobiles apps, in: 2017 First International Conference on Embedded Distributed Systems (EDiS), 2017, pp. 1–5. doi:10.1109/EDIS.2017.8284045.

P15. O. El Beggar, B. Bousetta, T. Gadi, Generating methods signatures from transition state diagram: A model transformation approach, in: 2012 Colloquium in Information Science and Technology, 2012, pp. 4–9. doi:10.1109/CIST.2012.6388054.

P16. K. Mizuoka, M. Koga, Mda development of manufacturing execution system based on automatic code generation, in: Proceedings of SICE Annual Conference 2010, 2010, pp. 3103–3106.

P17. A. Gonzalez, M. Uva, M. Frutos, Refactoring java code by transformation rules in qvt-relation, in: 2013 XXXIX Latin American Computing Conference (CLEI), 2013, pp. 1–9.doi:10.1109/CLEI.2013.6670658.

P18. R. E. A. Pinel, R. Monteiro, G. Silva, J. Souza, Collaborative support embedded in information system through automatic code generation, in: IEEE 16th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2012, May 23–25, 2012, Wuhan, China, 2012, pp. 328–333. doi:10.1109/CSCWD.2012.6221839.

P19. M. Brun, J. Delatour, Y. Trinquet, Code generation from aadl to a real-time operating system: An experimentation feedback on the use of model transformation, in: 13th IEEE International Conference on Engineering of Complex Computer Systems (iceccs 2008), 2008, pp. 257–262. doi:10.1109/ICECCS.2008.19.

P20. M. Lachgar, A. Abdali, Dsl and code generator for accelerating ios apps development, in: 2015 Third World Conference on Complex Systems (WCCS), 2015, pp. 1–8. doi:10.1109/ICoCS.2015.7483269.

P21. J. d. Monte-Mor, E. O. Ferreira, H. F. Campos, A. M. da Cunha, L. A. V. Dias, Applying mda approach to create graphical user interfaces, in: 2011 Eighth International Conference on Information Technology: New Generations, 2011, pp. 766–771. doi:10.1109/ITNG.2011.206.

P22. R. S. Monteiro, G. Zimbrão, J. M. de Souza, Collaborative evolution process in mdarte: Exchanging solutions for information systems development among projects, in: Proceedings of the 2014 IEEE 18th International Conference on Computer Supported Cooperative Work in Design (CSCWD), 2014, pp. 569–574. doi:10.1109/CSCWD.2014.6846907.

P23. L. Abdellatif, M. Chhiba, T. Abdelmoumen, A uml profile for security and code generation, International Journal of Electrical and Computer Engineering (IJECE) 8 (2018) 5278. doi:10.11591/ijece.v8i6.pp5278-5291.

P24. M. Natale, D. Perillo, F. Chirico, A. Sindico, A. Sangiovanni-Vincentelli, A model-based approach for the synthesis of software to firmware adapters for use with automatically generated components, International Journal on Software and Systems Modeling 17 (1) (2018) 11–33. doi:10.1007/s10270-016-0534-0.

P25. L. Abdellatif, M. Chhiba, A. Tabyaoui, O. Mjihil, Mda approach for

application security integration with automatic code generation from communication diagram, in: G. Noreddine, J. Kacprzyk (Eds.), International Conference on Information Technology and Communication Systems, Springer International Publishing, Cham, 2018, pp. 297–310.

P26. T. Rodrigues, F. C. Delicato, T. Batista, P. F. Pires, L. Pirmez, An approach based on the domain perspective to develop wsan applications, International Journal on Software and Systems Modeling 16 (4) (2017) 949–977. doi:10.1007/s10270-015-0498-5.

P27. M. Lachgar, A. Abdali, Modeling and generating native code for cross-platform mobile applications using dsl, Intelligent Automation & Soft Computing 23 (3) (2017) 445–458. doi:10.1080/10798587.2016.1239392.

P28. L. Abdellatif, M. Chhiba, T. Abdelmoumen, O. Mjihil, Mdaasi: Model driven architecture approach for application security integration, Journal of Theoretical and Applied Information Technology 95 (2017) 1655–1668.

P29. O. Betari, M. Erramdani, S. Roubi, K. Arrhioui, S. Mbarki, Model transformations in the mof meta-modeling architecture: From uml to codeigniter php framework, in: Á. Rocha, M. Serrhini, C. Felgueiras (Eds.), Europe and MENA Cooperation Advances in Information and Communication Technologies, Springer International Publishing, Cham, 2017, pp. 227–234.

P30. M. Overeem, S. Jansen, An exploration of the 'it' in 'it depends': Generative versus interpretive model-driven development, in: Conference on Model-Driven Engineering and Software Development (MODELSWARD), 2017.

P31. A. Tsadimas, G.-D. Kapos, V. Dalakas, M. Nikolaidou, D. Anagnostopoulos, Simulating simulation-agnostic sysml models for enterprise information systems via devs, Simulation Modelling Practice and Theory 66 (2016) 243 – 259. doi:https://doi.org/10.1016/j.simpat.2016.04.001.

P32. S. Gotti, S. Mbarki, Toward ifvm virtual machine: A model driven ifml interpretation, in: Proceedings of the 11th International Joint Conference on Software Technologies - Volume 1: ICSOFT-EA, (ICSOFT 2016), INSTICC, SciTePress, 2016, pp. 220–225. doi:10.5220/0005986102200225.

P33. Z. Morales, C. Magańa, J. A. Aguilar, A. Zaldívar-Colado, C. Tripp-Barba, S. Misra, O. Garcia, E. Zurita, A baseline domain specific language proposal for model-driven web engineering code generation, in: O. Gervasi, B. Murgante, S. Misra, A. M. A. Rocha, C. M. Torre, D. Taniar, B. O. Apduhan, E. Stankova, S. Wang (Eds.), Computational Science and Its Applications – ICCSA 2016, Springer International Publishing, Cham, 2016, pp. 50–59.

P34. H. Benouda, R. ESSBAI, M. Azizi, M. MOUSSAOUI, Modeling and code generation of android applications using acceleo, International Journal of Software Engineering and Its Applications 10 (2016) 83–94. doi:10.14257/ijseia.2016.10.3.08.

P35. H. Benouda, M. Azizi, R. Esbai, M. Moussaoui, Code generation approach for mobile application using acceleo, International Review on Computers and Software (IRECOS) 11 (2016) 160. doi:10.15866/irecos.v11i2.8480.

P36. M. Di Natale, F. Chirico, A. Sindico, A. Sangiovanni-Vincentelli, An mda approach for the generation of communication adapters integrating sw and fw components from simulink, in: J. Dingel, W. Schulte, I. Ramos, S. Abrahão, E. Insfran (Eds.), Model-Driven Engineering Languages and Systems, Springer International Publishing, Cham, 2014, pp. 353–369.

P37. M. Rahmouni, S. Mbarki, Model-driven generation: From models to mvc2 web applications, International Journal of Software Engineering and Its Applications 887 (2014) 73–94. doi:10.14257/ijseia.2014.8.7.07.

P38. N. Methakullawatm, Y. Limpiyakorn, Reengineering legacy code with model transformation, International Journal of Software Engineering and Its Applications 8 (3) (2014) 97–110. doi:10.14257/ijseia.2014.8.3.10.

P39. G.-D. Kapos, V. Dalakas, M. Nikolaidou, D. Anagnostopoulos, An integrated framework for automated simulation of sysml models using devs, SIMULATION: Transactions of The Society for Modeling and Simulation International 90. doi:10.1177/0037549714533842.

P40. C.-P. Hwang, M.-S. Chen, The uml diagram to vhdl code transformation based on mda methodology, in: Y. Tan, Y. Shi, H. Mo (Eds.), Advances in Swarm Intelligence, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 496–503.

P41. G. Sedrakyan, M. Snoeck, A pim-to-code requirements engineering framework, in: MODELSWARD 2013 - Proceedings of the 1st International Conference on Model-Driven Engineering and Software Development, SciTePress, 2013, pp. 163–169.

P42. N. Obrenović, A. Popovic, S. Kordić (Aleksić, I. Luković, Transformations of check constraint pim specifications, Computing and Informatics 31 (2012) 1045–1079.

P43. A. Saurabh, D. Dahiya, R. Mohana, Maximizing automatic code generation: Using xml based mda, in: M. Parashar, D. Kaushik, O. F. Rana, R. Samtaney, Y. Yang, A. Zomaya (Eds.), Contemporary Computing, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 283–293.

P44. G. Milosavljevic, D. Ivanovic, D. Surla, B. Milosavljević, Automated construction of the user interface for a cerif-compliant research management system, The Electronic Library 29 (2011) 565–588. doi:10.1108/02640471111177035.

P45. L. Favre, L. Martinez, C. Pereira, Mda-based reverse engineering of object oriented code, in: T. Halpin, J. Krogstie, S. Nurcan, E. Proper, R. Schmidt, P. Soffer, R. Ukor (Eds.), Enterprise, Business-Process and Information Systems Modeling, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 251–263.

P46. Y.-C. Huang, C.-P. Chu, Z.-A. Lin, M. Matuschek, Transformation from web psm to code, in: Proceedings: DMS 2009 - 15th International Conference on Distributed Multimedia Systems, 2009, pp. 16–19.

P47. T. Katayama, Y. Kikkawa, Y. Kita, H. Yamaba, K. Aburada, N. Okazaki, Development of a tool to keep consistency between a model and a source code in software development using mda, Journal of Robotics, Networking and Artificial Life 3 (2017/02) 231–235. doi:https://doi.org/10.2991/jrnal.2017.3.4.5.

P48. T. Katayama, Y. Kikkawa, Y. Kita, H. Yamaba, K. Aburada, N. Okazaki, Proposal of a modification method of a source code to correspond with a modified model in mda, Journal of Robotics, Networking and Artificial Life 2 (2015/04) 135–139. doi:https://doi.org/10.2991/jrnal.2015.2.2.15.

P49. C. Blanco, I. García-Rodríguez De Guzmán, E. Fernández-Medina, J. Trujillo, M. Piattini, Automatic generation of secure multi-dimensional code for data warehouses: An mda approach, in: Proceedings of the OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008. Part II on On the Move to Meaningful Internet Systems, OTM '08, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 1052–1068.

P50. C. Blanco, I. García-Rodríguez De Guzmán, E. Fernández-Medina, J. Trujillo, M. Piattini, Obtaining secure code in sql server analysis services by using mda and qvt, in: International Workshop on Security in Information Systems (WOSIS), INSTICC, SciTePress, 2008, pp. 38–48. doi:10.5220/0001745200380048.

References

[1] S. Mellor, K. Scott, A. Uhl, D. Weise, MDA Distilled: Principles of Model-Driven Architecture, Addison Wesley Professional, Boston, MA, USA, 2004.

[2] A.G. Kleppe, J. Warmer, W. Bast, MDA Explained: The Model Driven Architecture: Practice and Promise, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[3] G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, L. Bouillon, J. Vanderdonckt, A Unifying Reference Framework for Multi-target User Interfaces (3) (2003) 289–308.

[4] S. Feuerstack, E.B. Pizzolato, Engineering device-spanning, multimodal web applications using a model-based design approach, in: G. Bressan, R.M. Silveira, E.V. Munson, A. Santanchà, M. da Graça Campos Pimentel (Eds.), WebMedia, ACM, 2012, pp. 29–38. http://dl.acm.org/citation.cfm?id=2382636
[5] S. Conn, L. Forrester, Model driven architecture: a research review for information systems educators teaching software development, 4 (2006).
[6] Z. Bizonova, Model driven e-learning platform integration, in: K. Maillet, T. Klobucar, D. Gillet, R. Klamma (Eds.), Proceedings of the EC-TEL 2007 PROLEARN Doctoral Consortium, Crete, Greece, September 18, 2007, CEUR Workshop Proceedings, 288 CEUR-WS.org, 2007.
[7] N. Lahiani, D. Bennouar, A model driven approach to derive e-learning applications in software product line, Proceedings of the International Conference on Intelligent Information Processing, Security and Advanced Communication (IPAC 15), (2015), https://doi.org/10.1145/2816839.2816850.
[8] N. Koch, A. Kraus, Towards a Common Metamodel for the Development of Web Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 497–506. 10.1007/3-540-45068-8_92.
[9] P. Blumschein, W. Hung, D. Jonassen, J. Strobel, Model-Based Ap- proaches to Learning: Using Systems Models and Simulations to Improve Understanding and Problem Solving in Complex Domains, Brill, 2009, https://doi.org/10.1163/9789087907112.
[10] Y. Tian, H. Yang, L. Landy, MDA-based development of music-learning system, in: S. Zhang, D. Li (Eds.), Proceedings of the 14th Chinese Automation & Computing Society Conference, UK, Brunel University, West London, 2008, pp. 97–102.
[11] R. Dehbi, M. Talea, A. Tragha, A model driven methodology approach for e-learning platform development, Int. J. Inf. Edu. Technol. 3 (1) (2013) 10–15.
[12] S. Tang, M. Hanneghan, A model-driven framework to support development of serious games for game-based learning, Developments in E-systems Engineering, (2010), pp. 95–100.
[13] J. Conallen, Building web Applications with UML, 2nd, Addison Wesley, Reading, Massachusetts, 2002.
[14] T. Isakowitz, E.A. Stohr, P. Balasubramanian, RMM: A methodology for structured hypermedia design, Commun. ACM 38 (8) (1995) 34–44.
[15] S. Ceri, P. Fraternali, A. Bongio, Web modeling language (webML): A modeling language for designing web sites, Proceedings of the Ninth International World Wide Web Conference, Elsevier, Amsterdam, Netherlands, 2000.
[16] L. Samimi-Dehkordi, B. Zamani, S. Kolahdouz-Rahimi, Leveraging product line engineering for the development of domain-specific metamodeling languages, J. Comput. Lang. 51 (2019) 193–213, https://doi.org/10.1016/j.cola.2019.02.006. http://www.sciencedirect.com/science/article/pii/S1045926X1830212X
[17] M. Brambilla, P. Fraternali, Interaction flow modeling language: Model-Driven UI engineering of web and mobile apps with IFML, 1st, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2014.
[18] A. Mehmood, D.N. Jawawi, Aspect-oriented model-driven code generation: a systematic mapping study, Inf. Softw. Technol. 55 (2) (2013) 395–411, https://doi.org/10.1016/j.infsof.2012.09.003. Special Section: Component-Based Software Engineering (CBSE), 2011
[19] K. Wakil, D.N.A. Jawawi, Model driven web engineering: a systematic mapping study, e-Informatica Softw. Eng. J. 9 (2015) 107–142, https://doi.org/10.5277/E-INF150106.
[20] B. Uzun, B. Tekinerdogan, Model-driven architecture based testing: a systematic literature review, Inf. Softw. Technol. 102 (2018) 30–48, https://doi.org/10.1016/j.infsof.2018.05.004.
[21] I. Ruiz-Rube, J.M. Dodero, M. Palomo-Duarte, M. Ruiz, D. Gawn, Uses and applications of software & systems process engineering meta-model process models. a systematic mapping study, J. Softw. Evolut. Process 25 (9) (2013) 999–1025, https://doi.org/10.1002/smr.1594.
[22] M.O. Ahmad, D. Dennehy, K. Conboy, M. Oivo, Kanban in software engineering: a systematic mapping study, J. Syst. Softw. 137 (2018) 96–113, https://doi.org/10.1016/j.jss.2017.11.045.
[23] K. Petersen, S. Vakkalanka, L. Kuzniarz, Guidelines for conducting systematic mapping studies in software engineering: an update, Inf. Softw. Technol. 64 (2015) 1–18, https://doi.org/10.1016/j.infsof.2015.03.007.
[24] J. Bézivin, H. Bruneliere, F. Jouault, I. Kurtev, Model engineering sup- port for tool interoperability, in: Workshop in Software Model Engineering (WiSME'2005) - a MODELS 2005 Satellite Event, HAL CCSD, Montego Bay, Jamaica, 2005 URL https://hal.inria.fr/hal-01272245.
[25] C. Wohlin, P. Runeson, M. Hst, M.C. Ohlsson, B. Regnell, A. Wessln, Experimentation in Software Engineering, Springer Publishing Company, Incorporated, 2012.
[26] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empir. Softw. Eng. 14 (2) (2008) 131, https://doi.org/10.1007/s10664-008-9102-8.
[27] K. Petersen, C. Wohlin, A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case, J. Syst. Softw. 82 (9) (2009) 1479–1490, https://doi.org/10.1016/j.jss.2009.03.036. SI: QSIC 2007
[28] B.A. Kitchenham, D. Budgen, O.P. Brereton, Using mapping studies as the basis for further research - a participant-observer case study, Inf. Softw. Technol. 53 (6) (2011) 638–651, https://doi.org/10.1016/j.infsof.2010.12.011. Special Section: Best papers from the APSEC
[29] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic mapping studies in software engineering, Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering, EASE'08, BCS Learning & Development Ltd., Swindon, UK, 2008, pp. 68–77.
[30] T. Dybå, T. Dingsøyr, Empirical studies of agile software development: a systematic review, Inf. Softw. Technol. 50 (9) (2008) 833–859, https://doi.org/10.1016/j.infsof.2008.01.006.
[31] M. Unterkalmsteiner, T. Gorschek, A.K.M.M. Islam, C.K. Cheng, R.B. Permadi, R. Feldt, Evaluation and measurement of software process improvement - a systematic literature review, IEEE Trans. Software Eng. 38 (2) (2012) 398–424, https://doi.org/10.1109/TSE.2011.26.
[32] B. Kitchenham, P. Brereton, A systematic review of systematic review process research in software engineering, Inf. Softw. Technol. 55 (12) (2013) 2049–2075, https://doi.org/10.1016/j.infsof.2013.07.010.
[33] T. Kosar, S. Bohra, M. Mernik, A systematic mapping study driven by the margin of error, J. Syst. Softw. 144 (2018) 439–449, https://doi.org/10.1016/j.jss.2018.06.078. http://www.sciencedirect.com/science/article/pii/S0164121218301353
[34] H. Zhang, M.A. Babar, P. Tell, Identifying relevant studies in software engineering, Inf. Softw. Technol. 53 (6) (2011) 625–637, https://doi.org/10.1016/j.infsof.2010.12.010. Special Section: Best papers from the APSEC

Capítulo 4

Modeling Language-Learning Applications

Citación: G. Sebastian, R. Tesoriero, J. A. Gallud, Modeling language-learning applications, IEEE Latin America Transactions 15 (9) (2017) 17711776. doi:10.1109/TLA.2017.8015084.

Tipo de publicación: Revista Internacional (IF: 0.804, Q4)

Esta publicación avala esta disertación por compendio de publicaciones.

IEEE LATIN AMERICA TRANSACTIONS, VOL. 15, NO. 9, SEPTEMBER 2017 1771

Modeling Language-Learning Applications

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

***Abstract*— Nowadays, more and more people are interested in learning a second and even a third foreign language due to the globalization phenomenon and the extensive use of Internet. The process of learning a foreign language is defined by methodologies and supported by technology. The development of these kind of applications is complex, so that this article proposes a model-driven approach to develop software to support different language learning processes. The article describes a meta-model that defines the entities and its relationships. This meta-model allows us to support different methods to learn a second language.**

***Keywords*— Model-driven development, languages learning methodologies and apps.**

I. INTRODUCCIÓN

EL INTERÉS por estudiar un idioma extranjero está aumentando en los últimos tiempos debido al fenómeno de la globalización y al uso generalizado de Internet como plataforma para la proyección tanto personal como de las empresas.

No hay más que ver cómo crece el número de estudiantes en escuelas y academias de enseñanza de idiomas, o el aumento de la oferta bilingüe en centros educativos a todos los niveles (primaria, secundaria y universidad). También es interesante señalar la numerosa oferta de diferentes métodos para aprender idiomas.

La tecnología no ha sido ajena a este interés de la gente por el aprendizaje de idiomas. Son también numerosas las aplicaciones y herramientas relacionadas con este campo, que se ofrecen al usuario. Con el rápido crecimiento del uso de smartphones, las aplicaciones móviles se están convirtiendo, junto a la Web, en uno de los ámbitos preferidos por los usuarios para el aprendizaje de un idioma extranjero.

El desarrollo de este tipo de aplicaciones es complejo, por la diversidad de metodologías para el aprendizaje de idiomas que existen, por la diversidad de entornos de ejecución posibles (Web, móvil y escritorio) y por la diversidad de tecnologías que se pueden utilizar para desarrollarlas. Por ello, resulta muy oportuno aplicar la solución del Desarrollo Dirigido por Modelos (MDD) y la Arquitectura Dirigida por Modelos (MDA), pues puede ahorrar costes en el desarrollo de aplicaciones complejas multi-plataforma.

Este artículo presenta un primer resultado en esta dirección. Se trata de un meta-modelo que modela las características principales que están presentes en la mayoría de métodos para el aprendizaje de un idioma extranjero.

El artículo está organizado en las siguientes secciones. En la sección II se recorre el contexto de la investigación junto con los trabajos relaciones. En la sección III se presenta el meta-modelo propuesto, incluyendo una descripción de los grandes bloques de que se compone. La sección IV muestra un ejemplo para ilustrar los conceptos. En la sección V se presenta un análisis de otras capacidades de modelado a partir del meta-modelo propuesto. Finalmente se presentan las conclusiones y el trabajo futuro.

II. CONTEXTO DE LA INVESTIGACIÓN

El contexto de la investigación, objeto de esta sección, tiene dos elementos importantes. En primer lugar, se presentan los conceptos fundamentales relaciones con los métodos para aprender idiomas, características generales y algunos ejemplos de métodos que se estudian en este artículo. En segundo lugar, se presentan trabajos relacionados con el desarrollo dirigido por modelos.

El aprendizaje de idiomas es un ámbito en el que proliferan los métodos, técnicas y herramientas, a cada cual más prometedora. En este trabajo nos hemos centrado en aquellas metodologías que ofrecen alguna clase de soporte tecnológico (sitio Web, app móvil o similar).

Entre las metodologías que fueron objeto de estudio se encuentran Lexiway [1], Duolingo [2], Babbel [3] y Busuu [4]. Lexiway es una metodología novedosa que pone un mayor énfasis en la pronunciación, en el vocabulario y en la composición de frases (y no tanto en la gramática). Lexiway cuenta con un buen apoyo de herramientas: un aula virtual y una aplicación iOS. Duolingo es un método para aprender inglés que ha surgido con la profusión de apps móviles. Este método da una mayor importancia a la motivación de los usuarios, para lo cual recurre a la gamificación [8]. Babbel es una metodología que da mucha importancia a que el alumno amplíe su vocabulario; con cada ejercicio, detecta sus puntos fuertes y débiles, y va personalizándo cada lección. Babbel tiene la virtud de conseguir que la gramática se aprenda intuitivamente, casi sin que el alumno se de cuenta. Busuu es un método que mejora la comprensión lectora, el diálogo y la escritura de los alumnos con una gran variedad de ejercicios. El entorno de aprendizaje de Busuu, permite además a sus alumnos interactuar con su propia comunidad internacional de hablantes nativos.

Los enfoques que ofrecen las distintas metodologías son muy diversos, aunque es posible identificar elementos comunes. El enfoque del desarrollo basado en modelos se encargará de abstraer los elementos presentes en los diferentes métodos. Pero antes necesitamos estudiar algunas

G. Sebastián, Universidad de Castilla-La Mancha (UCLM), Albacete, España, gabriel.sebastian@uclm.es

R. Tesoriero, Universidad de Castilla-La Mancha (UCLM), Albacete, España, ricardo.tesoriero@uclm.es

J. A. Gallud, Universidad de Castilla-La Mancha (UCLM), Albacete, España, jose.gallud@uclm.es

(*Corresponding autor: G. Sebastián)*

metodologías, las más relevantes, en un campo concreto como es el aprendizaje de un idioma.

El concepto de MDA fue propuesto por el Object Management Group (OMG) en 2001. En MDA Guide [7], MDA se define como una solución para utilizar modelos en el desarrollo software. Esta solución proporciona un mayor protagonismo a los modelos en los sistemas que se van a desarrollar. Como el propio nombre indica, se dice dirigido por modelos porque cubre las etapas de concepción, diseño, construcción, desarrollo, mantenimiento y modificación.

En el núcleo de MDA hay una serie de estándares importantes de OMG: Unified Modeling Language (UML), la Meta Object Facility (MOF), XML Metadata Interchange (XMI) y la Common Warehouse Metamodel (CWM). Estos estándares definen la infraestructura central de MDA, que se ha utilizado con éxito en el modelado y desarrollo de muchos sistemas modernos.

En el ámbito de la Interacción Persona-Ordenador se puede encontrar un enfoque que utiliza modelos para obtener la interfaz de usuario (Desarrollo Basado en Modelos de Interfaces de Usuario MB-UID) [5]. En este sentido, este trabajo se propone el desarrollo de sistemas interactivos, con lo cual comparte objetivos y enfoque con aquellos que utilizan MB-UID.

Muy pocos dudan del protagonismo que tendrá el MDD en el futuro cercano del desarrollo software, a medida que los desarrolladores cuenten con herramientas cada vez más sofisticadas. Quizá por eso no se ha encontrado en la bibliografía trabajos previos que formalicen mediante un meta-modelo una metodología para el aprendizaje de idiomas.

Son numerosos los trabajos relacionados que utilizan un enfoque MDD o MDA en el desarrollo de aplicaciones Web [11][12][13]. En [6] se presenta un ejemplo de aplicación de una MDA para desarrollar una aplicación para el aprendizaje de música.

Una metodología para el modelado de aplicaciones de e-learning se puede ver en [9][10]. Esta metodología generalista para el desarrollo de aplicaciones de e-learning no está enfocada al desarrollo de aplicaciones para el aprendizaje de idiomas. Además, nuestra propuesta presenta un conjunto de modelos a diferentes niveles de abstracción (arquitectura), los cuales muestran diferentes puntos de vista de la aplicación dependiendo del nivel de abstracción.

III. METAMODELO

El estudio y análisis de las diferentes metodologías para el aprendizaje de idiomas nos permitió obtener, mediante un proceso de abstracción, los elementos comunes a todas ellas. En esta sección presentamos estos elementos y su formalización utilizando un meta-modelo ECORE con restricciones en Object Constraint Language (OCL).

Veamos en primer lugar los elementos comunes. Las metodologías analizadas manejan un conjunto de recursos para presentar un concepto de aprendizaje (contenidos). Los contenidos tienen una representación visual (gráfico o video), y pueden estar agrupados de diferente manera (presentación). Finalmente, las metodologías estudiadas definen unas actividades y una secuencia o conjunto de secuencias que las relacionan (flujo de control). Cada uno de estos elementos comunes a las metodologías analizadas representan un paquete de nuestro meta-modelo.

La arquitectura de dicho meta-modelo puede verse en el diagrama de paquetes que se muestra en la Fig. 1.

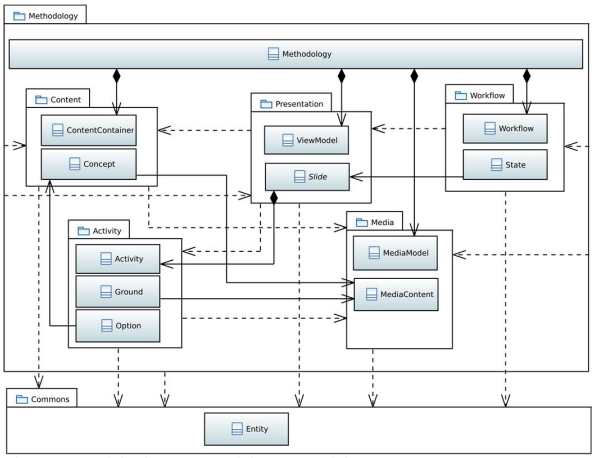


Figura 1. Modelo de paquetes del meta-modelo

La Fig. 2 muestra el meta-modelo en forma de diagrama de clases que se propone en este artículo.

El paquete central de esta arquitectura de modelos se denomina *methodology* y puede dividirse en 5 paquetes: *workflow, content, media, activity* y *presentation*.

El paquete *Content* (meta-clases en color purpura en la Fig. 2) se utiliza para representar la organización del contenido conceptual que se impartirá al alumno (por ej., nivel, unidad, lección, etc.). Este contenido está estructurado en forma de árbol (*Content*, *ContentContainer*, *Concept*) ya que permite relacionar los conceptos en función de la estrategia metodológica.

El paquete *workflow* (meta-clases en color gris oscuro en la Fig. 2) permite representar el flujo de la presentación de la aplicación. Para representar este flujo de información adoptamos la clásica representación en estructura de grafo, donde los nodos representan los estados (*State*) asociados a una presentación (*Slide* del paquete *presentation*) que puede tomar la aplicación, y donde las aristas representan las diferentes transiciones entre los estados o presentaciones de la aplicación.

El paquete *presentation* (meta-clases en color verde en la Fig. 2) se utiliza para representar la interfaz gráfica de los diferentes tipos de ejercicios que se desarrollan en la aplicación durante el proceso de aprendizaje. Este paquete define el concepto de dispositiva (*Slide*) que está asociado a una actividad concreta (*Activity* del paquete *activity*). Esto permite, por ejemplo, la reutilización de la vista de una diapositiva (*Slide*) con diferentes actividades (*Activity*) y viceversa.

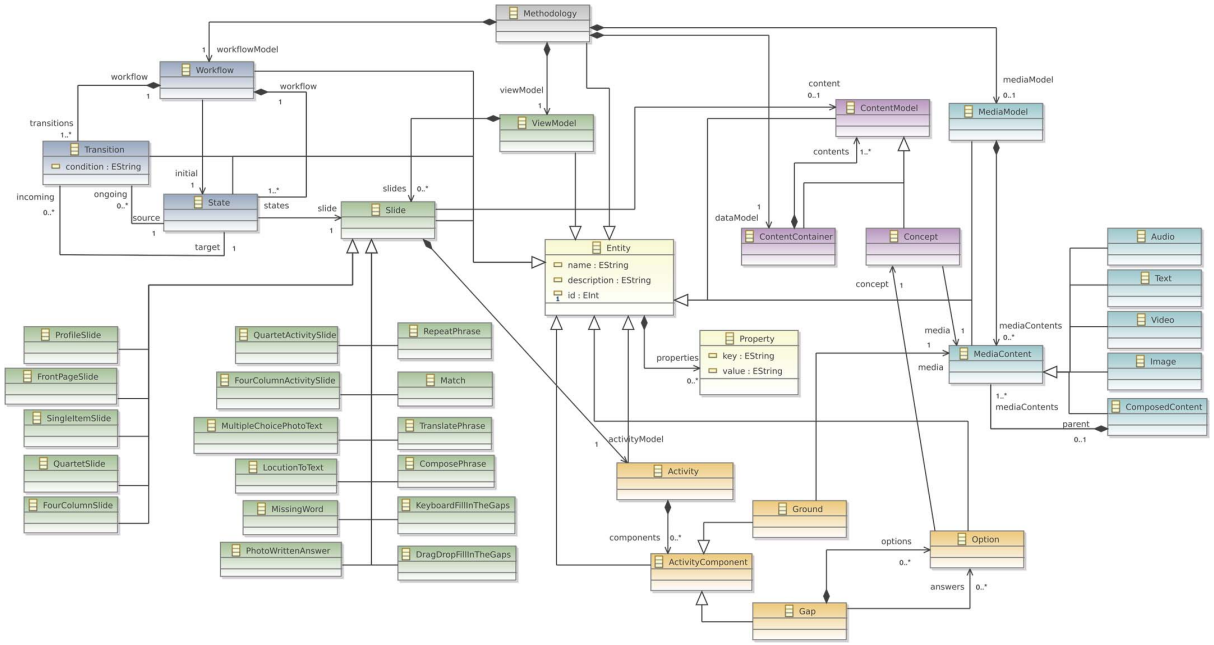


Figura 2. Meta-modelo que modela metodologías para el aprendizaje de un idioma

El paquete *activity* (meta-clases en color naranja en la Fig. 2) provee la capacidad de parametrizar el funcionamiento de las distintas actividades que se llevan a cabo durante el proceso de aprendizaje (por ejemplo, una actividad podría consistir en un ejercicio de elección múltiple, otra podría consistir en un ejercicio de rellenar los huecos, etc.). Toda actividad (*Activity*) define información para el usuario (*Ground*) definiendo un ejercicio o actividad; y define información que introduce el usuario (*Gap*) para llevar a cabo el ejercicio. La introducción de información por parte del usuario es contrastada con información asociada (*Option*) que representa tanto respuestas correctas como opciones candidatas a ser la/las respuesta/s. Toda información lleva asociada una o varias representaciones multimedia que permitirán parametrizar la presentación de la actividad.

El paquete *media* (meta-clases en color turquesa en la Fig. 2) se utiliza para representar los recursos (media) que se utilizarán en la presentación de acuerdo con la actividad que se lleva a cabo. Básicamente define 4 tipos de medios: audio, texto, video e imagen (*Audio*, *Text*, *Video* e *Image*) que pueden ser combinados para representar medios complejos (por ej. texto y locución).

Finalmente, el paquete *methodology* (meta-clase en color gris claro en la Fig. 2) define la meta-clase *Methodology* que representa a una metodología en su conjunto y está definida por el conjunto de modelos definidos por los meta-modelos de contenido, flujo de control, media y presentación.

Todas las meta-clases de este meta-modelo son descendientes de la meta-clase *Entity*, la cual está definida en un paquete denominado *commons* (meta-clases en color amarillo en la Fig. 2). Este paquete provee a las demás meta-clases del meta-modelo la capacidad de identificarse (*Entity*) y de extenderse (*Property*).

Para manipular los modelos conformes al meta-modelo presentado se ha desarrollado un editor de metodologías de enseñanza de idiomas, desarrollado como un plug-in de la plataforma Eclipse utilizando el Eclipse Modeling Framework (EMF). Este plug-in se ha definido a partir de un meta-modelo escrito en OclInEcore, el cual es un dialecto del Essential Meta-Object Facility (EMOF) de la OMG. OclInEcore además permite definir restricciones en OCL que se utilizan para definir los invariantes y las consultas de modelos.

IV. CASO DE ESTUDIO

El objetivo de este caso de estudio es ilustrar la versatilidad del meta-modelo propuesto para modelar distintos métodos de enseñanza de idiomas. Para ello, mostraremos un ejemplo de modelo que incluye una actividad que da soporte a un ejercicio de tipo elección múltiple en el que el usuario escucha una palabra y debe elegir entre 4 opciones de pares imagen-texto. Este caso de estudio muestra además los elementos comunes de dos actividades en dos metodologías distintas (Duolingo y Lexiway Junior).

La Fig. 3 muestra cómo se modela la presentación de una actividad en el caso de Lexiway. Por un lado, se muestra la actividad, tal como el usuario la maneja, y por otro se muestra el modelo (en concreto, la parte que representa el flujo de control o *workflow*). La Fig. 4 ilustra el poder de expresividad de esta tecnología, ya que muestra cómo un mismo modelo sirve para representar dos actividades similares (pero diferentes) de dos metodologías de aprendizaje diferentes.

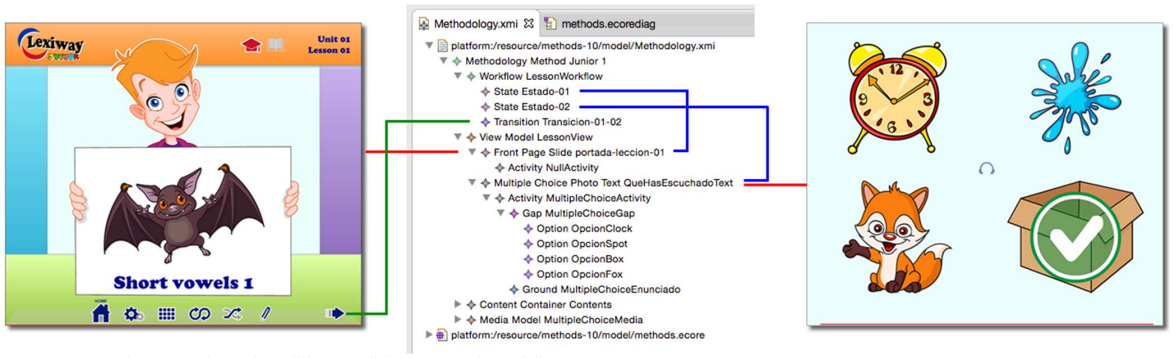


Figura 3. Presentación en Lexiway de 2 slides y su flujo de control (workflow).

El método ejemplo está basado en Lexiway Junior (una versión para niños del método Lexiway), y tiene sólo 2 diapositivas: una meramente informativa a modo de portada de una lección, y otra con una actividad tipo elección múltiple.

Un modelo de metodología (instancia de *Methodology*) define 4 intereses (concerns) diferentes: *workflow*, *content*, *media* y *presentation* (véase la Fig. 3 y la Fig. 4).

Como ya hemos visto, con el paquete *Content* es posible representar cualquier organización de contenido conceptual. En el caso del método de Duolingo (para aprender inglés teniendo el español como lenguaje nativo), éste está formado por 59 unidades (*ContentContainers*), cada una de las cuales tiene un número variable de lecciones (*ContentContainers*); a su vez, cada lección trabaja un conjunto de palabras (*Concepts*) nuevas, propias de esa lección, si bien una vez conocidas, dichas palabras pueden reforzarse en lecciones posteriores. En el caso del método Lexiway Junior, éste está formado por 6 bloques (*ContentContainers*), cada uno de los cuales consta de 4 unidades (*ContentContainers*) con 2 lecciones cada una; cada lección trabaja con 16 palabras agrupadas en 4 cuartetos (*ContentContainers*). En nuestro caso de estudio, definiremos una lección que contiene cuatro conceptos.

Desde el punto de vista del modelo de medios (*MediaModel*), cada uno de los cuatro conceptos tiene asociados una serie de recursos multimedia (*ComposedContent*), que en nuestro ejemplo están compuestos por diferentes recursos: imagen (*Image*) y texto (*Text*). Además, este modelo define los medios relacionados con el enunciado de la actividad: audio (*Audio*) y texto (*Text*). En concreto el texto del enunciado es "¿Qué palabra has escuchado?" y la locución es la locución de la palabra "Box". Esto requiere la definición de otro contenido de recursos multimedia de tipo compuesto (*ComposedComponent*).

En el paquete *activity* definimos la información necesaria para llevar a cabo la actividad. Esta información está contenida en una instancia de actividad (*Activity* MultipleChoiceActivity) que comprende, tanto el enunciado (*Ground* MultipleChoiceEnunciado), como la respuesta del usuario (*Gap* MultipleChoiceGap). En este caso, el *Gap* tendrá 4 opciones posibles (*Option*) y una respuesta establecida como correcta.

Con el paquete *Presentation* se modelan las vistas (*Slides*) de las actividades de la metodología. Éstas son: una actividad de introducción de la lección y una actividad de elección múltiple. En el caso del modelado de la introducción a la lección, se instancia una vista de presentación (*FrontPageSlide* portada-leccion-01) que define una actividad nula (*Activity* NullActivity) conteniendo la información de la introducción. Para definir la vista de la actividad de elección múltiple, se instancia una vista de elección múltiple (*MultipleChoicePhotoText* QueHasEscuchadoText), vinculada la actividad MultipleChoiceActivity.

Para definir el modelo de *workflow* utilizamos el paquete *workflow* donde se definen 2 estados (*States*) y una transición (*Transition*) con la precondición establecida a verdadero (true). Cada uno de los estados está asociados a una vista (ver las líneas azules en la Fig. 3). Por lo tanto, la transición entre ellos define un cambio en las vistas. Gráficamente esta transición se corresponde con un botón (siempre habilitado por la precondición definida) en la interfaz de usuario (ver la línea verde de la Fig. 3).

La Fig. 4 muestra cómo se corresponden los diferentes elementos del modelo del caso de estudio, en particular en la actividad de elección múltiple, en las metodologías Lexiway y Duolingo.

V. ANÁLISIS DE LA PROPUESTA

En esta sección abordaremos las distintas alternativas de modelado en función de los intereses (concerns) que son parte de la representación de la metodología de aprendizaje.

En el caso del modelado del contenido de la metodología de aprendizaje (*ContentModel*), éste permite definir diferentes tipos de estructura. Por ejemplo, en el caso de Lexiway, se modela una estructura en 5 niveles (nivel, bloque, unidad, lección e ítems); en el caso de Duolingo, se modela una estructura de 3 niveles (unidades, lecciones y palabras).

Los recursos pueden organizarse en una estructura de tipo árbol a través de la definición de un *MediaModel,* permitiendo la reutilización de recursos en diferentes metodologías. La independencia entre los modelos de contenido y recursos (*ContentModel* y *MediaModel*) permite que un recurso pueda asociarse a distintos conceptos (*Concept*) y viceversa. Además, la independencia entre el modelo de recursos (*MediaModel*) y el modelo de actividad (*ActivityModel*), permite la reutilización o intercambio de recursos para distintos componentes (*ActivityComponent*) de una actividad (*Activity*).

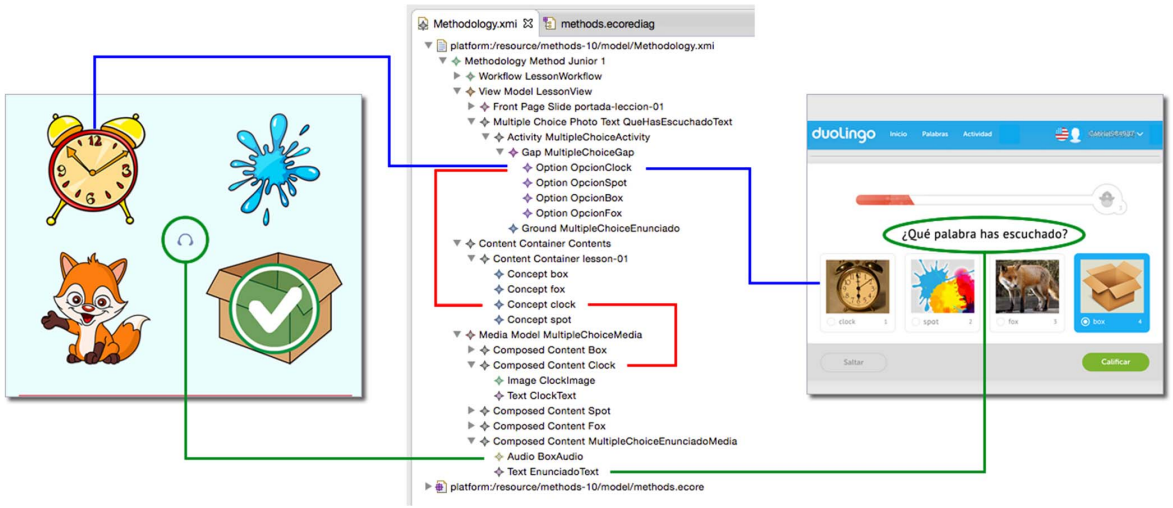


Figura 4. Presentación en Lexiway y en Duolingo de una actividad tipo elección múltiple con los mismos elementos.

El modelado de los ejercicios se realiza a través de la definición de dos modelos: el modelo de presentación (*Slide*) y el modelo de la actividad (*Activity*). Esta separación permite reutilizar diferentes actividades para el mismo comportamiento en la presentación (ver Fig. 4). Así pues, se pueden definir actividades tan diferentes como las siguientes:

- Rellenar huecos (bien arrastrando y soltando, o bien escribiendo)
- Componer frases
- Traducir frases
- Asociar los elementos de dos conjuntos
- Repetir una frase
- Escribir la palabra faltante
- Transcribir una locución
- Elección múltiple (con opciones de cualquier tipo)

Así mismo, se pueden representar todo tipo de diapositivas informativas. Por ejemplo, la portada de una unidad o lección, distintas distribuciones (layouts) de conceptos, etc.

En el caso del modelado del flujo de control (*workflow*) de una metodología de aprendizaje de idiomas se puede definir cualquier flujo de control mediante un grafo condicional. Por ejemplo, el orden en el que se estudian las lecciones o las lecciones se pueden estudiar. Por ejemplo, en el caso de Lexiway es necesario superar las lecciones de una unidad para pasar a la siguiente unidad. Además, dentro de una lección, es preceptivo el pasar por las actividades de “presentation”, para poder acceder a las de “practice”, “variations” o “spelling”. En el caso de Duolingo, existen lecciones o pruebas que una vez superadas, exoneran de tener que superar determinadas lecciones.

Finalmente, es importante recalcar que el meta-modelo propuesto, gracias al paquete *Commons,* permite modelar instancias de clases, proveyéndolas de la capacidad de identificarse (*Entity*) y de extenderse (*Property*).

VI. CONCLUSIONES

En este artículo se presenta un meta-modelo que nos permite recoger las principales características que están presentes en la mayoría de métodos para el aprendizaje de idiomas.

El meta-modelo es la primera y fundamental pieza de la arquitectura dirigida por modelos que nos permitirá desarrollar aplicaciones para el aprendizaje de idiomas.

Para ello se plantea la construcción de 5 modelos en torno del modelo de metodología: contenido, medio, flujo de control, actividad y presentación. Cada uno de ellos, representa un punto de vista diferente para conseguir la proyección de la metodología modelada.

Para ilustrar la flexibilidad del meta-modelo, se ha presentado un ejemplo en el que se compara el modelado de la misma actividad en dos metodologías de aprendizaje de idiomas distintas. Como consecuencia, se ha visto que los modelos resultantes no varían de manera significativa y tienen una gran capacidad de reutilización.

Todos estos modelos siguen los estándares de la OMG para conseguir la interoperabilidad entre estos modelos y otros de terceros. Esta interoperabilidad se implementa a partir editores reflexivos basados en el Eclipse Modeling Framework que permite la creación de plug-ins al entorno de desarrollo Eclipse.

El trabajo futuro pasa por desarrollar una herramienta basada en GMF para la obtención y verificación de modelos basados en este meta-modelo. Posteriormente, se desarrollarán las transformaciones que nos permitirán obtener código a partir de los modelos.

Además, se está trabajando en la implementación de un sistema de seguimiento personalizado para los usuarios del sistema y así poder trabajar con los datos de su perfil y experiencia.

AGRADECIMIENTOS

Este trabajo ha sido parcialmente financiado con la Ayuda a Grupos de la UCLM 2016. Así mismo, agradecemos al equipo de trabajo de Lexiway su colaboración y aportaciones.

REFERENCIAS

[1] LEXIWAY. URL: http://academialexiway.com/metodo-academia-ingles
[2] DUOLINGO. URL: https://es.duolingo.com/
[3] BABBEL. URL: https://es.babbel.com/

[4] BUSUU. URL: https://www.busuu.com/es/
[5] A. R. Puerta, "A model-based interface development environment", IEEE Software, vol. 14, no. 4, pp. 40-47, 1997.
[6] Y. Tian, H. Yang, L. Landy, "MDA-based Development of Music-Learning System", Proceedings of the 14th Chinese Automation & Computing Society Conference in the UK, Brunel University, West London, Sep. 2008.
[7] Object Management Group. Model driven Architectures. URL: http://www.omg.org/mda/
[8] R. Vesselinov, J. Grego. "Duolingo Effectiveness Study: FINAL REPORT". URL: http://static.duolingo.com/s3/DuolingoReport_Final.pdf
[9] H. M. Fardoun, F. Montero, V. López Jaquero. "eLearniXML: Towards a model-based approach for the development of e-Learning systems considering quality", Advances in Engineering Software, no. 40, pp. 1297-1305, 2009.
[10] H. M. Fardoun, "Elearnixml: towards a model-based approach for the development of e-learning Systems", PhD Thesis, University of Castilla-La Mancha, 2011.
[11] N. Koch, A. Kraus, "Towards a Common Metamodel for the Development of Web Applications", 3rd International Conference on Web Engineering (ICWE 2003), Springer Verlag LNCS 2722, pp. 497-506, July, 2003.
[12] S. Retalis, A. Papasalouros, M. Skordalakis. "Towards a generic conceptual design metamodel for web-based educational applications", Proc. of IWWWOST'02, CYTED, 2002.
[13] P. Blumschein, W. Hung, D. Jonassen, J. Strobel (Eds.), "Model-Based Approaches to Learning: Using Systems Models and Simulations to Improve Understanding and Problem Solving in Complex Domains", Sense Publishers, ISBN 978-90-8790-710-5 (hardback), 2009.

Gabriel Sebastián Rivera obtuvo la Licenciatura en Informática en la Universidad Politécnica de Valencia (UPV), y el título de Máster en Informática en la Universidad de Castilla-La Mancha (UCLM). Sus principales intereses de investigación están centrados en los campos del Multimedia, la Interacción Persona-Ordenador, y la Ingeniería del Software. Está implicado en el desarrollo de muchos proyectos de investigación relacionados con Interfaces de Usuario Distribuidas y con el Desarrollo Basado en Modelos de Interfaces de Usuario poniendo el foco en la Web como plataforma de desarrollo. Ha publicado más de 20 artículos de investigación y capítulos de libro en Revistas y Congresos Internacionales. Actualmente, trabaja como Gestor de Proyectos en el grupo de investigación Interactive Systems Everywhere. Es estudiante de doctorado del Departamento de Sistemas Informáticos (Escuela Superior de Ingeniería Informática de Albacete) en la UCLM, e investigador en el Instituto de Investigación en Informática de Albacete (I3A) en Albacete, España.

Ricardo Tesoriero obtuvo la Licenciatura en Informática en la Universidad Nacional de La Plata (UNLP), Argentina, y el título de Doctor en Informática en la Universidad de Castilla-La Mancha. Su principal interés de investigación es el Desarrollo Basado en Modelos de Interfaces de Usuario poniendo el foco en la Web como plataforma de desarrollo, así como la Interacción Persona-Ordenador en entornos de computación ubicua. Ha publicado más de 70 artículos de investigación y capítulos de libro en Revistas y Congresos Internacionales. Realizó una estancia postdoctoral en la Universidad Católica de Lovaina (UCL) en Lovaina-La Nueva, Bélgica, donde llevó acabo actividades de investigación en Desarrollo Basado en Modelos de Interfaces de Usuario. Ha sido miembro del comité en varios congresos científicos y workshops, como DUI (Distributed User Interfaces), INTERACCCION, ISEC, IADIS/WWW (La Web), etc. Es profesor en el Departamento de Sistemas Informáticos en la Escuela Superior de Ingeniería Informática de Albacete de la UCLM impartiendo docencia en las asignaturas de Métodos Avanzados de Desarrollo de Software y de Interacción Persona-Ordenador desde 2008.

Jose A. Gallud obtuvo la Licenciatura en Informática en la Universidad Politécnica de Valencia (UPV), y el título de Doctor en Informática en la Universidad de Murcia. Su principal interés de investigación está focalizado en la Interacción Persona-Ordenador, y en particular en el desarrollo de Sistemas Interactivos y de Interfaces de Usuario Distribuidas. El Dr. José A. Gallud ha publicado extensamente en esas áreas. Ha sido Editor Invitado para varias revistas internacionales, tales como JSS (The International Journal of Software and Systems), IJHCS (International Journal of Human Computer Studies), JUCS (Journal of Universal Computer Sciences). Tiene publicados algunos Libros y Capítulos en el campo de la Interacción Persona-Computador. Es miembro de diferentes asociaciones nacionales e internacionales (ACM y AIPO). Actualmente, trabaja como profesor en la Universidad de Castilla-La Mancha.

Capítulo 5

Model-based approach to develop learning exercises in language-learning applications

Citación: G. Sebastián, R. Tesoriero, J. A. Gallud, A model-based approach to develop learning exercises in language-learning applications, IET Software. URL http://digital-library.theiet.org/content/journals/10.1049/iet-sen.2017.0085

Tipo de publicación: Revista Internacional (IF: 0.695, Q4)

Esta publicación avala esta disertación por compendio de publicaciones.

IET Software

Research Article

IET Journals
The Institution of Engineering and Technology

Model-based approach to develop learning exercises in language-learning applications

ISSN 1751-8806
Received on 24th April 2017
Revised 22nd December 2017
Accepted on 27th January 2018
E-First on 12th March 2018
doi: 10.1049/iet-sen.2017.0085
www.ietdl.org

Gabriel Sebastián Rivera[1] ✉, *Ricardo Tesoriero*[2], *Jose Antonio Gallud*[2]

[1]*Instituto de Investigación en Informática de Albacete, University of Castilla-La Mancha, Albacete, Spain*
[2]*Escuela Superior de Ingeniería Informática de Albacete, University of Castilla-La Mancha, Albacete, Spain*
✉ *E-mail: gabriel.sebastian@uclm.es*

Abstract: Owing to factors such as globalisation and the extensive use of the Internet, people are increasingly interested in learning a second and even a third language. From an academic perspective, the process of foreign language learning is defined by methodologies and supported by technology. Language learning applications are composed of learning exercises, which are pedagogical tools to introduce new language concepts (new vocabulary, grammar etc.). The development of this type of applications is complex due to the diversity of language learning methodologies, the variety of execution environments (web, mobile and desktop) and the number of different technologies that can be used. This study proposes a model-driven approach developing software to support different language learning processes. These processes consist of different learning exercises running on different platforms. The authors' proposal describes a metamodel that defines the entities and their relationships to define learning exercises for learning applications. This metamodel enables designers to support the development of language learning applications. This study illustrates the expressiveness and reuses power of the proposal by modelling learning activities from two different learning processes (Lexiway and Duolingo).

1 Introduction

Interest in studying a foreign language has increased in recent years. The globalisation process and generalised access to the Internet are the main factors that explain this tendency.

The number of students enroled in language academies and schools is increasing each year. Moreover, the number of bilingual studies is growing at all educational levels: primary, secondary and universities. The number of language learning methodologies is also increasing at the same rate.

Technology has had a role in this process, as it has in many other fields. Consequently, users have many applications and tools related to this learning process at their disposal. The rapid growth in smartphone usage has led to mobile apps being, together with the web, one of the favourite environments in which to learn a language.

The development of this kind of application is somewhat complex due to the diversity of learning language methodologies, the variety of execution environments (web, mobile and desktop) and the number of different technologies that can be used. For this reason, the proposal presented in this paper makes use of model-driven development (MDD) and MD architecture approach, as a means to develop complex solutions and maximise savings in time and cost.

The proposal consists of a metamodel that models the main features present in most language learning methodologies. The solution is illustrated by modelling two similar activities from two language learning methodologies, using almost the same model.

The research in this paper can be summarised by the next research questions:

RQ1: Would it be possible to model most known learning language activities or exercises by means of the proposed metamodel?
RQ2: Would it be possible to model any workflow, any content and any resources taken from the studied learning language methodologies?

Regarding the success criteria, we have considered two success tests. The first one consists on considering a variety of exercises, from different methodologies, to check whether it is possible to model them using the metamodel. The second test is performed after applying the transformations rules that convert the model into code, and the success is checked empirically by analysing the resultant code compared with the original model's elements.

This paper is organised as follows: in the next section, we describe the problem in more detail, while Section 3 describes the research context together with the related work. Section 4 presents the metamodel proposed including a description of the main blocks of which it is composed. Section 5 shows an example to illustrate the concepts. Section 6 includes an analysis of additional modelling capabilities derived from our proposal. In Section 7, we discuss our solution. Finally, we present the conclusions and future work.

2 Description of the problem and elements of the solution

The development of learning exercises to implement a language learning application is a repetitive and tedious process. The process involves repeating the same resource management tasks many times and having duplicated code, which is difficult to maintain. Moreover, the problem of duplicated code and task repetition is multiplied by the number of different target platforms, making the overall process more complex and prone to errors. In this paper, we will refer to this as the traditional approach, which will be improved by the model-driven architecture (MDA) approach.

The following problems were experienced during the development of applications to support the Lexiway [1] language learning methodology: the client first wanted to develop an iOS app. However, she then requested a web version, so the development team reused the resources to produce a new source project. The scenario was set with two different, independent projects, with divergent resource and code repositories. The resulting environment was difficult to maintain, impacting negatively on subsequent projects such as the development of an Android app which was abandoned due to high costs.

The aim of a learning exercise is to teach a concept by means of an interactive activity. The exercise may use different mechanisms such as fill in the gaps, joining the lines or drag and drop images (see Fig. 1).

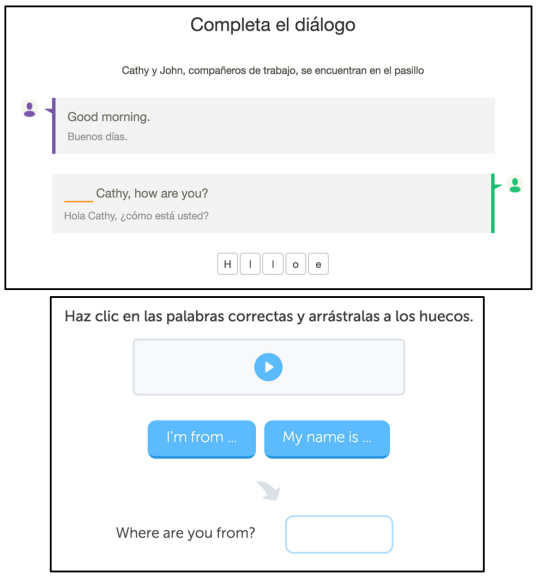


Fig. 1 *Example of fill in the gaps activities in Babbel and Busuu methods*

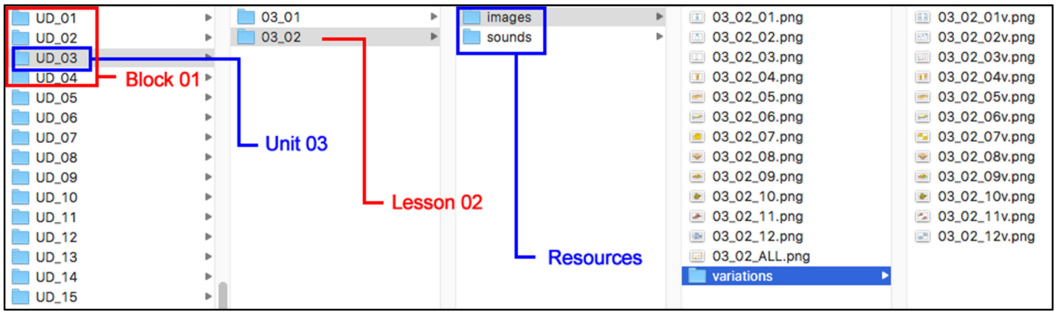


Fig. 2 *Resource folder structure of the Junior 1 level of the Lexiway method*

The traditional software development approach usually forces us to manually generate the different learning activities or exercises, with their corresponding media resources (audio, images or video). For example, the Junior 1 level in Lexiway learning methodology comprises six different blocks, each one composed of four units. Each unit consists of 2 lessons with 12 words (see Fig. 2). Manually managing all these learning activities involves much resource duplication. The validation of the final solution is also difficult to manage.

Thus, following the traditional approach to develop multimedia interactive learning applications, the idea is to programme several prototypical games. For each specific activity, a configuration file or a database register is manually defined. This configuration file is used by the logic layer to obtain the data and resources needed in that exercise or activity.

Finally, another conventional aspect in the development of interactive learning applications is that the navigation among activities and the progression of the level of difficulty are controlled by complex conditional structures that are difficult to manage and are possibly a source of problems.

Experience in the development of applications in a specific domain such as the development of learning exercises (in an application to learn a new language) is described by defining software artefacts (components, frameworks etc.) and repetitive tasks that can reduce the development time and, therefore, the software development cost. There are two conceptual tools that software engineering has traditionally employed to accomplish these challenges: increasing the level of abstraction and maximising the reuse.

The development of learning exercises requires the specification of a great variety of aspects. The following may be highlighted: the concept structure to be learnt; the media resources employed to represent those concepts; the mechanisms used to manage, link and present the concepts; moreover, the interaction flow control the user must follow. Let us illustrate these aspects with examples. An example of a concept the exercise might present is names of fruits (orange, apple etc.) or vegetables (potatoes, tomatoes etc.). The representation of the concept can vary, and so the exercise may use simple text, or an image, an audio or even a video. These representations can be managed in different ways. Finally, the learning is performed by relating all the concepts using different exercises. For instance, the exercise might be designed to drag and drop images to learn vocabulary, which is then complemented by a fill in the gaps exercise using the same vocabulary.

As has been shown, the aspects to be represented can have variations. Therefore, it is natural to conceive different ways to represent them, resulting in the need to build a domain-specific language to represent each different aspect. For example, the language used to represent a multimedia resource is different from that used to represent the interaction flow among the learning exercises of an application.

From the development perspective, all learning exercises are different. However, there are some common aspects. While the mechanism used in these exercises is the same (i.e. fill in the gaps or joining concepts), the concepts to be taught vary. For example, an exercise might associate concepts to enable the user to learn vocabulary (fruits, vegetables, transportation etc.). The

mechanisms to teach these concepts could be matching words to pictures or dragging pictures into gaps.

3 Research context

This section contains the research context, which includes two main elements. First, the essential concepts and general features extracted from different learning methodologies are presented. These elements will be used to apply the abstraction process. Second, selected related work in the field of MDD is presented. MDD is the chosen approach to tackle the problems described in the previous section.

Learning a foreign language is a process involving different methods, techniques and tools, each one appearing to be more effective than the others. In this paper, we focus on methodologies that offer some type of technological support (web site, mobile app or similar).

The methodologies we have analysed include Lexiway [1], Duolingo [2], Babbel [3] and Busuu [4]. Lexiway is a newer methodology that emphasises pronunciation, in both vocabulary and sentence composition, rather than grammar. Lexiway offers a variety of tools: a virtual classroom and an iOS app. Duolingo is a method to learn English that utilises the generalisation of mobile apps. This method emphasises student motivation and is based on gamification [5]. Babbel is a methodology that places more importance on vocabulary. In each exercise, the method detects strengths and weaknesses in learning, which form the basis of its lesson customisation. With Babbel, students learn grammar intuitively, with little awareness of the actual process. Busuu is a method that improves reading comprehension, conversation and writing, using a variety of exercises. In some aspects, Busuu is similar to Duolingo. Busuu's learning environment allows students to interact with an international community of native participants.

Although these methodologies take different approaches, it is possible to identify some common elements. The MDD approach will abstract all the elements present in the different methodologies. The MD architecture approach was proposed by the object management group (OMG) in 2011. In the MDA guide [6], it is defined as a solution to use models in software development. This solution gives a leading role to the models in the systems under development. As the name indicates, it is model driven since it runs all the phases from inception, design, building and development to maintenance and modification.

Software development technologies are constantly evolving. Following a traditional development approach, the functionality code and the implementation technology code are mixed together. Consequently, when the technology is enhanced, the old functionality must be updated or the entire project may even have to be rewritten to introduce the new paradigm. Web applications are a good example of this scenario since many server-side applications are being transformed into representational state transfer (REST) applications [7]. In scenarios such as these, MD architecture introduces abstraction levels and software reuse by using the concept of design-time interoperability [8]. This kind of interoperability is possible due to software for a platform-independent model specification that allows us to split the functionality specification from the implementation technology.

Thus, it is possible to combine the same functionality specification with different implementation technologies. Moreover, this functionality can be executed on different platforms or software architectures practically without changes. Platform-independent models require the implementation of model transformation (model compiler) that will allow us to automatically generate source code directly from the models [9].

In summary, thanks to MDA technology, it is possible to generate multi-platform applications from a unique platform-independent model. There are many advantages of using this approach. For example, let us assume we want to develop the same functionality, a learning exercise based on fill in the gaps, to be run on three different platforms, i.e. iOS, web and Android. Following a traditional approach, we would develop three different and independent source code projects. If we follow the MDA approach, we specify only one model, which will give us the code for the three different platforms.

In the MDA core, there are a number of important OMG standards: the unified modelling language, the meta object facility (MOF), extensible markup language metadata interchange and the common warehouse metamodel. These standards define the central MDA infrastructure, which has been used successfully in the modelling and development of modern systems.

In the field of human–computer interaction, we can find an approach that makes use of models to obtain the user interface (UI) [model-based UI development (MB-UID)] [10]. Hence, since our work focuses on the development of interactive systems, MB-UID can provide us with some useful elements.

It is generally thought that MDD will play an important role in the near future of software development, provided that more power MDD tools are delivered [11, 12]. Perhaps for this reason, there are so few related works formalising a metamodel to support a foreign language learning methodology.

Nevertheless, there are a number of related works using an MDD or MDA approach in the development of web applications [13–15]. In [16], MDA is used to develop a music learning application.

A methodology to model e-learning applications can be found in [17, 18], though this methodology does not focus on developing language learning applications. Moreover, our approach presents a set of models at different abstraction levels (architecture), which have different application viewpoints depending on the abstraction level. In [19], the experience of different research groups working in formal and informal learning language design by using mobile devices is described. There are several related research works on the developing of e-learning applications [20–23].

With regard to the use of models to build websites, some methodologies [24, 25] and models such as relationship management methodology (RMM) [26], web modelling language (WebML) [27] and WIED [28] have a direct influence on this research. They are interesting research works because their modelling can be located at the software level. However, our level of abstraction would be higher. WebML allows us the high-level description of a website considering several orthogonal dimensions as its content data (structural model): the pages that compose the site (composition model); the link topology among pages (presentation model); and the personalisation characteristics that allow the one-on-one content delivery (personalisation model). In [28], a formal model of information that can be considered a WebML plugin is presented. This plugin facilitates information modelling at this higher level of abstraction. WIED's authors argue that this modelling approach will provide a clearer connection between the business models and the processes, and the lower level designs, which are usually represented in the previous models.

4 Proposed metamodel

After studying and analysing the different language learning methodologies, we have identified, through an abstraction or generalisation process, the elements common to them all. In this section, we present these common elements and their formalisation by means of an Ecore metamodel with object constraint language (OCL).

First, we describe the common elements of the language learning applications used by the different methodologies.

All the analysed applications manage concepts (which can be words and sentences). These concepts are organised hierarchically (for example, lesson, units etc.). Each concept is associated with a set of resources. The concept ‘house’, for example, can be associated with one or more images, audios or videos.

Second, all the language learning applications have exercises. There are many types of exercises such as multiple choice, filling the gaps and sentence composition.

The third pillar in all methodologies is the established order by which the concepts are studied. For instance, in many methodologies, students can only begin a determined level if they have passed the previous level. This established order defines the workflow.

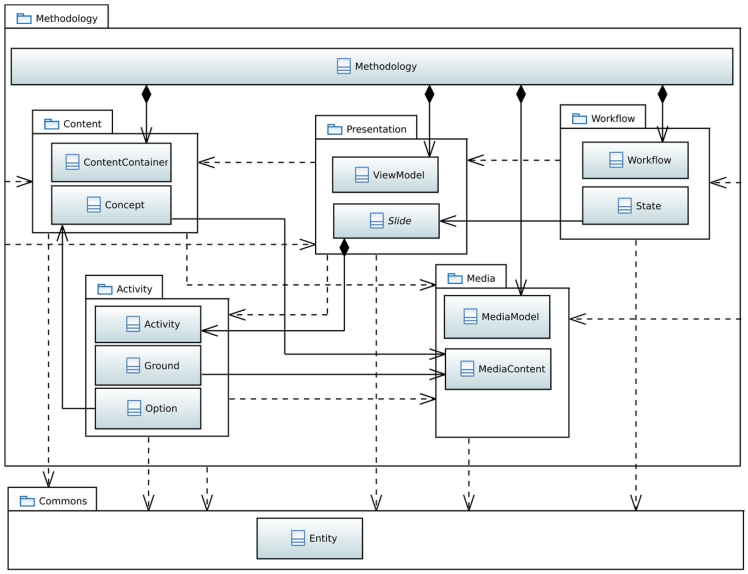


Fig. 3 *Metamodel package model*

In summary, the common elements are: concepts and concept hierarchy, resources, exercises and workflow.

The next step is to model all the common elements using an abstraction process, taking into account these general MDA principles: abstraction and reuse.

As a summary of the abstraction process, we can say that a methodology model (an instance of *methodology*) defines four different concerns: *workflow* (*Workflow* package), *content* (*content* package), *media* (*media* package) and *presentation* (*activity* and *presentation* package).

Fig. 3 shows different packages representing the common elements. The central package of this model architecture is *Methodology* and is divided into five packages: *workflow*, *content*, *media*, *activity* and *presentation*. A sixth package called *commons* contains utilities that will be explained later.

The following paragraphs describe the mapping among the common elements and the packages.

The first common elements are the 'concepts' managed by the methodology, which are modelled by the different elements of the *content* package (containing the metaclasses *concept* and *ContentContainer*). The hierarchy is also modelled by these same packages (specifically by the metaclass *ContentContainer*).

The resources are modelled by the elements defined in the *media* package.

The exercises are organised into two packages (*presentation* and *activity*). The *Presentation* package represents the particular kind of exercise such as multiple choices or filling the gap. The *activity* package represents the content of the exercise, for example, the different options of the multiple exercises or the statement.

The last common element, the workflow is modelled by the elements defined in the *workflow* packageA detailed view of the metamodel is shown in Fig. 4, which presents the defined metaclasses.

The *content* package (see the corresponding metaclasses in Fig. 4) is used to represent the organisation of the conceptual content that will be presented to the student. This content has a tree structure, allowing designers to establish relationships among concepts depending on the strategy defined by the methodology. For example, in a given methodology, a level can be composed of units, and a unit can be composed of lessons. The metaclasses *content*, *ContentContainer* and *concept* are defined following a composite design pattern.

The *media* package (see the corresponding metaclasses in Fig. 4) is used to represent the resources that will be used in the presentation. It defines four kinds of media: audio, text, video and image (*audio*, *text*, *video* and *image* metaclasses), which can be combined to represent complex media (for instance, text and speech).

The concepts of the *content* package are associated with elements of the *media* package.

The *Presentation* package (see the corresponding metaclasses in Fig. 4) is used to represent the graphical interface of the different exercises offered to the students in the learning process. This package defines the concept of the slide (*slide* metaclass), which is associated with a particular activity (*activity* metaclass, from the *Activity* package). This feature allows the reutilisation of a view of a slide (*slide* metaclass) in different activities (*activity* metaclass) and vice versa.

The *activity* package (see the corresponding metaclasses in Fig. 4) provides the capability to parametrise the functionality of the activities conducted in the learning process. Every *activity* metaclass provides the user with information (*ground* metaclass), which is used to define an exercise or activity through elements of the *MediaContent* (the text of the statement). Additionally, an *activity* metaclass defines information provided by the user through the *gap* metaclass in order to complete the exercise. A *gap* metaclass defines set of options and answers using the *option* metaclass. *Option* metaclass is associated with a concept (*concept* metaclass), which is associated with a *MediaContent*.

Let us illustrate this with an example based on a multiple-choice exercise. In this case, the exercise consists in playing a speech and showing different images, one of which corresponds with the speech. This exercise is represented by the *MultipleChoicePhotoText* metaclass, which defines the mechanism (playing the speech and the possibility to select different options). Both the speech (*ground* metaclass) and the options (*gap*) are parametrised by an instance of *activity*.

The *workflow* package (see the corresponding metaclasses in Fig. 4) allows us to represent the application's presentation flow (*workflow* metaclass). To represent this information flow, we adopt the classic graph notation, where the nodes represent the states (*state* metaclass) associated with a representation (*slide* metaclass in the *presentation* package) and the edges represent the different transitions (*Transition* metaclass) between states or slides.

Finally, the *methodology* package (see the corresponding metaclasses in Fig. 4) defines the metaclass *methodology*, which represents the complete methodology.

All the metaclasses in this metamodel descend from the *entity* metaclass, which is defined in a package called *commons* (see the corresponding metaclasses in Fig. 4). This package provides other

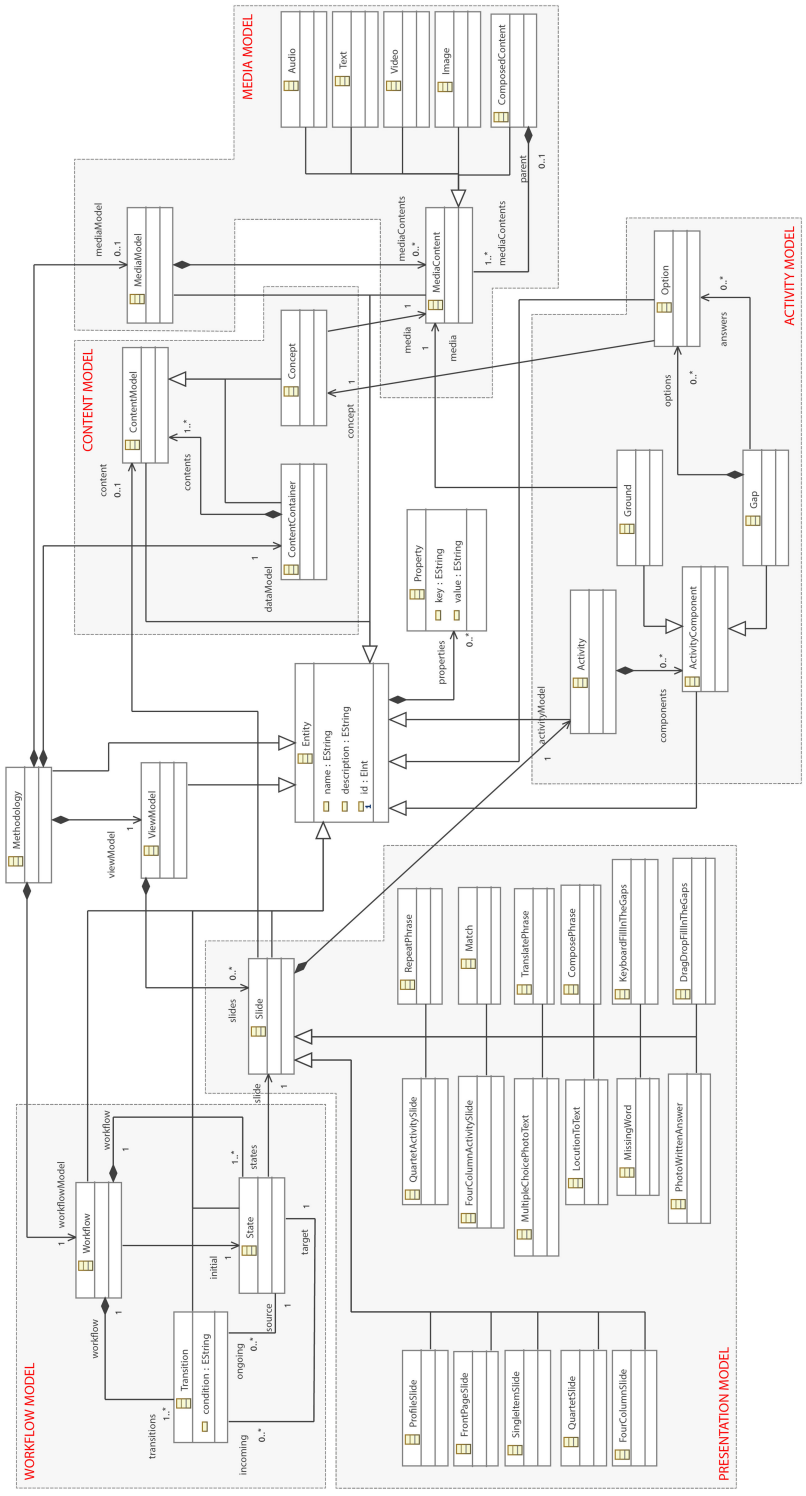


Fig. 4 *Metamodel that models language learning methodologies*

metaclasses with the identification (*entity* metaclass) and extension (*property* metaclass) capabilities.

Additionally, an editor has been developed to manage (create, edit and verify) the models according to the metamodel. This editor has been developed as an eclipse plugin, by using the eclipse modelling framework (EMF). This plugin has been defined by using OclInEcore, which is a dialect of the OMG essential MOF. OclInEcore allows us to define OCL constraints that are used to define the model invariants and queries.

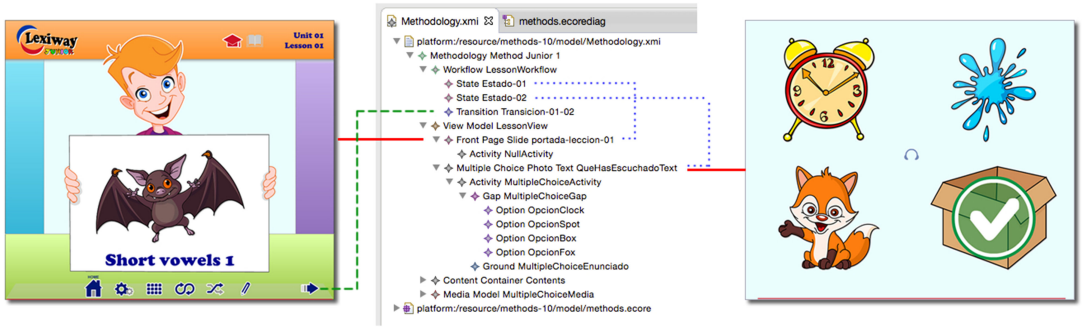


Fig. 5 *Two slides, presentation and workflow in Lexiway*

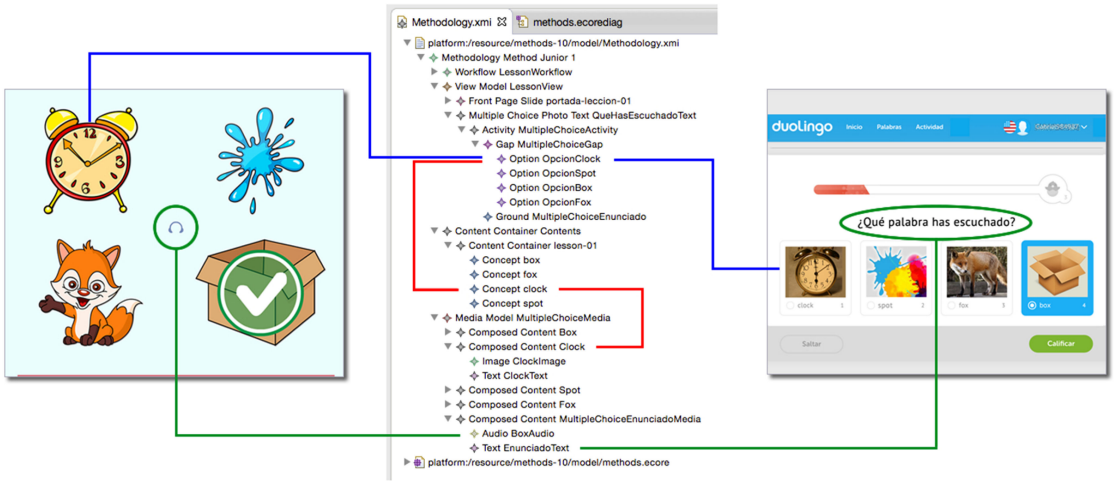


Fig. 6 *Presentation in Lexiway and Duolingo of a multiple-choice activity using the same elements*

5 Case study

The main goal of this case study is to illustrate the flexibility and adaptability of the proposed metamodel to model different language learning methodologies. We show an example of a model including an activity based on a multiple-choice exercise. During the activity, the student hears a word and has to select one option from four image–text pairs.

This case study presents the common elements of two similar activities taken from two different methodologies (Duolingo and Lexiway).

Fig. 5 shows how to model the presentation of the activity in the case of Lexiway. This figure shows two parts: first, the exercise (graphical UI]) as presented to the student and, second, the model (specifically, the workflow).

The example is based on Lexiway Junior (a version of the method specially adapted to young people), and it has just two slides: the first one contains information on the exercise the second one includes the multiple-choice exercise.

Let us review the metamodel concepts explained using this example.

As mentioned in the previous section, a methodology model (an instance of *methodology*) defines four different areas: *workflow*, *content*, *media* and *presentation* (see Figs. 5 and 6).

With the content package, we can represent any conceptual organisation. In the case of Duolingo (a method to learn English using Spanish as the language of origin), the organisation is based on 59 units (*ContentContainers*). Each unit has a variable number of lessons (*ContentContainers*). Each lesson contains a set of new and related words (*concepts*). Once users have studied these words, they can reinforce them in later lessons. In the case of the Lexiway Junior, the method is organised in six blocks (*ContentContainers*). Each block is organised in four units (*ContentContainers*), which contain two lessons each. A lesson contains 16 words grouped in 4 quarters (*ContentContainers*).

In this example, we define a lesson containing four concepts.

From the media model viewpoint (*MediaModel* metaclass), each of the four concepts has an associated set of multimedia resources (*ComposedContent* metaclass), which are composed of the different image (*image*) and text (*text*) resources. This model also defines the media elements related to the exercise statement: audio (*audio*) and text (*text*). In this exercise, the statement text is defined as '¿Qué palabra has escuchado?' ('What word did you hear?') and the speech is 'Box'. This requires the definition of another composed multimedia content resource (*ComposedComponent*).

The *activity* package is used to define the information needed to carry out the exercise. This information is contained in an instance of activity (*activity* MultipleChoiceActivity) including both the statements (*ground* MultipleChoiceEnunciado) and the user's answer (*gap* MultipleChoiceGap). In this case, the gap has four possible options (*option*) and only one established as the right answer.

The *presentation* package is used to model the activity views (*slides*) of the methodology. The views in this example are the following: an activity with the introduction to the lesson and a multiple-choice activity. In the modelling of the introduction to the lesson, a presentation view is instantiated (*FrontPageSlide* portada-lección-01) that defines a null activity (*activity* NullActivity containing the information from Section 1. To define the view of the multiple-choice activity, a multiple-choice view is instantiated (*MultipleChoicePhotoText* QueHasEscuchadoText), linked to the MultipleChoiceActivity.

The *Workflow* package is used to define the workflow model, where two states (*states*) and one transition (*transition*) with the

```
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5    <title>Frontpage slide</title>
6    <meta charset="utf-8">
7    <meta name="viewport" content="width=device-width, initial-scale=1"
     >
8    <link rel="stylesheet" href="http://maxcdn.bootstrapcdn.com/
     bootstrap/3.3.5/css/bootstrap.min.css">
9    <link rel="stylesheet" href="lexiway.css" type="text/css" />
10   <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/
     jquery.min.js"></script>
11   <script src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/
     bootstrap.min.js"></script>
12 </head>
13
14 <body>
15   <div class="container" id="container" style="background-image: url(
     'images/presentation.png')" alt="PRESENTATION">
16     <div class="row" style="margin-top:550px;">
17       <div class="col-xs-12 text-center">
18         <a href="home.html"><img src="images/home.png" alt="HOME"></a
           >
19         <a href="config.html"><img src="images/config.png" alt="
           CONFIG"></a>
20         <a href="index.html"><img src="images/index.png" alt="INDEX">
           </a>
21         <a href="repeat.html"><img src="images/repeat.png" alt="
           REPEAT"></a>
22         <a href="restart.html"><img src="images/restart.png" alt="
           RESTART"></a>
23         <a href="practice.html"><img src="images/practice.png" alt="
           PRACTICE"></a>
24         <a href="next.html" style="float:right"><img src="images/
           next.png" alt="NEXT"></a>
25       </div>
26     </div>
27   </div>
28 </body>
29
30 </html>
```

Fig. 7 *Source code for instances of slide metaclasses*

precondition established to true are defined. Each of the states is associated with a view (see the dotted lines in Fig. 5). Thus, the transitions between them define a change of views. Graphically, this transition is shown as a button (always enabled due to the precondition) on the UI (see dashed line in Fig. 5).

Fig. 6 shows the correspondence among the different elements of the model of the case study, the multiple-choice activity from both methodologies Lexiway and Duolingo. Fig. 6 illustrates the expressiveness power of this technology since it shows how the same model is used to represent two similar activities from two different learning methodologies.

6 Validation of the proposal

This section presents how the models confirming the methodology metamodel are mapped into the source code of a generated application.

We select the Lexiway methodology model presented in Figs. 5 and 6, and hypertext markup language (HTML) as the target language to how the methodology models are mapped into source code.

The methodology metamodel defines five packages containing the metaclasses to model different language learning application features.

Instances of these metaclasses are used to generate application source code using transformation rules. These rules are defined using model transformation languages such as ATL, MOFScript, ACCELEO etc.

Fig. 7 shows the source code generated for the *portada-leccion-1* instance of *FrontPageSlide*. This code presents the structure of the HTML five compliant documents generated for all *slide* instances.

These instances require the page title (line 5), the presentation image and alternative text (line 15) and navigation control input parameters to generate this code.

The arguments for these parameters are obtained from the *NullActivity Activity* instance. This instance contains the *portada-leccion-1-titulo ground* instance which defines the content of *title* tag (line 5) using the resource pointed by the *portada-leccion-1-titulo text* instance defined in the *MediaModel* instance.

This *activity* instance also contains the *portada-leccion-1-contenido-imagen* and *portada-leccion-1-contenido-texto Ground* instances which define the presentation image background and its alternative text in line 15 using the resources pointed by *portada-leccion-1-contenido-imagen image* instance and *portada-leccion-1-contenido-text text* instance, respectively.

Finally, it contains two *ground* instances for each navigation button defining the button image and alternative text (e.g. *portada-leccion-1-contenido-next-imagen* and *portada-leccion-1-contenido-next-text*). These *ground* instances are linked to *image* and *text* instances referring to navigation button contents (e.g. *portada-leccion-1-contenido-next-imagen* and *portada-leccion-1-contenido-next-text*).

The *portada-leccion-01 FrontPageSlide* is linked to other *slide* instances. These links are represented in the *LessonWorkflow workflow* instance where *state* instances are linked to *slide* instances representing views. Therefore, *transition* instances linking *state* instances represent transitions between views. These transitions are represented in terms of event conditions described in *condition* properties of *transition* instances.

For instance, *Transicion-01-01*is a *transition* instance which *source* property points to *Estado-01 state* instance, *target* property points to *Estado-02 state* instance and *condition* property is set to *portada-leccion-1-contenido-next-imagen.clicked*. As *Estado-01* points to *portada-leccion-1 FrontPageSlide* instance and *Estado-02* points to *QueHasEscuchado MultipleChoicePhotoText* instance, when *portada-leccion-1-contenido-next-imagen image* is clicked, the view represented by the *portada-leccion-1 FrontPageSlide* instance is replaced by the view represented by the *QueHasEscuchado MultipleChoicePhotoText* instance.

Fig. 8 shows the source code generated for the *QueHasEscuchadoText MultipleChoicePhotoText* instance. While lines 1–14 present the code to load web frameworks such as Bootstrap and jQuery; lines 15–36 present the layout of the slide depicted on the left-hand side of Fig. 6 which is organised in a grid consisting of 2 rows×2 columns.

The content of the *QueHasEscuchadoText MultipleChoicePhotoText* instance is defined by the *MultipleChoiceActivity Activity* instance which consists on a *Gap* instance (i.e. *MultipleChoiceGap*) containing four *option* instances (i.e. *OpcionClock, OpcionSpot, OpcionBox* and *OpcionFox*) and a *Ground* instance (i.e. *MultipleChoiceEninciado*).

While the *MultipleChoiceEninciado ground* instance is directly associated to *MediaContent* instances (e.g. *BoxAudio Audio* instance); *option* instances are related to *Concept* instances (e.g. *clock*).

Therefore, each grid cell identifies a *Concept* instance (e.g. *clock*) which is attached to a *MediaContent* instance (e.g. *ClockImage*, *ClockText*) that parametrises *img* tags.

Thus, the validation of the proposal can be verified by means of model entity instances and the generated source code.

Regarding the evaluation, we adopted the number of code lines generated per model instance created as the metric to evaluate this proposal.

The overall results for this case of study consisting on two slides deliver the results presented in Table 1.

7 Discussion

In this section, we analyse different modelling alternatives. The advantages and disadvantages of the proposed solution are also discussed.

The different options will depend on the concerns defined in the learning processing methodology.

The modelling of the learning methodology content (*ContentModel*) allows defining different kinds of structures. Let us see an example. In the case of Lexiway, a structure is modelled in five levels (level, block, unit, lesson and item); moreover, in the case of Duolingo, the structure is modelled in three levels (units, lessons and words).

The resources can be organised using a tree structure through the *MediaModel* definition, allowing the reutilisation of resources from different methodologies. The independence between the content and resource models (*ContenModel* and *MediaModel*) allows a resource to be associated with different concepts

```
<!DOCTYPE html>
<html>
<head>
  <title>practice-01-04</title>
  <meta charset='utf-8'>
  <meta name='viewport' content='width=device-width, initial-scale=1'
  >
  <link rel='stylesheet' href='lexiway.css' type='text/css'>
  <script src='https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/
  jquery.min.js'>
  </script>
  <script src='http://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/
  bootstrap.min.js'>
  </script>
  <script src='script.js'>
  </script>
</head>
<body onload='playSound(this,"/media/MultipleChoiceMedia/Box/
  BoxAudio.mp3");'>
  <div id='container'>
    <div class='quartet-item quartet-item-1'>
      <img src='/media/MultipleChoiceMedia/Clock/ClockImage.png'
      class='quartet-item quartet-item-' onclick='checkAnswer(this,4,
        1);' alt='clock'> </img>
      <div id='practice-check-1'> </div>
    </div>
    <div class='quartet-item quartet-item-2'>
      <img src='/media/MultipleChoiceMedia/Spot/SpotImage.png' class=
      'quartet-item quartet-item-' onclick='checkAnswer(this,4,2);'
      alt='spot'> </img>
      <div id='practice-check-2'> </div>
    </div>
    <div class='quartet-item quartet-item-3'>
      <img src='/media/MultipleChoiceMedia/Fox/FoxImage.png' class='
      quartet-item quartet-item-' onclick='checkAnswer(this,4,3);'
      alt='fox'> </img>
      <div id='practice-check-3'> </div>
    </div>
    <div class='quartet-item quartet-item-4'>
      <img src='/media/MultipleChoiceMedia/Box/BoxImage.png' class='
      quartet-item quartet-item-' onclick='checkAnswer(this,4,4);'
      alt='box'> </img>
      <div id='practice-check-4'> </div>
    </div>
    <div id='practice-enunciado' onclick='playSound(this,"/media/
      MultipleChoiceMedia/Box/BoxAudio.mp3")'> </div>
  </div>
  <div id='actions-container' onmouseover='showActions()' onmouseout=
  'hideActions()'> </div>
</body>
</html>
```

Fig. 8 *Source code for instances of slide metaclasses*

Table 1 Source code generation evaluation

Model	Slide			
	Front page		Multiple choice	
	Ins.	Lines	Ins.	Lines
workflow	2	30	2	37
view and activity	4		8	
content			5	
media	15		14	
total	**21**	**30**	**29**	**37**
ratio	**1.43**		**1.28**	

The bold values indicate to highlight the total and ratio values.

(*Concept*) and vice versa. Additionally, the independence between the resources (*MediaModel*) and activity (*ActivityModel*) models allow the reutilisation or interchange of resources among different components (*ActivityComponent*) of an activity (*Activity*).

The modelling of the exercises is carried out by defining two models: the presentation model (an instance of *Slide* metaclass) and the activity model (an instance of *Activity* metaclass). This separation allows us to reuse different activities taking the same presentation behaviour as invariant (see Fig. 6). Thus, it is possible to define different activities such as: filling the gaps (either dragging and dropping or by writing), sentence composition, sentence translation, match elements from two sets, sentence repetition, write the missing word, speech to text and multiple choices (with different options).

Similarly, it is possible to represent different kinds of exercises such as the introduction of a unit or concept layouts.

In the case of modelling the learning methodology workflow, it is possible to define any workflow by using a conditional graph, for example, to establish an order for the lessons or to decide what lessons are enabled for study. In the case of Lexiway, for instance, the student has to pass all the lessons in a unit before starting the next lesson. Within a lesson, the 'presentation' is mandatory before accessing the 'practise', 'variations' or 'spelling' lessons. In Duolingo, however, if students pass certain lessons or tests, they can then bypass other lessons.

It is important to note that by using the *Commons* package, we are able to model class instances, with the capability to identify (*Entity*) and to extend (*Property*) them.

One of the disadvantages of using the traditional approach to develop the same application for different platforms is that we must develop the same number of applications as platforms. This approach is prone to errors since it is hard to maintain coherence among the applications. All applications with the same functionality should be developed, implemented and tested separately. Using an MDA approach, since we have a platform-independent model, we can derive the platform specific model of the application automatically and hence the source code.

The use of a platform-independent model is useful not only during the software development process, but also reduces the maintenance costs. The modifications are applied in the model and to target platforms after their verification.

Another consequence related to the platform representation divergence and the documentation needed is the ability to maintain perfect synchronisation between the source code and the software documentation. This is possible since the documentation is automatically generated from the models. As is commonly known, using the traditional approach, synchronisation does not exist without great, and usually manual effort which is also a source of problems.

The concept of design-time interoperability is one of the main features of the MDA-based technologies. This feature allows us to capture the different application aspects by using independent models. These models are integrated into the last stage of development, just after the generation of the source code. The advantage of using this mechanism is that making changes to one particular model has a minimum impact on the other models. For example, the representation of a concept (the image) can be changed by modifying only the media model without affecting the other models (content, workflow).

Finally, it is time to revisit the research questions:

RQ1: Would it be possible to model most of the known learning language activities or exercises by means of the proposed metamodel? Section 5 shows the flexibility of the metamodel, since we have used the same elements to represent two different activities from two different learning languages methodologies?

RQ2: Would it be possible to model any workflow, any content and any resources taken from the studied learning language methodologies? The case study also shows that the different elements of an activity can be easily represented?

Regarding the success criteria, we have considered two success tests. The first one consists of considering a variety of exercises to check whether it is possible to model them using the metamodel. The obtained results can be considered positive, though in this paper we have presented only two activities. We have repeated this test with many other activities with the same good results.

The second test is performed after applying the transformations to convert the model into code, and the success is checked empirically by comparing the code obtained from each metamodel element with the expected one. Section 6 contains enough evidence to demonstrate the capability of the metamodel to represent the variety analysed up to now.

8 Conclusions

This paper presents a metamodel that allows us to collect and represent the main features considered common to most of the learning language methodologies.

This metamodel is the first and fundamental piece in the MD architecture, which will allow us to develop learning language applications.

Our MDA proposal involves defining four models: content, media, workflow and presentation. Each one represents a different view that allows us to obtain the projection of the modelled methodology.

To illustrate the flexibility of the proposed metamodel, an example is presented. This example is used to compare the modelling of a similar activity taken from two different methodologies. As a consequence, it has been proved that the resultant models do not vary significantly and have a high degree of reuse.

Additionally, this paper includes a detailed explanation of the mapping between the initial model and the resultant code obtained.

All the defined models follow the OMG standards, which guarantee interoperability with other external models following the OMG. This interoperability is implemented using reflexive editors based on the EMF, which allow us to create Eclipse plugins.

Future work includes developing a graphical modeling framework (GMF) based tool to obtain and verify models generated with our metamodel. The most suitable transformations will be developed to generate code from the models.

Finally, a customised tracking system for users is under development, which will allow us to analyse their profiles and experience when using the language learning applications.

9 Acknowledgments

This work has been partially supported by the 'Ayuda a Grupos UCLM 2016'. We also wish to thank the Lexiway team for their collaboration.

10 References

[1] El Método Lexiway (Lexiway website). Available at http://academialexiway.com/academia-ingles/metodo-academia-ingles.html, accessed 28 November 2016

[2] Duolingo website. Available at https://www.duolingo.com/, accessed 28 November 2016

[3] Babbel website. Available at https://www.babbel.com/, accessed 28 November 2016

[4] Busuu website. Available at https://www.busuu.com/, accessed 28 November 2016

[5] 'Duolingo effectiveness study'. Final Report, 2012. Available at http://static.duolingo.com/s3/DuolingoReport_Final.pdf, accessed 28 October 2016

[6] Object management group model-driven architectures. Available at http://www.omg.org/mda/, accessed 28 October 2016

[7] Fielding, R.: 'Architectural styles and the design of network-based software architectures'. PhD thesis, University of California, Irvine, 2000, Chapter 5: 'Representational State Transfer (REST)'

[8] Mellor, S., Scott, K., Uhl, A., *et al.*: '*MDA distilled: principles of model-driven architecture*' (Addison Wesley, Boston, 2004), ISBN 0-201-78891-8

[9] Kleppe, A., Warmer, J., Bast, W.: '*MDA explained: the model driven architecture™: practice and promise*' (Addison Wesley, Boston, 2003), ISBN 0-321-19442-X

[10] Puerta, A.: 'A model-based interface development environment', *IEEE Softw.*, 1997, **14**, (4), pp. 40–47

[11] Conn, S., Forrester, L.: 'Model driven architecture: a research review for information systems educators teaching software development'. Proc. ISECON 2005, 2006

[12] Bizonova, Z., Ranc, D., Drozdova, M.: 'Model driven E-learning platform integration'. Proc. Second PROLEARN Doctoral Consortium in Technology Enhanced Learning, Crete, Greece, September 2007, vol. **18**, pp. 8–15

[13] Koch, N., Kraus, A.: 'Towards a common metamodel for the development of web applications'. Proc. Third Int. Conf. Web Engineering 2003, (LNCS, 2722), pp. 497–506

[14] Retalis, S., Papasalouros, A., Skordalakis, M.: 'Towards a generic conceptual design metamodel for web-based educational applications'. Proc. IWWOST'02, CYTED, 2002

[15] Blumschein, P., Hung, W., Jonassen, D., *et al.* (Eds.): '*Model-based approaches to learning: using systems models and simulations to improve understanding and problem solving in complex domains*' (Sense Publishers, Rotterdam, hardback, 2009), ISBN 978-90-8790-710-5

[16] Tian, Y., Yang, H., Landy, L.: 'MDA-based development of music-learning system'. Proc. 14th Chinese Automation & Computing Society Conf., UK, Brunel University, West London, September 2008

[17] Fardoun, H., Montero, F., Jaquero, V.: 'eLearniXML: towards a model-based approach for the development of e-learning systems considering quality', *Adv. Eng. Softw.*, 2009, **40**, pp. 1297–1305

[18] Fardoun, H.: 'eLearniXML: towards a model-based approach for the development of e-learning systems'. PhD thesis, University of Castilla-La Mancha, 2011

[19] Bárcena, E., *et al.*: 'State of the art of language learning design using mobile technology: sample apps and some critical reflection'. Proc. 2015 EUROCALL Conf., Padova, Italy, 2015, pp. 36–43

[20] García, F., Sarasa, A., Sierra, J.: 'Educational software: case studies and development methods', *IEEE Rev. Iberoamericana Tecnol. Aprendizaje*, 2014, **9**, (2), pp. 41–42

[21] García, F.: '*Advances in E-learning: experiences and methodologies*', (Information Science Reference, New York, 2008), ISBN 978-1-59904-756-0

[22] Oye, N., Salleh, M., Iahad, N.: 'E-learning methodologies and tools', *Int. J. Adv. Comput. Sci. Appl. (IJACSA)*, 2012, **3**, (2), pp. 48–52

[23] Lanzilotti, R., Ardito, C., Costabile, M., *et al.*: 'eLSE methodology: a systematic approach to the e-learning systems evaluation', *Educ. Technol. Soc.*, 2006, **9**, (4), pp. 42–53

[24] Halasz, F., Schwartz, M.: 'The dexter hypertext reference model', *Commun. ACM*, 1994, **37**, (2), pp. 30–39

[25] Conallen, J.: '*Building web applications with UML*' (Addison Wesley, Boston, 1999), Object Technology Series, p. 336

[26] Isakowitz, T., Stohr, E., Balasubramanian, P.: 'RMM: a methodology for structured hypermedia design', *Commun. ACM*, 1995, **38**, (8), pp. 34–44

[27] Ceri, S., Fraternali, P., Bongio, A.: 'Web modeling language (WebML): a modeling language for designing web sites'. Proc. WWW9 Conf., Amsterdam, 2000, pp. 137–157

[28] Tongrungrojana, R., Lowe, D.: 'WIED: a web modeling language for modeling architectural-level information flows', *J. Digit. Inf.*, 2005, **5**, (2)

Capítulo 6

A Unified Abstract Mechanism to Model Language Learning Activities

Citación: G. Sebastián, R. Tesoriero, J. A. Gallud, A Unified Abstract Mechanism to Model Language Learning Activities, Computing And Informatics.

Tipo de publicación: Revista Internacional (IF: 0.421, Q4)

(Enviado)

Computing and Informatics, Vol. 39, 2019, 1001–1025, V 2020-Feb-14

A UNIFIED ABSTRACT MECHANISM TO MODEL LANGUAGE LEARNING ACTIVITIES

Gabriel Sebastián

Albacete Research Institute of Informatics
University of Castilla-La Mancha
Campus Universitario, s/n
02071 Albacete, Spain
e-mail: gabriel.sebastian@uclm.es

Ricardo Tesoriero, Jose A. Gallud

Faculty of Computer Science Engineering
University of Castilla-La Mancha
Campus Universitario, s/n
02071 Albacete, Spain
e-mail: ricardo.tesoriero@uclm.es, jose.gallud@uclm.es

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 conceptual model to develop the activities of language learning applications. It defines a new abstraction mechanism to model these activities as part of a model-driven approach to develop applications supporting different language learning processes running on different hardware and software platforms. We define a metamodel that describes the entities and relationships representing language learning activities as well as a series of examples that use the proposed abstraction mechanism to represent different language learning activities. The modelling process is simplified using a common representation that does not affect neither the visual presentation, nor the interaction of each activity. The article includes an evaluation that analyses the product correctness, robustness, extensibility, and reusability of the obtained code. These results conclude that the code

generated using the proposed approach overcomes the code generated following a traditional approach.

Keywords: Model-driven development, languages learning methodologies and apps, Web technologies.

1 INTRODUCTION

Language learning applications development is complex due to the diversity of learning language methodologies, the variety of execution environments (Web, mobile and desktop) and the number of different technologies that can be used [9].

Besides, the development of learning exercises to implement a language learning application is a repetitive and tedious process. The process involves repeating the same resource management tasks many times having duplicated code, which is difficult to maintain. Moreover, the problem of duplicated code and task repetition is multiplied by the number of different target platforms, making the overall process more complex and prone to errors. This approach, aka the traditional approach, can be improved using a Model-driven Architecture (MDA) to capture common features in Computation Independent Models (CIMs).

To capture this common features, we performed an analysis of the Lexiway[1] language learning methodology and we experienced the following problems. Initially, the client requested the development of a mobile application for the iOS platform. Later on, they requested a Web version of the same application. Therefore, the development team adapted software resources to produce a new source project for the new version of the application.

This new development scenario consisted of two independent branches for the same project which leads to divergent resources and source code. The resulting environment was difficult to maintain, impacting negatively on subsequent projects. For instance, the development of an Android version of the application was abandoned due to the high development costs.

The aim of a learning activity is teaching a concept by means of an interactive experience. Learning activities employ different interaction mechanisms such as fill-in the gaps (see the right side of Fig. 4), joining the lines, Drag & Drop images (see the right side of Fig. 5), an so on.

The traditional software development approach usually forces us to manually generate the different learning activities or exercises, with their corresponding media resources (audio, images or video). For example, the JUNIOR 1 level of the Lexiway learning methodology consists of 6 different blocks composed of 4 units each. In addition, each unit consists of 2 lessons containing 12 words. Managing all these learning activities manually involves a high level of resource duplication which leads

[1] http://academialexiway.com/academia-ingles/metodo-academia-ingles.html

to software validation difficult to manage.

Thus, following the traditional approach to develop multimedia interactive learning applications, the idea is to develop several prototypical games. For each activity type, a configuration file or a database register is manually defined. This configuration file defines the data and resources required by the logic layer of the activity to be executed.

Finally, another conventional aspect in the development process of interactive learning applications is that the navigation among activities and the progression of the level of difficulty is controlled by complex conditional structures that are difficult to manage and maintain which usually became a source of problems.

From the experience achieved during several years of developing language learning applications, we have learnt that the use of software artefacts (i.e. components, frameworks) and performing repetitive tasks reduces considerably the development time and costs. There are two conceptual tools that software engineering has traditionally employed to accomplish these challenges: increasing the level of abstraction and reuse.

The development of learning activities requires the specification of a great variety of aspects. Among the most relevant of them, we mention:

- the concept structure to be learnt,
- the media resources employed to represent these concepts,
- the mechanisms to manage, link and present these concepts,
- the activity workflow that should be followed to learn these concepts.

From the development perspective, all learning activities are different; however, they could share common aspects. For instance, different activities could employ the same interaction mechanisms (e.g. fill-in the gaps or joining concepts) to teach completely different concepts (e.g. fruits, vegetables, transportation, etc.).

This article proposes a conceptual model to develop activities in language learning applications. In particular, the article presents a new abstraction mechanism that allows designers to use (and reuse) the same model to represent many different learning activities, which have been taken from the learning methodologies under study. It defines the "fill-in the gaps" activity as universal abstraction to represent different kinds of activities. This abstraction is the basis of a model-driven approach to develop activities for different language learning methodologies.

This article also shows a series of examples where the "fill-in the gaps" abstraction is used to represent different kinds of activities for different learning language methodologies. Thus, the modelling process is simplified, since every activity is modelled using a common representation which favours the reuse of models. It is worth to note that this abstraction does not affect neither the visual presentation, nor the interaction of each activity.

This paper is organized as follows. Section 2 describes the research context together with the related work. Section 3 briefly describes the metamodel where the "fill-in the gaps" abstraction is defined. Section 4 analyses a set of additional

modelling capabilities derived from this proposal. Section 5 describes the users' evaluation carried out to evaluate the quality in use of our proposal as well as the product quality. Finally, we present conclusions and future works.

2 RESEARCH CONTEXT

This section contains the research context regarding the development of language learning applications, which includes two main elements: the essential concepts and features extracted from different learning methodologies to abstract the language learning process, and the most relevant related work in the field of the model-driven development which was employed to tackle the problems described in Section 1.

Learning a foreign language is a process involving different methods, techniques and tools, each one appearing to be more effective than the others. In this paper, we focus on methodologies that offer some type of technological support (Web site, mobile applications or similar).

We have analysed the following methodologies: Lexiway[2], Duolingo[3], Babbel[4] and Busuu[5].

Although these methodologies take different approaches, it is possible to identify some common elements.

The Model-driven Architecture (MDAs)[6] approach proposed by the Object Management Group (OMG) in 2011 presents a set of tools to abstract these common elements to improve the software development. This solution gives a leading role to models in the software development during all phases (i.e. inception, design, building, development, and maintenance).

The main reason behind this approach is the constant evolution of the software technologies. Following a traditional development approach, the functionality code and the implementation technology code are interweaved. Consequently, when the technology is enhanced, the functionality is rewritten using the new technology.

Under these scenarios, MDAs introduce abstraction levels to promote the software reuse by emphasizing the design-time interoperability [20]. This kind of interoperability is possible due to the specification of Platform Independent Models (PIMs) that enable developers to separate the specification of the application functionality from the technology that implements it.

Thus, it is possible to reuse the specification of the application functionality for different implementation technologies. Moreover, this functionality can be executed on different hardware and software platforms only with minor changes.

The source code of applications is automatically derived from models using model transformations [16].

[2] http://academialexiway.com/academia-ingles/metodo-academia-ingles.html
[3] https://www.duolingo.com/
[4] https://www.babbel.com/
[5] https://www.busuu.com/
[6] http://www.omg.org/mda/

In summary, the use of the MDA technology enables the generation of multi-platform applications from PIMs. This fact leads to several advantages; for example, let us assume we want to develop a learning activity using the fill-in the gaps interaction mechanism for different platforms (e.g. iOS, Web and Android). Following a traditional approach, we should develop 3 different and independent source code projects. Following an MDA approach, we specify only one PIM to generate the source code for the 3 platforms.

The core of the MDA infrastructure is defined in terms of the following OMG standards: the Unified Modeling Language (UML)[7], the Meta Object Facility (MOF)[8], XML Metadata Interchange (XMI)[9] and the Common Warehouse Metamodel (CWM)[10] which were successfully used in the modelling and development of modern systems.

From the Human-Computer Interaction perspective, we can find different approaches that makes use of models to generate user interfaces.

Hence, since our work focuses on the development of interactive systems, the Model-based User Interface Development (MbUID) provides useful elements to analyse based on the CAMELEON Reference Framework (CRF) [5].

In recent years, other approaches such as [11] have also encouraged the use of models to develop multi-modal user interfaces.

The use of Model-driven Development (MDD) for learning applications was applied in different works, such as those exposed in [8, 1, 18]. However, none of them formalizes the definition of language learning activities using OMG compliant metamodels. Nevertheless, there are several works that use MDD techniques based on MDAs to develop Web applications [17, 2].

An interesting approach that defines a MDA to develop music learning applications is exposed in [25]. In particular, in this approach a MDA-based System Development Lifecycle is defined, three Learn-Models are built, and the important developing phases are described. The idea of using submodels in a complex metamodel has been applied in our work. And a methodology to model e-learning applications can be found in [10]; however, this methodology does not focus on developing language learning applications. Unlike this approach, our approach presents a set of models at different abstraction levels providing different points of view of the application depending on the level of abstraction.

A work where the experience of different research groups working in formal and informal learning language design using mobile devices is presented in [4]. Among the most relevant works regarding the development of e-learning applications we can find those presented in [13, 19, 21, 24, 9].

With regard to the use of models to build Web sites, some methodologies (e.g. [7]) and models (e.g. RMM [14], WebML [6] have a direct impact on this research because they focus on modelling applications at the software level. However, our

[7] http://www.omg.org/spec/UML/2.5/PDF

[8] http://www.omg.org/spec/MOF/2.5.1/PDF

[9] http://www.omg.org/spec/XMI/2.5.1/PDF

[10] http://www.omg.org/spec/CWM/1.1

interest is focused on higher level of abstraction where the learning activity is the centre of our modelling interest.

WebML enables to define the high-level description of Web sites considering several orthogonal dimensions. For instance, the Web site contents (i.e. structural model), the Web pages that compose the Web site (i.e. composition model), the link topology among Web pages (i.e. presentation model), and the personalization characteristics enabling the one-on-one content delivery (i.e. personalization model).

The standard Interaction Flow Modeling Language (IFML)[11] is designed for expressing the content, user interaction and control behaviour of the front-end of software applications in general. In [3], authors describe how to apply model-driven techniques to the problem of designing the front end of software applications (i.e. the user interaction).

Our approach proposes a domain specific language based on the definition of a set of models of a higher level of abstraction presented in [22]. These models represent the interaction techniques used in language learning activities to maximize code reuse and minimize maintenance costs. This article presents the abstraction mechanism that allows designers to use (and reuse) the same model to represent many different learning activities, which have been taken from the learning methodologies under study.

3 METAMODEL TO DEVELOP LANGUAGE LEARNING APPLICATIONS

The goal of this article is the definition of a common representation to model language learning applications. To accomplish this goal, this section describes a metamodel that supports the representation of this type of applications. This metamodel is based on the metamodel presented in [23]. This description presents the modelling concepts to define applications as well as a set of examples showing how these concepts are assembled. The formalization of the concepts and the relationships among them were defined in ECORE[12] (Essential MOF dialect[13] enriched with expressions in OCL[14]. As result of the analysis of different methodologies, we have found a set of common elements.

The first element in common is the definition language concepts (e.g. words, sentences, etc.) that are hierarchically organized in lessons, units, etc.

The second element is the definition of different representations for these concepts. These representations are media resources (e.g. images, audio recordings, videos, text, etc.) that can be associated to methodology concepts. For instance, the "house" concept can be associated to an image of a house, a video of a house or an audio recording that contains the voice of a person pronouncing the word

[11] http://www.omg.org/spec/IFML/1.0/

[12] https://wiki.eclipse.org/Ecore

[13] http://www.omg.org/spec/MOF/2.5.1/PDF

[14] http://www.omg.org/spec/OCL/2.4

"house".

The third element is the definition of activities enabling users to interact with the application. There are several types of activities; for instance, multiple choice, filling the gaps and sentence composition activities.

The fourth element to take into account is the order in which the activities should be performed by the user. For instance, some methodologies only enable students to start a lesson if they have passed the previous one. The order in which activities are carried out is known as the methodology workflow. In summary, the common elements of language learning applications are:

1. the language concepts and concept hierarchy
2. the media resources
3. the learning activities
4. the activity workflow

To model these common elements, we leverage the level of abstraction and reuse following the MDA principle of interoperability at design time. Therefore, we define four concerns regarding the modelling of language learning methodologies.

Thus a learning methodology (represented by an instance of the *Methodology* metaclass that is part of the *Methodology* package) is composed by 4 models.

The language concept model, representing the language concepts, is modeled by an instance of the *ContentContainer* metaclass that is part of the *Content* package. The media resource model representing the resources used in the presentation (or view) of learning activitites, is modeled by an instance of the *MediaModel* metaclass that is part of the *Media* package. The model that defines the set of learning activities is represented by an instance of the *ViewModel* that is part of the *Presentation* package. And the activity workflow model is represented by an instance of the *Workflow* metaclass that is part of the *Workflow* package.

Fig. 1 shows the proposed metamodel exposing the packages of metaclasses that represent the language learning methodology common elements. The package structure for this metamodel is based on the *Methodology* package which uses the *Commons* package containing the *Entity* and *Property* metaclasses that are used as super-metaclasses for all metaclasses in the metamodel. The *Methodology* package is the core package of the model architecture, and contains the Methodology metaclass, and a set of packages that contains the metaclasses to represent all models (*Workflow*, *Content*, *Media*, *Activity* and *Presentation*). The sixth package, aka the *Commons* package, provides metamodel entities with extension features. Next paragraphs explain the most relevant metaclasses of each package.

The language concepts and the concept hierarchy are represented by instances of the metaclasses defined in the *Content* package. Fig. 1 presents the *Concept* and *ContentContainer* metaclasses of *Content* package. While *Concept* metaclass instances represent simple concepts, such as nouns (e.g. orange, apple, peaches) and verbs (i.e. stare, watch, glance); *ContentContainer* metaclass instances represent sets of related concepts (e.g. fruits, ways of looking, etc.). For example, in a given

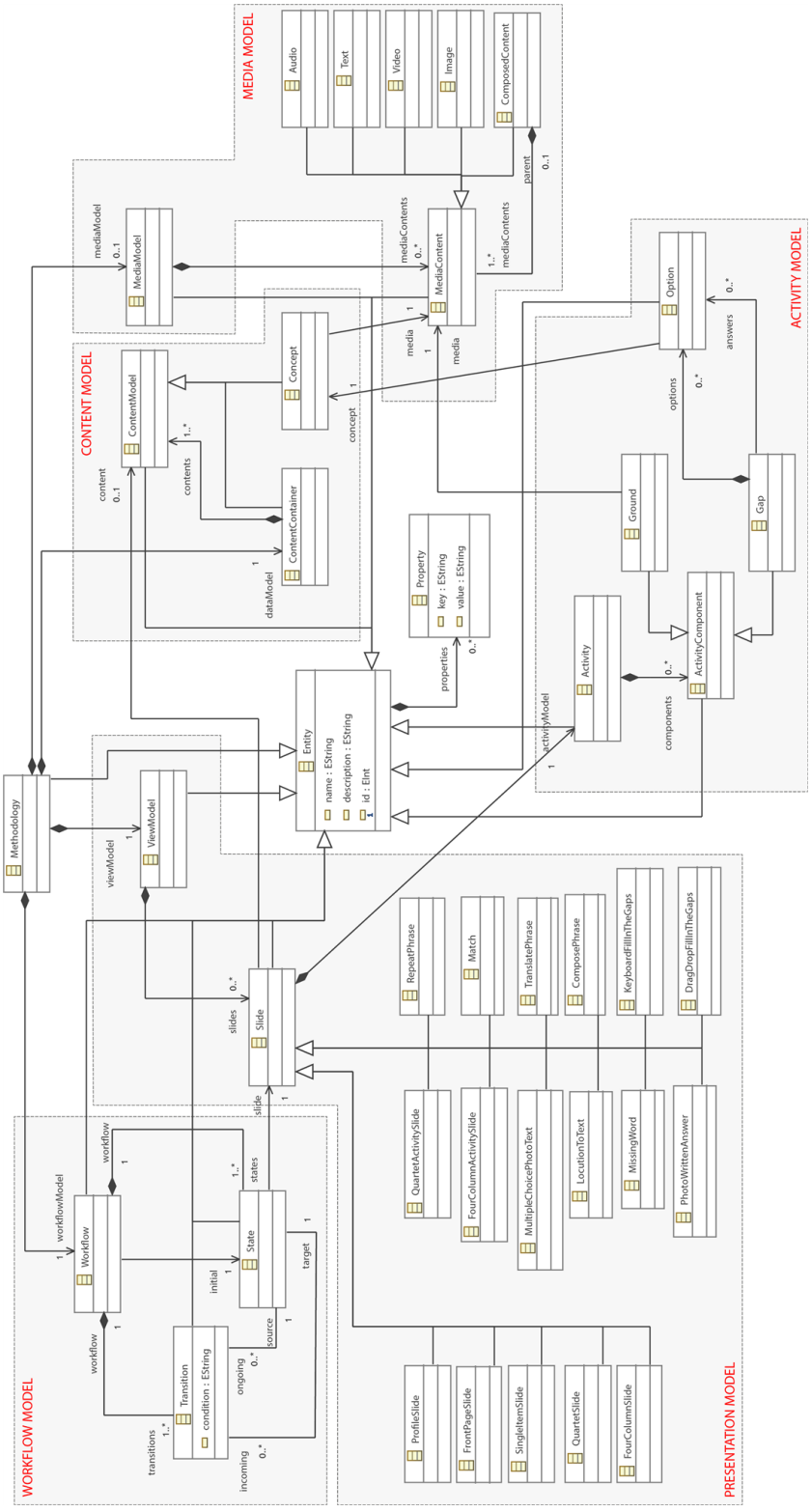


Fig. 1. Language learning methodology metamodel

methodology, a level can be composed of units and, a unit can be composed of lessons.

The *Media* package contains the metaclasses to represent the resources used in the presentation (or view) of learning activities. It enables developers to define 4 types of media resources: audio, text, video and image. These types of media are represented by *Audio*, *Text*, *Video* and *Image* metaclass instances. Combinations of this type of media can be combined to represent complex media resources such as text and speech. Again, the Composite design pattern [12] is applied to create a tree-based structure of media resources. The *MediaContent* metaclass plays the role of Component, the *ComposedContent* plays the role of Composite and the *Audio*, *Video*, *Image* and *Text* meclasses play the role of Leaves. The media relationship between the *Concept* metaclass and the *MediaContent* metaclass associates language concepts to media resources to provide these concepts with a concrete representation to activity presentations.

The definition of the activities of a learning methodology is organized into the *Activity* and the *Presentation* packages. On the one hand, the *Activity* package enables developers to parametrize the functionality of the activities conducted during the learning process. Every *Activity* metaclass instance provides users with information to perform learning activities. While *Ground* metaclass instances define activity statements represented by *MediaContent* metaclass instances; *Gap* metaclass instances define the information to be introduced by students. *Gap* metaclass instances are enriched with *Option* metaclass instances to define the potential information (including the correct answer to the activity) to be introduced by students. *Option* metaclass instances are related to language concepts that are linked to instances of the *MediaContent* metaclass which provide the presentation of the concept in the activity.

On the other hand, the *Presentation* package defines the *ViewModel* and the *Slide* metaclasses to represent the user interface and interaction mechanism of the activities offered to the students during the learning process. This package defines the concept of slide. It is defined by *Slide* metaclass instances that associate activities represented by *Activity* metaclass instances defined from the *Activity* package to specific interaction mechanisms. Consequently, students can interact with the same activity information using different interaction mechanisms (modalities), and vice-versa.

For instance, users can identify fruits matching images using the drag and drop or joining with lines interaction mechanisms. While an *Activity* metaclass instance contains the text for the statement of the activity; the options to be presented to the user and the option that solves the statement are represented by instances of the *Option* metaclass. And a *MultipleChoicePhotoText* metaclass instance defines the interaction mechanism.

The activity workflow defines the order in which learning activities should be performed by students, which is a crucial issue in the definition of learning methodologies. The *Workflow* package is responsible for representing this aspect of learning methodologies. This package defines the *Workflow* metaclass, whose instances de-

fine graphs. The nodes of the graph are defined by instances of the *State* metaclass, which are associated to instances of the *Slide* that represent them. The edges of the graph represent transitions (i.e. *Transition* metaclass instances) between states leading to transitions between learning activities.

Finally, the *Methodology* package defines the *Methodology* metaclass representing all the elements that define learning methodologies.

All the metaclasses in this metamodel inherit from the *Entity* metaclass defined in the *Commons* package which provides identification (i.e. *Entity* metaclass) and extension (i.e. *Property* metaclass) features to the rest of the metaclasses. This package is designed to take into account variable aspects of the activitites (aesthetical customization, look and feel of the user interface, structural limitations defined by a methodology, etc.).

Additionally, we have developed a language learning methodology model editor to create, edit and verify models according to the proposed metamodel. It was developed as an Eclipse plugin employing the Eclipse Modeling Framework (EMF)[15] to follow the MDA OMG standards. The metamodel was defined in OclInEcore[16] which is a dialect of the OMG Essential Meta-Object Facility (EMOF) enriched with OCL. This language is used to define the model invariants and queries that are the foundations for model verification.

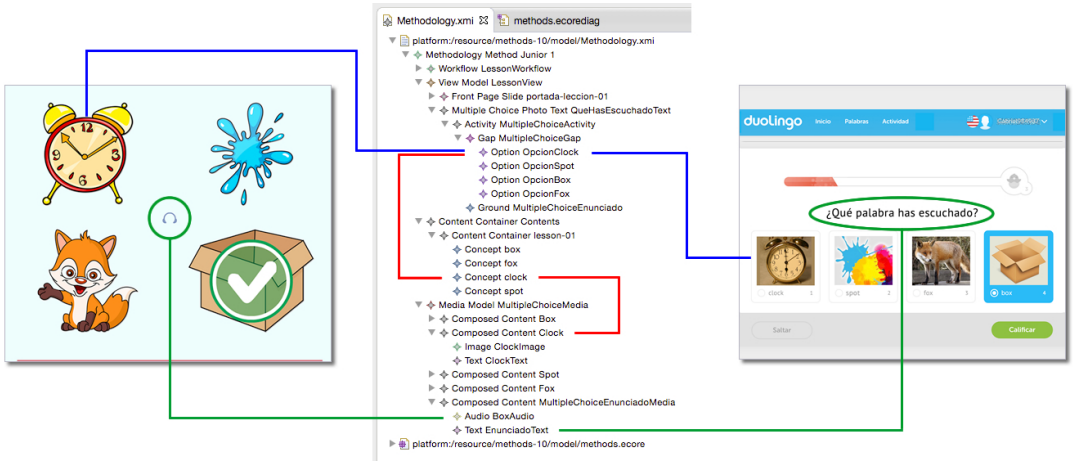


Fig. 2. Reusing of the multiple-choice learning activity model for Lexiway and Duolingo methodologies

3.1 Analysis of Language Learning Activity Modelling

In [22] we illustrate the flexibility and adaptability of the proposed metamodel to represent different language learning methodologies. Fig. 2 shows the correspon-

[15] http://www.eclipse.org/modeling/emf/

[16] https://wiki.eclipse.org/OCL/OCLinEcore

dence among the different elements of the model of a multiple-choice learning activity Lexiway as well as Duolingo. Thus, Fig. 2 illustrates the expressiveness power of this modelling tool, since the same model represents two similar activities in two different learning methodologies.

As we have mentioned, the user interfaces of learning activities are defined in the presentation model which is an instance the *ViewModel* metaclass. Each type of user interface is defined by a *Slide* sub-metaclass.

Learning activity user interfaces are customized with information provided by the activity model represented by an instance of the *Activity* metaclass. *Activity* metaclass instances define two types of *ActivityComponent* metaclass instances that composes the definition of an activity model. *Ground* metaclass instances represent fixed parts of the activity (e.g. parts of sentences, audio recordings, videos, etc.). *Gap* metaclass instances represent the user inputs to introduce information in the activity (e.g. an input field to type a word, an input area to type a sentence, a combo box to select a word, a list to select an image, etc.). In any case, *Gap* metaclass instances define a set of *Option* metaclass instances that represent the options that are associated to list or combo box items as well as references to the set of options that are considered correct answers to the activity. *Gap* and *Option* metaclass instances are linked to *Concept* metaclass instances to provide *Slide* sub-metaclass instances with multi-modal user interface representations.

This modelling approach enables developers to reuse different media models in different activities as well as provide customized different learning activities with different looks. For instance, you could provide customized media resources to adapt the learning activities to colour-blind people.

Besides, this approach also enables developers to reuse the same activity model in interaction mechanisms. For instance, the *Match* and *MultipleChoicePhotoText Slide* metaclass instances reuse the same activity model to present the same activity employing different interaction techniques.

The learning methodology activity workflow model enables developers to reuse learning activities in different learning paths. For instance, developers reuse the learning activities defined for the workflow of the Standard version of Lexiway in the workflow of the Junior version of Lexiway because the main difference between these two versions lays on the number of concepts that are presented to students on each lesson.

Finally, we expose different ways to extend the proposed metamodel. Firstly, the *Slide* metaclass can be extend to introduce new interaction mechanisms to learning activities. Secondly, metamodel entities represented by sub-metaclasses of the *Entity* metaclass defined in the *Commons* package can be extended by adding *Property* metaclass instances.

4 THE NEW ABSTRACTION TO MODEL LANGUAGE LEARNING ACTIVITIES

This section explains how to use the "Fill-in the Gaps" activity model as a universal abstraction to represent different language learning activities. Fig. 3 depicts the *Activity* package which contains the "Fill-in the Gaps" Activity metamodel used to model language learning activities.

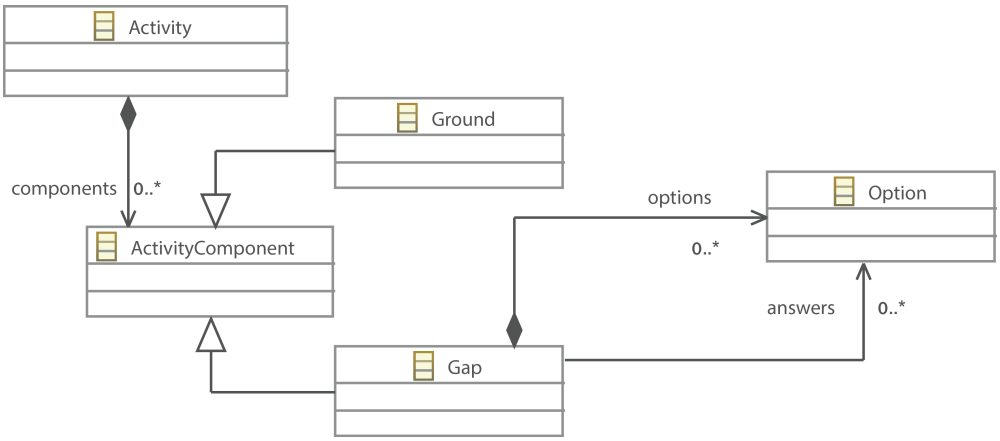


Fig. 3. Activity metamodel

As we have mentioned, this model links media resources (i.e. *MediaContent* metaclass instances) to *Slide* sub-metaclass instances using *Concept* metaclass instances as the glue between these two aspects.

4.1 Fill-in the gaps learning activity

The fill-in the gaps learning activity is a straightforward application of the fill-in the gaps abstraction. Fig. 4 depicts the model and presentation of a fill-in the gaps activity in the Bussu methodology. While the right side of the figure shows the actual user interface for the activity; the left side of the figure shows the model that represents the activity.

4.2 Word ordering learning activity

The word ordering learning activity asks students to order a set of words to compose a meaningful sentence. Fig. 5 shows the Busuu methodology version of this activity. The presentation model of this activity is defined by an instance of the *ComposePhrase* metaclass. The information to customize the activity is defined by an instance of the *Activity* metaclass.

In this case, the activity defines 4 gaps (represented by instances of the *Gap* metaclass) composed by 4 instances of 4 options (represented by instances of the

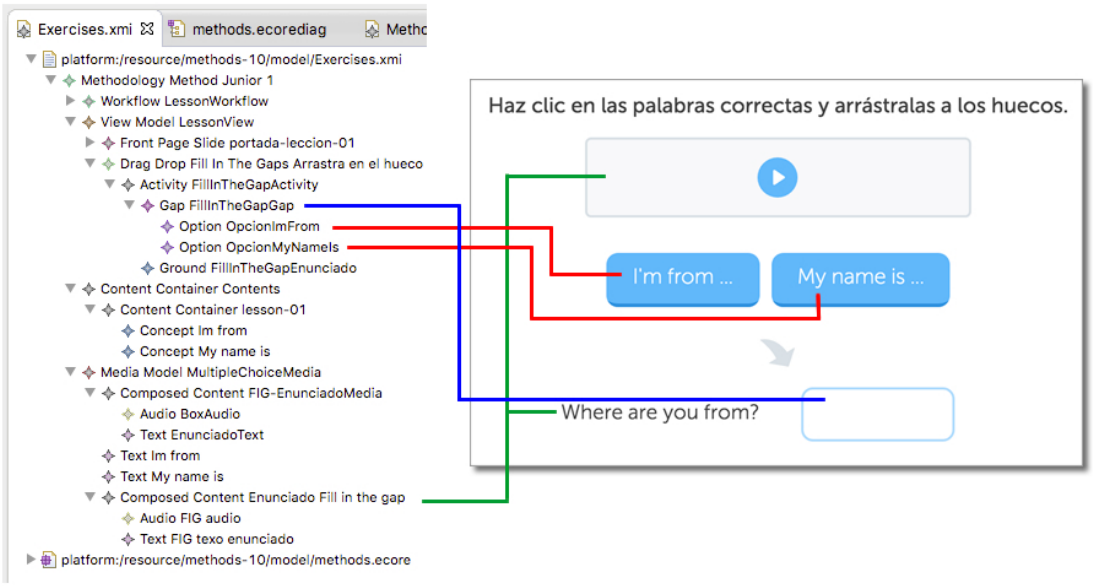


Fig. 4. Fill-in the gaps activity in Busuu

Option metaclass) for each gap. These options are linked to the same text strings (represented by instances of the *Text* metaclass) to enable users to choose one string (i.e. *are*, *Where*, *from?*, *you*) in any position of the sentence. The order of the gaps defines the order of the words in the sentence. And each gap defines only one option as the correct answer to set only one word ordering as correct. Finally, the *Ground* metaclass instance defines the learning activity statement. *Ground* metaclass instances can also be used as part of the sentence to include fixed words that cannot be modified by the user (e.g. punctuation marks, words, etc.).

4.3 Match-up learning activity

This section describes how to model a match-up learning activity in the Busuu methodology using the fill-in the gaps abstraction. Fig. 6 depicts the learning activity user interface which consists in locating and matching up the 3 elements on the left with the corresponding 3 elements on the right.

In this case, the presentation of the activity is defined by an instance of the *Match* metaclass. The activity information is modelled as a fill-in the gaps exercise including 3 gaps representing the 3 elements depicted on the right side of the figure (i.e. *¿Cómo te llamas?*, *¿De dónde eres?*, *¿Cuántos años tienes?*). Each of gap presents the same 3 options to be linked to the 3 elements on the left side (*How old are you?*, *Where are you from?*, *Whats your name?*) where only one of these options is defined as the correct answer. As in the previous example, the options (represented by instances of the *Option* metaclass) are linked to instances to the *Text* metaclass.

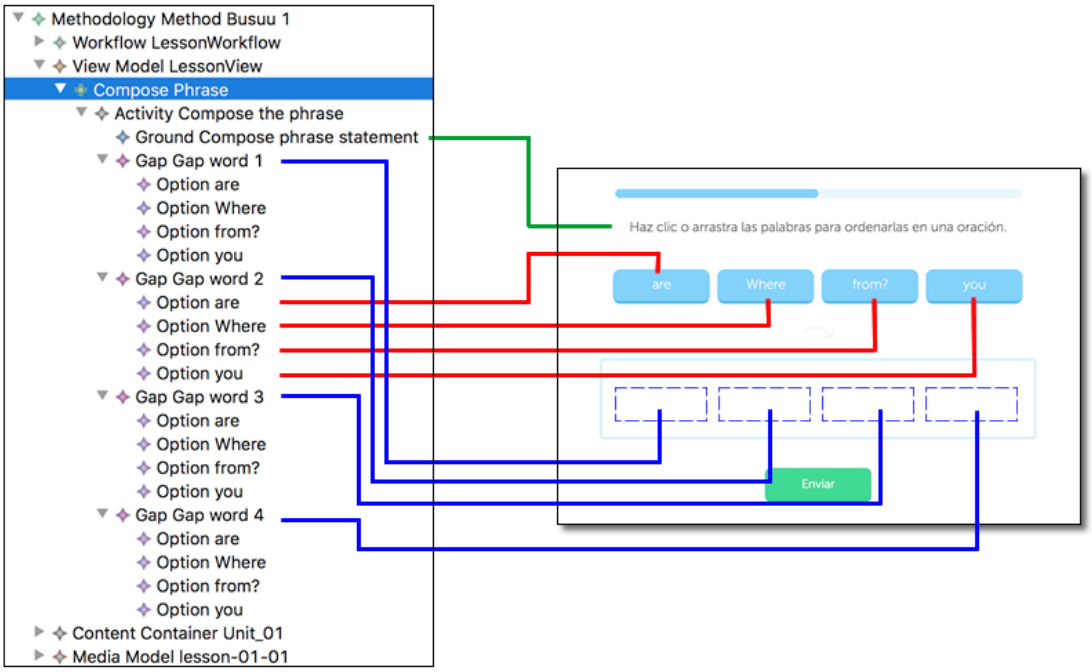


Fig. 5. Word ordering learning activity in Busuu

4.4 Locution to text, multiple choice and translate phrase learning activities in Duolingo

Fig. 7, Fig. 8 and Fig. 9 depict 3 learning activities that illustrate the similarities of the activity models in different learning activities in the Duolingo methodology. All the activity models of these learning activities define only one gap (instance of the *Gap* metaclass) including several options (instances of the *Option* metaclass) where at least one of them is set as the correct one.

These models also define instances of the *Ground* metaclass to represent the statement of the activity (e.g. *Escucha y escribe*, *Marca todas las respuestas correctas*, *Traduce este texto*). The *Ground* and *Option* metaclass instances are linked to media resources represented by instances of the *ComposedContent* or *Text* metaclasses. The *ComposedComponent* metaclass instances enables developers to provide different types of media resources (e.g. *Text* and *Audio* metaclass instances).

Finally, the locution to text, multiple choice and translate phrase learning activities are represented by instances of the *LocutionToText*, *MultipleChoice* and *TranslatePhrase* metaclasses respectively.

4.5 Multiple choice learning activity in Babbel, Bussu and Duolingo

The modelling process can be analogously applied to model the information or content of the same learning activity for different methodologies.

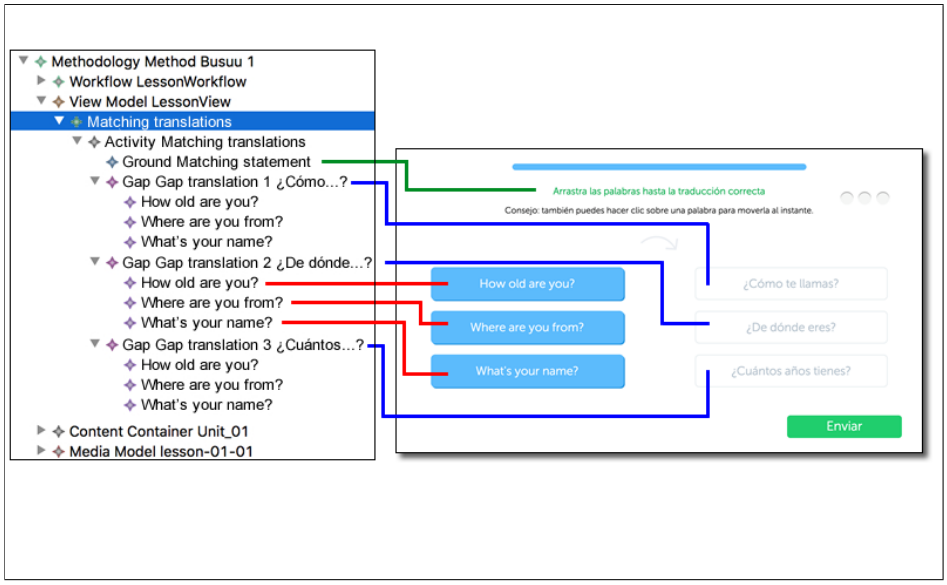


Fig. 6. Match-up learning activity in Busuu

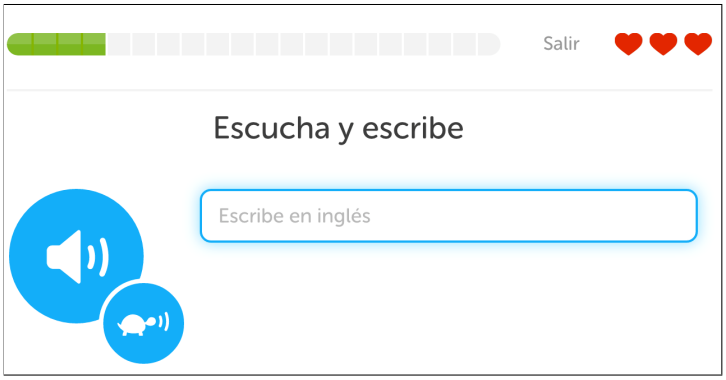


Fig. 7. Locution to text learning activity in Duolingo using one gap

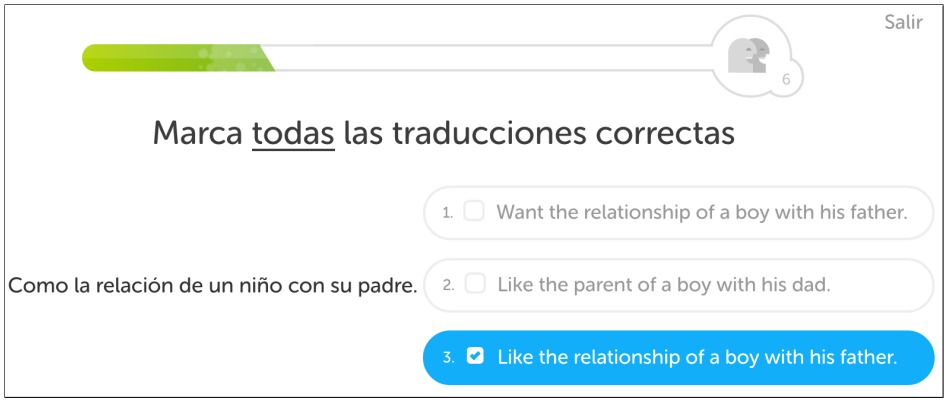


Fig. 8. Multiple choice learning activity in Duolingo using one gap

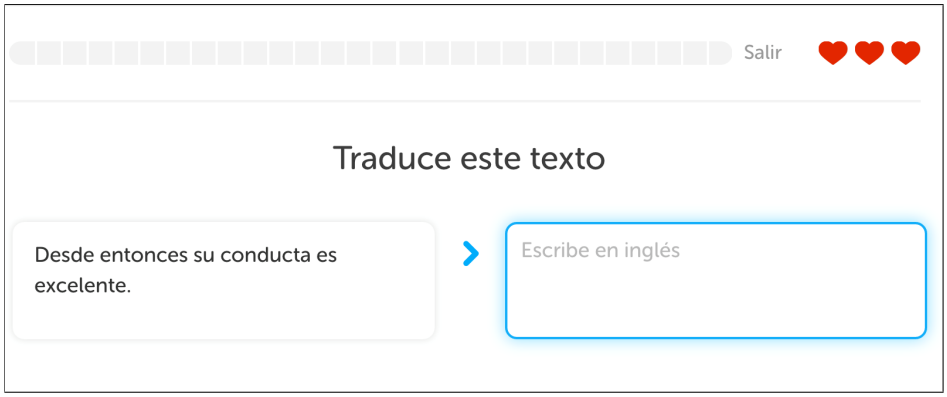


Fig. 9. Translate phrase learning activity in Duolingo using one gap

Fig. 10, Fig. 11 and Fig. 12 depict 3 examples of multiple choice activities in 3 different learning methodologies. (i.e. Babbel, Busuu and Duolingo). All these examples represent the activity exposed in Section 3 and depicted in Fig. 2.

Fig. 10. Multiple choice learning activity in Babbel

All these 3 examples define one gap with 3 options. However, each exercise defines different kinds of media resources to represent the activity statement and options. For instance, while Fig. 10 and Fig. 12 depict Babbel and Doulingo version of a multiple choice activity using 3 pictures to represent gap options; the Bussu version of the activity depicted in Fig. 11 uses a video to represent the activity statement.

Therefore, the idea of employing the fill-in the gaps activity as an abstraction for all kinds of learning activities simplifies the modelling process since all activities are modelled in the same way which favours the reuse of models.

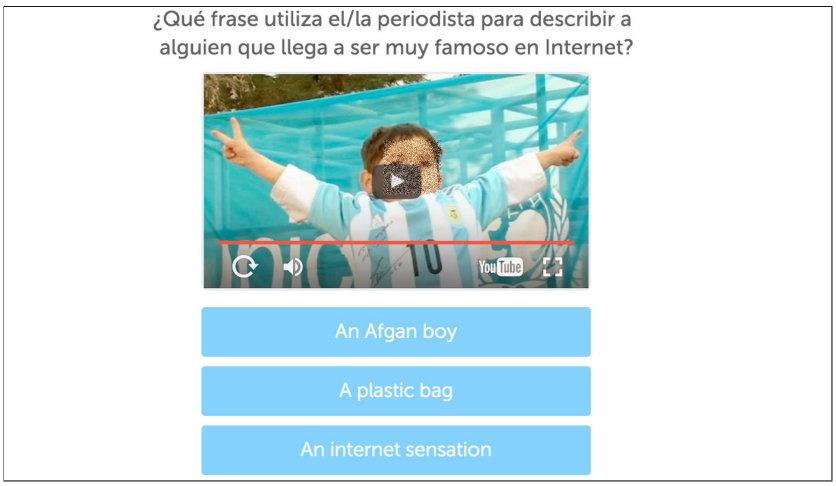


Fig. 11. Multiple choice learning activity in Busuu

Fig. 12. Multiple choice learning activity in Duolingo

5 EVALUATING THE PROPOSAL

The main goal of the evaluation section is to evaluate the quality of the code obtained after applying the MDA approach proposed in this article (product quality evaluation). The quality of the code is evaluated by comparing the code obtained using the MDA approach against the code generated by an expert (called traditional approach). The comparison is performed using a set of well-known quality factors. Before performing the product quality evaluation, we have to prepare the artefacts using both approaches. The next subsection describes the preparation process.

5.1 Preparation Process

This section describes the process followed to generate the artefacts (learning activities) that will be compared in the next subsection.

There are two different mechanisms to obtain the learning activities: the pro-

posed approach and the traditional approach.

5.1.1 Objectives of the Preparation Process

The objective of the preparation is to develop a set of learning activities considering both the proposed and traditional approaches.

5.1.2 Participants of the Preparation Process

This evaluation is carried out by two participants of different profiles.

The first participant (male, age 23, university graduated) is expert in HTML, CSS and JavaScript technologies and develops learning activities following a traditional approach (HTML expert).

The second participant (male, age 41, Phd student) is an expert in the Eclipse Modeling Framework (EMF) technology designed for model-driven development and develops learning activities following the proposed approach (MDA expert).

5.1.3 Computing environment

Each participant performed the preparation test in the ISE Research Group Interaction laboratory located in the Albacete Research Institute of Informatics (I3A) building in Albacete, Spain. This location is equipped with computers and multimedia equipment (e.g. video cameras, microphones, and so on) that makes it suitable for performing HCI (Human-Computer Interaction) interaction evaluations.

Both development processes were performed using the same computing equipment. The hardware consists of a MAC Book Pro 13" Retina laptop computer with 8 GB RAM and 256 GB SSD. This computer runs the High Sierra iOS, SublimeText (ver. 3.0), and the Eclipse Modeling Tools NEON 3 IDE including ATL (ver. 3.6.6), ACCELEO (ver. 3.7.0) and the proposed approach reflexive model editor (ver. 1.0.0) and transformation (ver. 1.0.0) plugins.

5.1.4 Tasks of the Preparation Process

Participants receive the same list of requirements proposing the development of four Lexiway learning activities to review student language vocabulary. These activities look like the learning activity depicted on the left side of Fig. 2 that presents four images and plays the audio file of a word when it starts. Learners should click on an image corresponding to the word that was played. When an image is clicked, they receive the result of the matching in terms of negative or positive reinforcement. If the clicked image does not correspond to the audio played, the file is played again and the user is asked to click on an image again until they choose the correct image.

Each participant followed a different path to develop the learning activities. The MDA Expert had to define a model for each activity, validate the model and generate the code. The HTML Expert had to use his favourite HTML editor, locate the resources, write the code, and test the solution.

However, it is possible to identify some general tasks, no matter the tools used to get them. The development of each learning activity represents a task. Each task is divided into the sub-tasks, which are defined in Table 1.

Table 1. Common tasks performed by the participants

T	Description
1	Define the graphic design of the user interface for the learning activity presented on the left side of Fig. 2 using the set of images defined in a specific folder. 1. Slide option images (e.g. clock, spot, fox and box) 2. Common images (e.g. headphones, correct, wrong)
2	Define the audio modality of the user interface for the learning activity presented on the left side of Fig. 2 using audio files defined in a specific folder. 1. Possible slide statement locutions (e.g. clock, spot, fox and box) 2. Common sounds (e.g. correct and wrong answers)
3	Define the learning activity statement linking the click event on the headphones image to one of the possible locutions for the activity
4	Define the learning activity answer linking answer images to option images according to the selected statement locution for the activity (e.g. the correct answer image for the learning activity depicted on the left side of Fig. 2 is box and the rest of options are linked to the wrong image)
5	Define the learning activity answer linking answer sounds to option images according to the selected statement locution for the activity (e.g. the correct answer sound for the learning activity depicted on the left side of Fig. 2 is box and the rest of the options are linked to the wrong sound)
6	Define the learning activity behaviour when learners click on the wrong answer (i.e. play the statement locution again)

5.1.5 Review and Testing

Both participants knew that the learning activities they developed, would be analyzed by a group of experts. Therefore, they spent some time to check the product obtained. This process was carried out in different ways by both participants, since

the tools used in each case were different. In the case of the HTML Expert, this process includes activities like refactoring, refining, testing, and so on. In the case of the MDA Expert, this process consists on model review, validate the OCL restrictions, operate the transformations and review the results.

5.2 Product Quality

This evaluation analyses the quality of language learning activity source codes generated with the proposed and traditional approaches, obtained in the previous section. To carry out this task, we propose an heuristic evaluation where a set of 5 experts evaluates 4 software attributes related to software quality characteristics.

5.2.1 Objective

This heuristic evaluation compares the source code quality of the language learning application depicted on left side of Fig 2 generated with the traditional and proposed development processes in terms of software *correctness*, *robustness*, *extensibility* and *reusability*; where software *correctness* refers to the Functional Correctness sub-characteristic of the Functional Suitability characteristic defined in the Product quality model of the ISO 25010:2011(E) standard [15], software *robustness* refers to the Fault tolerance and Recoverability sub-characteristics of the Reliability characteristic defined in the Product quality model of the ISO 25010:2011(E) standard [15], software *extensibility* refers to the Modularity and Modifiability sub-characteristics of the Maintainability characteristic defined in the Product quality model of the ISO 25010:2011(E) standard [15], and software *reusability* refers to the Reusability subcharacteristic of the Maintainability defined in the Product characteristic quality model of the ISO 25010:2011(E) standard [15].

5.2.2 Participants

This evaluation is performed with 5 experts, whose profiles are exposed in Table 2. Participant profiles include information such as gender, age, and experience in HTML, JavasScript, Language Learning Applications and Software Quality.

Table 2. Participant profiles

Participant	**Gender**	**Age**	**Experience**			
			HTML	**JavaScript**	**L. L. Apps.**	**Soft. Quality**
1	M	38	5	5	3	5
2	F	35	5	4	5	4
3	M	39	5	5	3	4
4	M	45	4	5	5	4
5	F	48	5	5	4	5

5.2.3 Computing Environment

This evaluation was carried out in the ISE Research Group interaction laboratory located in the Albacete Research Institute of Informatics (I3A) building in Albacete, Spain. This location is equipped with computers and multimedia equipment (e.g. video cameras, microphones, and so on) that makes it suitable for performing HCI interaction evaluations.

The evaluation was performed on a Dell XPS 702x laptop computer running Microsoft Windows 10. The Internet browser used to run both implementations is Chrome version 64.

5.2.4 Metrics

The metrics to evaluate software *correctness*, *robustness*, *extensibility* and *reusability* are scored from 1 to 5 according to experts' criteria where 1 and 5 represents the lowest and highest scores of the software product for an specific attribute respectively.

5.2.5 Procedure

The evaluation procedure starts when participants receive the source codes of the language learning activity (both, traditional and proposed) presented on the left side of Fig. 2, and a form to score these source codes in terms of selected software attributes as well as an extra section to justify the software product scoring. It is worth to highlight that how source codes were generated is unknown to participants.

The W3C Validator [17] and the JSHint[18] tools are available to participants to help them to score the software product.

5.2.6 Results

The overall results of the comparison between the traditional approach and the proposed approach are exposed in Table 3.

According to experts, the proposed approach based on models overcomes the traditional approach because while the proposed approach scores 4.75 out of 5 in the overall scoring, the traditional approach only obtained 3.4 out of 5. Moreover, the scores of the proposed approach are higher than 4 out of 5 in all evaluated software attributes.

The standard deviation on the proposed approach also delivers scores on all attributes are close to the average score which is above 4.6 out of 5 showing an homogeneous consensus on the experts.

The proposed approach does not only overcomes the traditional approach in the overall results; it also overcomes it all evaluated software attributes.

[17] `https://validator.w3.org/`

[18] `http://jshint.com/about/`

Table 3. Heuristic evaluation results

Part.	Approach	Correctness	Robustness	Extensibility	Reusability	Average
1	Traditional	3	4	2	3	**3**
	Proposed	5	5	5	5	**5**
2	Traditional	3	3	3	3	**3**
	Proposed	5	5	5	5	**5**
3	Traditional	5	2	1	2	**2.5**
	Proposed	5	5	4	4	**4.5**
4	Traditional	5	5	3	5	**4.5**
	Proposed	4	5	5	5	**4.75**
5	Traditional	5	4	3	4	**4**
	Proposed	5	4	4	5	**4.5**
Average	Traditional	**4.2**	**3.6**	**2.4**	**3.4**	**3.4**
	Proposed	**4.8**	**4.8**	**4.6**	**4.8**	**4.75**
Std. Dev.	Traditional	1.09	1.14	0.89	1.14	
	Proposed	0.44	0.45	0,55	0.45	
Max	Traditional	5	5	3	5	
	Proposed	5	5	5	5	
Min	Traditional	3	2	1	2	
	Proposed	4	4	4	4	

The score is even more significant when evaluating the *extensibility* of the software product (it almost doubles the score of the traditional approach). Participants also highlighted the quality of the code using the traditional approach is more difficult to extend than the code generated using the proposed approach avoiding the a great impact on code modifications.

Although the proposed approach obtains a higher score than the traditional approach in terms of *correctness*; the average score in this subject for the traditional approach is really good obtaining a difference less than 0.6 with a standard deviation of 1.09 with respect to the proposed approach.

Comparing both approaches in terms of *robustness*, participants state that conditional structures in the code generated using the traditional approach are not as well-structured as in the proposed approach.

About the code *reusability*, they mention that the code generated using the proposed approach organizes multimedia resources (i.e. media files, JavaScript and Cascade Style Sheets) more efficiently than the code generated using a traditional approach, because the proposed approach groups common resources to all activities encouraging their reuse. Moreover, all participants highlight the code structure generated using the proposed approach because it defines parametrized functions encouraging their reuse too.

6 CONCLUSIONS

This article proposes the fill-in the gaps abstraction to model language learning activities in the context of a MDA to develop language learning applications.

It shows, by means of a series of examples, how the fill-in the gaps abstraction is

used to represent different learning activities in different language learning methodologies. These examples show how the modelling process is simplified since activity models can be easily reused to:

1. customize the interaction mechanism of the learning activity
2. adapt the application look to users' needs (i.e. colour-blind people)
3. model the same activity for different learning methodologies

Traditional approaches force developers to build applications for different platforms (e.g. Web, iOS, Android, Windows, etc.) leading to different development branches which are prone to errors, difficult to maintain and test (e.g. changes and fixes on the application domain model should be addressed in all platforms).

Model-driven architectures decouples application functionality from technology which enable developers to create Platform Independent Models (PIMs) and Platform Specific Models (PSMs) to derive application source code semi-automatically using model transformations.

One of the main features of employing MDAs is the design time interoperability. This feature captures different application concerns in independent models which are integrated at the last stage of development (just before the generation of the application source code).

This proposal defines 5 concerns regarding the development of activities for language learning methodologies (i.e. learning methodology contents, learning activity workflows, learning media resources, learning activity interaction mechanisms, and learning activity model). The main advantage behind this feature is the capability of modifying the model with minimum impact on the others. For example, the representation of a concept (i.e. and image) can be changed by modifying only the media resource model without affecting the other models (i.e. content, workflow, interaction mechanism or activity model).

Besides, we have made progress in the development of model-to-model and model-to-text transformations to generate the source code of language learning activity applications using the ATL (ATL Transformation Language)[19] and Acceleo[20] transformation languages respectively.

We also performed a users' evaluation to analyse the product quality of our proposal. The product quality analyses the product correctness, robustness, extensibility, and reusability. These results conclude that the code generated using the proposed approach overcomes the code generated using a traditional approach in the selected metrics.

The future work for this project includes the development of graphic modelling editor using the GMF framework to create, edit and verify learning methodology models generated with our metamodel. We are working on extensions to the proposed metamodel that allow improving the validation of the models through the def-

[19] `https://eclipse.org/atl/`

[20] `https://www.eclipse.org/acceleo/`

inition of customized OCL expressions that are loaded dynamically (using the Complete OCL plugin [21]). These extensions are defined based on the sub-classification of meta-classes, which allow introducing new features in a flexible and robust way.

REFERENCES

[1] Bizonova, Z. Model driven e-learning platform integration. In *Proceedings of the EC-TEL 2007 PROLEARN Doctoral Consortium, Crete, Greece, September 18, 2007* (2007), K. Maillet, T. Klobucar, D. Gillet, and R. Klamma, Eds., vol. 288 of *CEUR Workshop Proceedings*, CEUR-WS.org.

[2] Blumschein, P., Hung, W., Jonassen, D., and Strobel, J. *Model-Based Approaches to Learning: Using Systems Models and Simulations to Improve Understanding and Problem Solving in Complex Domains*. 2009.

[3] Brambilla, M., and Fraternali, P. *Interaction Flow Modeling Language: Model-Driven UI Engineering of Web and Mobile Apps with IFML*, 1st ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2014.

[4] Brcena, E. State of the art of language learning design using mobile technology: sample apps and some critical reflection. In *Proceedings of the 2015 EUROCALL Conference, Padova, Italy (2015)* (2015), pp. 36–43.

[5] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., and Vanderdonckt, J. A unifying reference framework for multi-target user interfaces. 289–308.

[6] Ceri, S., Fraternali, P., and Bongio, A. Web modeling language (webML): A modeling language for designing web sites. In *Ninth International World Wide Web Conference* (Amsterdam, Netherlands, may 2000), Elsevier.

[7] Conallen, J. *Building Web Applications with UML*, 2nd ed. Addison Wesley, Reading, Massachusetts, oct 2002.

[8] Conn, S., and Forrester, L. Model driven architecture: A research review for information systems educators teaching software development.

[9] et al., J. M. D. Development of e-learning solutions: Different approaches, a common mission. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje 9*, 2 (2014), 72–80.

[10] Fardoun, H., Simarro, F. M., and López-Jaquero, V. eLearniXML: Towards a model-based approach for the development of e-learning systems considering quality. *Advances in Engineering Software 40*, 12 (2009), 1297–1305.

[11] Feuerstack, S., and Pizzolato, E. B. Engineering device-spanning, multimodal web applications using a model-based design approach. In *WebMedia* (2012), G. Bressan, R. M. Silveira, E. V. Munson, A. Santanchà, and M. da Graça Campos Pimentel, Eds., ACM, pp. 29–38. http://dl.acm.org/citation.cfm?id=2382636.

[12] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison-Wesley, Massachusetts, 1995.

[21] https://marketplace.eclipse.org/content/eclipse-ocl

[13] García, F. *Advances in E-Learning: Experiences and Methodologies*. Information Science Reference, 2008.

[14] Isakowitz, T., Stohr, E. A., and Balasubramanian, P. RMM: A methodology for structured hypermedia design. *Communications of the ACM 38*, 8 (aug 1995), 34–44.

[15] ISO. ISO/IEC 25010. Systems and software engineering - Systems and software engineering Quality Requirements and Evaluation (SQuaRE) - Systems and software quality models, 2006.

[16] Kleppe, A., Warmer, J., and Bast, W. *MDA Explained: The Model Driven Architecture*. 2003.

[17] Koch, N., and Kraus, A. *Towards a Common Metamodel for the Development of Web Applications*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, pp. 497–506.

[18] Lahiani, N., and Bennouar, D. A model driven approach to derive e-learning applications in software product line. In *In Proceedings of the International Conference on Intelligent Information Processing, Security and Advanced Communication (IPAC 15)* (2015).

[19] Lanzilotti, R., Ardito, C., Costabile, M. F., and Angeli, A. D. eLSE methodology: A systematic approach to the e-learning systems evaluation. *Educational Technology & Society 9*, 4 (2006), 42–53.

[20] Mellor, S., Scott, K., Uhl, A., and Weise, D. *MDA Distilled: Principles of Model-Driven Architecture*. Addison Wesley, 2004.

[21] Rachid Dehbi, M. T., and Tragha, A. A model driven methodology approach for e-learning platform development. *International Journal of Information and Education Technology 3*, 1 (2013), 10–15.

[22] Rivera, G. S., Tesoriero, R., and Gallud, J. A. A model-based approach to develop learning exercises in language-learning applications. *IET Software* (February 2018).

[23] Sebastian, G., Tesoriero, R., and Gallud, J. A. Modeling language-learning applications. *IEEE Latin America Transactions 15*, 9 (2017), 1771–1776.

[24] Tang, S., and Hanneghan, M. A model-driven framework to support development of serious games for game-based learning. In *Developments in E-systems Engineering* (2010), pp. 95–100.

[25] Tian, Y., Yang, H., and Landy, L. MDA-based development of music-learning system. In *Proceedings of the 14th Chinese Automation & Computing Society Conference, UK, Brunel University, West London* (2008), S. Zhang and D. Li, Eds., pp. 97–102.

Capítulo 7

TagML: An Implementation Specific Model to Generate Tag-based Documents

Citación: R. Tesoriero, G. Sebastián, J. A. Gallud, TagML: An Implementation Specific Model to Generate Tag-based Documents, ACM Transactions on the Web.

Tipo de publicación: Revista Internacional (IF: 1.580, Q2)

(Enviado)

IEEE SOFTWARE, VOL. 14, NO. 8, AUGUST 2015

TagML: An Implementation Specific Model to Generate Tag-based Documents

Ricardo Tesoriero, Gabriel Sebastián and Jose A. Gallud

***Abstract*—This paper describes TagML, a method to generate XML documents using model-to-model (M2M) transformations. To accomplish this goal, we define the TagML meta-model and the TagML-to-XML model-to-text transformation. While TagML models represent the essential characteristics of XML documents, the TagML-to-XML transformation generates the textual representation of XML documents from TagML models. This approach enables developers to define model-to-model transformations to generate TagML models. These models are turned into text applying the TagML-to-XML transformation. Consequently, developers are able to use declarative languages to define model-to-text transformations that generate XML documents, instead of traditional archetype-based languages to define model-to-text transformations generating XML documents. The TagML model editor as well as the TagML-to-XML transformation were developed as Eclipse plug-ins using the Eclipse Modeling Framework. The plugin has been developed following the Object Modeling Group standards to ensure the compatibility with legacy tools. Applying TagML, unlike other previous proposals, implies the use of model-to-model transformations to generate XML documents instead of model-to-text transformations, which results on an improvement of the transformation readability and reliability, as well as a reduction of the transformation maintenance costs. The proposed approach helps developers to define transformations less prone to errors than using the traditional approach. The novelty of this approach is based on the way XML documents are generated using model-to-model transformations instead of traditional model-to-text transformations. Moreover, the simplicity of the proposed approach enables the generation of XML documents without the need for any transformation configuration, which does not penalize the model reuse. To illustrate the features of the proposal, we present the generation of XHTML documents using UML class diagrams as input models. The evaluation section demonstrates that the proposed method is less prone to errors than the traditional one.**

***Index Terms*—Tag languages, meta-model, M2T transformation, M2M transformation**

I. Introduction and Background

THE number of user interface description languages that use markup languages has increased during the last years. The main reason behind their growth is their use in the Web.

The Extensible Markup Language (XML) [1] is one of the most popular markup languages. For instance, the Extensible HyperText Markup Language (XHTML) [2] enables developers to define static HyperText Markup Language (HTML) [3] pages as valid XML documents.

The use of XML is not limited to the definition of static Web pages, most technologies supporting the generation of dynamic Web pages use XML-based languages; for instance, Java Server Pages (JSP) [4], Java Server Faces (JSF) [5], Standard Tag Library for JavaServer Pages (JSTL) [6], ASP.NET Web Pages [7], Extensible Application Markup Language (XAML) [8], Polymer components [9], AngularJS directives [10], Extensible 3D (X3D) [11], etc.

The use of XML to define user interfaces is not limited to the Web; it also used to define user interfaces in mobile platforms such as the Android [12] platform. It is neither limited to the description of UIs, the XML language is also employed to represent the information exchanged in Web Services. For instance, the Simple Object Access Protocol (SOAP) [13] defines the Web service communication using XML documents.

Another example that exposes the versatility of XML to represent information is the definition of the XML Model Interchange (XMI) [1] format which is the standard format defined by the Object Management Group (OMG) [14] that enables the use and manipulation of model information in Model-driven Architectures (MDAs) [15].

An interesting capability of XML is definition of XML meta-data in XML documents. For instance, the XML Schema Definition (XSD) [16] language is a XML-based language to specify the structure of valid XML documents. Besides, the XML Stylesheet Language Transformation (XSLT) [17] language enables developers to define XML document transformations in XML.

Model-driven Architectures (MDAs) enables developers to generate Platform Specific Models (PSMs) using Platform Independent Models (PIMs) as input parameters of model-to-model transformations. These PSMs are used as input parameters of model-to-text transformations to generate the source code of the application.

Most popular model-to-text transformation languages tend to adopt a archetype-based paradigm. Some examples of this type of language are ACCELEO [18], MOFScript [19], Java Emitter Templates (JET) [20], etc. These languages use text templates interwoven with OCL expressions. While the text templates generate static text; the OCL expressions generate dynamic text. They enable developers to define transformations any language or purpose.

This article describes a proposal that captures XML characteristics to enable developers to use model-to-model transformation languages to generate XML documents. Unlike most popular model-to-text transformation languages, model-to-model transformation languages, such as Query View Transformation (QVT) [21] or Atlas Transformation Language (ATL) [22], tend to adopt a declarative paradigm where rules are defined using OCL expressions.

G. Sebastián is researcher at the Albacete Research Institute of Informatics in Albacete, Spain.

R. Tesoriero and J. A. Gallud are professors of the Computing Systems Department belonging to the University of Castilla-La Mancha in Albacete, Spain.

Manuscript received April 19, 2005; revised August 26, 2015.

The level of abstraction is leveraged introducing the TagML meta-model to capture the essential properties of XML to model XML documents. In addition, the level of reuse is improved by defining the TagML-to-XML model-to-text transformation that generates XML documents using the TagML models.

The novelty of this approach lays on the way XML documents are generated using model-to-model transformations instead of traditional model-to-text transformations. Consequently, junior developers do not have to learn another programming paradigm to write model-to-text transformations to generate this kind of documents reducing costs (it is assumed that they already have skills in model-to-model transformation languages which usually employ declarative programming paradigms). Moreover, the simplicity of the proposed approach enables the generation of XML documents without the need for any transformation configuration. This fact does not penalize model reuse due to the proposed generation process.

Our proposal enable developers to generate XML documents using model-to-model transformations. To carry out this task, we encourage the interoperability of the architecture at design time following the MDA principles of leveraging the level of abstraction and reuse [23].

In [24], the authors argue for the importance of integrating XML-based models in the MDE process. In that article, the authors propose a lightweight approach for providing first-class support for managing XML documents within Epsilon model management language family [18] using the EOL [25] OCL-based imperative language to manipulate models. The philosophy behind our approach is the definition of an Implementation Specific Model (ISM) to deal with XML document generation using a declarative programming paradigm to make this process as simple as possible for junior developers (e.g. our approach does not require Epsilon to work). Though the goal and purpose of both approaches may be similar, the perspective to board the problem is completely different (i.e. imperative versus declarative paradigm to define model transformations).

Main difference between the study presented in [26] and our approach is the level of abstraction they use to address similar problems. While [26] formalizes templates engines; our approach is focused on the definition of an Implementation Specific Metamodel and a generation process for the specific purpose of generating tag-based documents using a model-to-model transformation engine (ATL) [22] instead of a model-to-text transformation engines that are traditionally employed to generate this type of documents.

Unlike the proposal presented in [27] where a concrete instantiation of a framework employing string, tree or triple graph grammars is required to generate source code; our approach only requires the definition of a model-to-model transformation to generate code because it addresses a more specific problem: the generation of tag-based documents only. As result, the generation process is simpler requiring only developers with skills EMF and ATL to generate code (i.e. any junior developer with basic skill in MDA can handle it).

We share the view presented in [28] where authors propose the use of small Domain Specific Languages (DSLs) well-focused on metamodels, instead of monolithic languages like UML. The authors illustrate this approach to solve the bug-tracking by defining a metamodel to represent XML documents in Section 4.1. Although the metamodel presented in [28] seems to be similar to the metamodel presented in this proposal, the purpose is completely different. While the goal of our approach is the definition of an Implementation Specific Model (ISM) to generate XML documents for application development; the goal of the metamodel presented in [28] is the definition of a generic document representation to pivot among different file formats. For instance, the metamodel presented in [28] lacks on the capability to define file and folder structures to model the location of XML documents. This may not be a major issue if you are dealing with document formats; however, it is a minor issue in code generation because this characteristic is present in all model-to-text transformation languages. Finally, authors in [28] do not provide any reference to the transformation that generates XML text documents, which is understandable because it is not the paper focus.

The work presented in [29] describes a representation to convert model structures using model-to-model transformations. Although Section 3 of [29] presents a similar process to tackle the problem of generating source code using model-to-model transformations instead of model-to-text transformation; the metamodel defined to carry out this task, which consists in a single metaclass (see Section 3.4 of [29]), provides a high level of genericity at the expense of an overload of StringNode instance specification (e.g. the prefix and suffix definition is mandatory for all instances, including those that do not require any, such as plain text). The proposed solution is based on the definition of the metamodel presented in Section 3.1 of this article where XML document specific characteristics are abstracted to represent these documents employing less effort to generate text (e.g. no need to specify prefixes and suffixes for each tag or attribute).

Regarding Xtend language, this approach employs a Java dialect to describe transformations using an imperative programming paradigm instead of a declarative paradigm which is the same that is employed by most of model-to-model transformation languages. Thus, developers do not need to use different programming paradigm to generate XML documents.

In Section II we highlight the main characteristics of XML documents and the limitations of using traditional model-to-text transformation languages that adopt the archetype-based paradigm. Then, in Section III we describe the generation method for documents defined in markup formats. This section includes Section III-A to describe the TagML meta-model which captures the essential characteristics of documents defined in markup formats, and Section III-B that describes the definition of the model-to-text transformation that generates the textual representation of these document defined in markup formats. A case study is presented in Section IV to illustrate how the defined method is applied in an scenario where input models are defined by a subset of the entities defined in the Essential Meta-object Facility (EMOF) [30]. The discussion about pro and cons of using this approach to generate documents in markup format is exposed in Section V. Finally, in Section VI we present the conclusions and future works.

IEEE SOFTWARE, VOL. 14, NO. 8, AUGUST 2015

II. Motivation

This section describes the main characteristics of XML documents and the limitations of using traditional model-to-text transformation languages using archetype-based paradigms. From the syntax perspective, these are the most relevant ones:

- The same tag keyword is used to define the beginning and the ending of XML document blocks. For instance, in XHTML, paragraphs are defined between the <P> and </P> tags where P represents the tag keyword.
- The same delimiters enclose keywords representing the beginning and the ending of XML document blocks. For instance, in XML, while the beginning of document blocks are defined by tag keywords enclosed by the < and > delimiters; the ending of document blocks are defined by tag keywords enclosed by the </ and > delimiters.
- The shortened text representation for empty document blocks do not require end tags. For instance; the BR document block in XHTML is enclosed by the < and /> delimiters
- The same delimiter is used to separate attributes in lists. For instance, the blank space is used to separate the rel="stylesheet" and href="./css/bootstrap.min.css" attributes in the XHTML rel="stylesheet" href="./css/bootstrap.min.css" attribute list.
- Attribute keys and values are separated by the same delimiters. For instance, in XHTML, the = character is used to separate the rel attribute key from the "stylesheet" attribute value.
- Attribute values are enclosed by the same delimiter. For instance, in XHTML, the " character is used to enclose the stylesheet attribute value.

Most transformation languages that define model-to-text transformations are based on archetypes. Some examples of this type of languages are JET Templates [20], MOFScript [19] and ACCELEO [18].

These languages are very versatile because they enable developers to generate code for any programming language. For instance, the listing 1 depicts the except of code that defines a model-to-text transformation rule in ACCELEO to generate XHTML documents.

Fig. 1 shows how the characteristics of XML documents are still present in the definition of model-to-text transformations using archetype-based languages that generate XML documents. The impact of these characteristics leads to information redundancy when defining model-to-text transformations.

The transformation rule depicted in Fig. 1 requires developers to write tag keywords twice to define document blocks; for instance, to define the BODY block, developers have to explicitly write the <BODY> and </BODY> tags.

It also requires developers to write terminals for all opening, closing and empty tag keywords; for instance, the <, /> have to be explicitly introduced when defining tags such as <BODY>, </HTML> and <LINK/>.

Attribute list, attribute key-value and attribute value delimiters are required too. For instance, developers introduce spaces when defining the rel and href attributes of the <LINK rel="stylesheet" href="./css/bootstrap.min.css"/> document block. They also separate attribute keys from values using the = character (i.e. href="./css/bootstrap.min.css"). And they enclose attribute values using quotes (i.e. "./css/-bootstrap.min.css")

```
[comment encoding = UTF-8 /]
[module generate('http://www.eclipse.org/uml2
    /5.0.0/UML', 'http://www.eclipse.org/uml2
    /5.0.0/Types', 'http://www.eclipse.org/emf
    /2002/Ecore')]

[template public generateDocument(aClass : Class)]
[file (aClass.name+'.html', false, 'UTF-8')]
<HTML lang="en">
  <HEAD>[aClass.name/]</HEAD>
  <LINK rel="stylesheet" href="./css/bootstrap.min
      .css"/>
  <LINK rel="stylesheet" href="./css/bootstrap-
      theme.min.css"/>
  <BODY>
    <H1>[aClass.name/]</H1>
    <P>Class [aClass.name/]</P>
    <SCRIPT src="./js/jquery-3.1.1.min.js"></
        SCRIPT>
    <SCRIPT src="./js/bootstrap.min.js"></SCRIPT>
  </BODY>
</HTML>
[/file]
[/template]
```

Fig. 1. The excerpt of the ACCELEO code that generates XHTML documents from UML class diagram models.

Another interesting issue to highlight about the definition of model-to-text transformations that generate XML documents using archetype-based languages is the fact that the generation of the textual representation of the XML document is interweaved with OCL expressions that calculates the information to be generated.

Under this scenario, the reuse of OCL expressions is poor because small changes in the syntax have a considerable impact in the definition of the transformation. For instance, suppose that the tag keyword delimiters changes from < and > to [and]. All transformations that have been defined using the < and > should be modified, or at least should be copied and modified if both versions of the transformation should be kept. This alternative present the same problems addressed by Model-driven architectures. For instance, the problem of divergent versions of the transformations when modifying model calculus [23]

As the result of the analysis of the peculiarities of the definition of model-to-text transformations using archetype-based languages, this proposal captures redundant information in XML document models following the Model-driven Architecture philosophy that leverages the abstraction and reuse principles to encourage the interoperability at design-time.

While the abstraction process captures XML document characteristics in the definition of a meta-model which is used to create XML document models; the reuse process is implemented by a model-to-text transformation that turns XML document models into text which generates redundant text.

Therefore, this approach enables developers to define model-to-model transformations instead of model-to-text transformations to generate XML text documents. It leads a

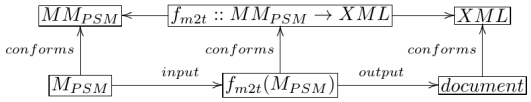
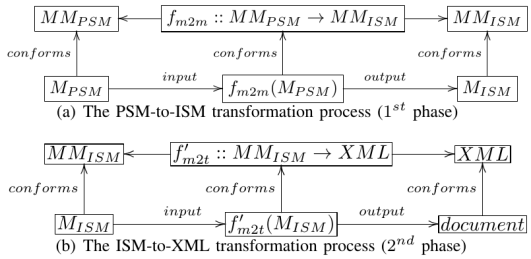


Fig. 2. Traditional model-to-text transformation process.



(a) The PSM-to-ISM transformation process (1^{st} phase)

(b) The ISM-to-XML transformation process (2^{nd} phase)

Fig. 3. The PSM-to-ISM transformation process (a) and the ISM-to-XML transformation process (b).

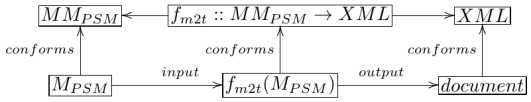


Fig. 4. The proposed model-to-text transformation process.

clear separation between calculus and syntax where model calculus are captured by the model-to-model transformation, and syntax modifications are captured by the model-to-text transformation. For instance, to change < and > tag keyword delimiters by [and], developers modify, or copy and modify, the model-to-text transformation avoiding any change on the calculus. Conversely, to change model calculus, developers modify, or copy and modify, the model-to-model transformation, avoiding any change on the text generation.

Consequently, this approach reduces the amount of information required to generate XML text documents which results on an increment of transformation definition reliability and a reduction of transformation maintenance costs.

III. The Method to Generate Documents

Fig. 2 depicts the traditional process to generate XML documents using model-to-text transformations in archetype-based languages.

The traditional process defines the $f_{m2t}(M_{PSM})$ model-to-text transformation function where M_{PSM} represents the Platform Specific Model (PSM) that should be turned into the text representation in XML format represented by $document$. Thus, f_{m2t} conforms to the $MM_{PSM} \rightarrow XML$ expression where MM_{PSM} represents the PSM meta-model and XML represents the XML format.

Conversely to the traditional approach, our proposal defines the process depicted in Fig. 4.

This proposal takes advantage of the XML document characteristics described in Section II to raise the level of abstraction while maximizing the reuse in the definition of model-to-text transformations that generate XML documents. Following the MDA philosophy, this proposal defines the Implementation Specific Model (ISM) abstraction layer described in section III-A between the Platform Specific Model(PSM) and the textual representation of XML documents to decouple the text generation process from the model calculus.

This process is divided into two phases. The first phase of the transformation process is depicted in Fig. 3(a). In this phase, transformation developers define the $f_{m2m}(M_{PSM})$ model-to-model transformation function that turns the M_{PSM} PSM conforming the MM_{PSM} meta-model into the M_{ISM} ISM conforming the MM_{ISM} meta-model which captures the XML document characteristics described in Section II.

The second phase of the transformation process is depicted in Fig. 3(b). It defines the $f'_{m2t}(M_{ISM})$ model-to-text transformation function, which is defined in section III-B. This function turns the M_{ISM} ISM conforming the MM_{ISM} meta-model generated in the previous phase into the textual representation of the XML $document$ the M_{ISM} ISM represents.

Employing this approach, transformation developers focus on the development of the model-to-model transformation that defines model calculus leaving the syntax issues to the model-to-text transformation.

A. The Implementation Specific Meta-model

The Implementation Specific Model (ISM) is defined in TagML. TagML is a Domain Specific Language (DSL) which was designed to capture XML-based documents characteristics to leverage the level of abstraction and reuse when defining model-to-text transformations that generate XML documents. It is defined in terms of the meta-model depicted in Fig. 5.

TagML was developed as an Eclipse platform feature using the Eclipse Modeling Framework (EMF) [31]. Consequently, this feature consists on a set of plug-ins that follow the Object Management Group (OMG) standards which ensure the interoperability of this tool with third party software. This way of developing tools presets some advantages with respect to those developed ad-hoc.

The capability of integrating this plug-in with third part plug-ins (e.g. UML plug-ins, transformation tools, etc.), the capability of Eclipse plug-ins to be executed in different platforms (e.g. Windows, Linux, Mac, etc.), the functionality that is inherited from the Eclipse platform (e.g. workspace, project and file management, user configuration, customized text editors, plug-in configuration management, task management, model repositories, etc.)

Both the TagML model editor and the M2T transformation have been developed as Eclipse plugins. These plugins were developed with the Eclipse Modeling Framework (EMF) technology. The use of EMF ensures that these tools follow the standards of the Object Management Group (OMG) ensuring their interoperability with third-party tools.

Developing tools such as Eclipse plugins has certain advantages with respect to the development of ad-hoc environments. The first one is to have a multi-platform development environment. In addition, we can mention the inherited functionality of the environment, which, for example, provides management of workspaces, projects, files, configuration for

IEEE SOFTWARE, VOL. 14, NO. 8, AUGUST 2015

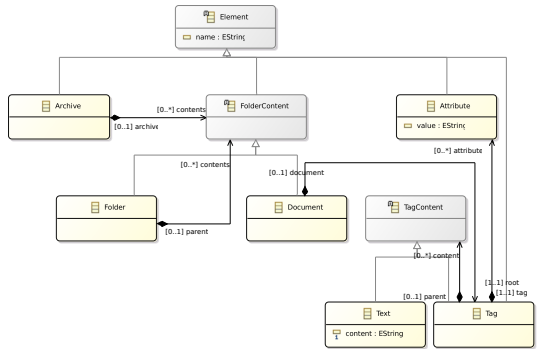


Fig. 5. The meta-model of TagML.

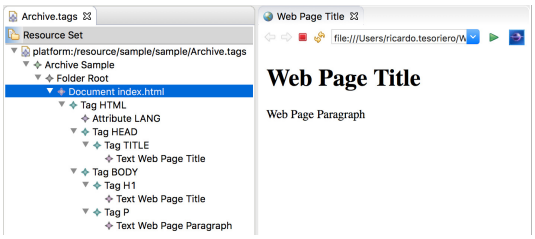


Fig. 6. Example of the TagML model of a Web document.

the user, custom text editors, integration with third-party tools, management of plugins configuration, management of tasks, model management, etc.

The goal of the TagML meta-model is to capture the characteristics of tag-based languages to raise the level of reuse of M2T transformations. Figure 5 shows the meta-model TagML.

As you can see in the Fig. 5, this meta-model defines Archive as the root of a TagML model by defining a collection of FolderContent instances. The FolderContent, Folder and Document meta-classes define an instance of the Composite design pattern [32]. The role of FolderContent is that of *Component*, that of Folder is that of *Composite* and that of Document is that of *Leaf*. This design pattern is ideal for representing the structure of files that contain the XML documents that will be generated.

Each Document instance defines a Tag instance as the *root* label of the document. This instance is part of another instance of the design pattern *Composite* where the TagContent fulfills the role *Composite*, Text the role *Leaf* and Tag fulfills the role *Composite*. Again, this design pattern is best suited to represent the tree structure that all label-based documents define.

Text instances represent plain text that can be part of the content of a label. Tag instances represent a tag that defines a collection of attributes represented by Attribute instances.

All meta-classes, except TagContent, define the attribute *name*, inherited from the Element meta-class abstract.

Figure 6 shows the example of a TagML instance model of a Web document in XHTML and its representation in a Web browser.

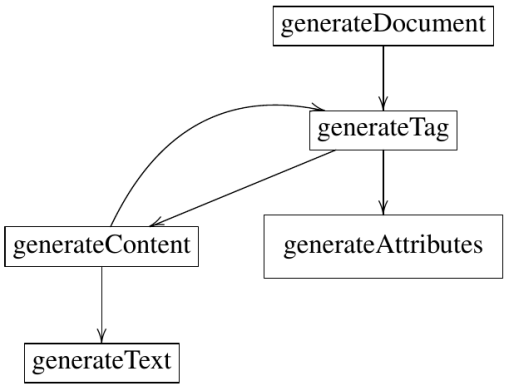


Fig. 7. Structure of XML document generation in ACCELEO.

```
[query public getPath(aDocument : Document) :
    String = if (self.parent.oclIsUndefined())
    then '/'+self.archive.name else self.parent.
    getPath() endif /]
[query public getPath(aFolder : Folder) : String =
     if (aFolder.parent.oclIsUndefined()) then '/'
    +self.archive.name else self.parent.getPath()
    endif +'/'+self.name  /]
[template public generateDocument(aDocument :
    Document)]
[comment @main/]
[file (aDocument.getPath()+'/'+aDocument.name,
    false, 'UTF-8')]
[self.root.generateTag()/][/file][/template]
```

Fig. 8. XML document generation template in ACCELEO.

B. Model-to-Text Transformation

The M2T transformation of a PSM is carried out in 2 phases:

1) The execution of the M2M transformation of the PSM to the TagML model
2) The execution of the M2T transformation of the TagML model to the tag language

In this section, we will focus on the M2T transformation that uses a model in TagML as an input model and, as an output, generates a document based on labels.

As mentioned above, TagML models are independent of the domain of the application. They only describe the content of the document to be generated.

The M2T transformation has been developed in ACCELEO. ACCELEO is an M2T transformation language based on archetypes.

The transformation module defines 7 templates (template) and 2 queries (query). The structure of document generation can be seen in the Fig. 7.

Figure 8 shows the code in ACCELEO of the transformation of an XML document described in a TagML model. This code template generates the file that will contain the XML document. The calculation of the file name is calculated with the query getPath (). In addition, it starts the code generation of the tags, starting with the root tag.

Figure 9 shows the code related to the generation of labels. This code is responsible for generating the code associated

```
[template public generateTag(aTag : Tag) post (
    trim())]
[if (aTag.content->isEmpty())]<[aTag.name/][aTag.
    generateAttributes()/]/>
[else]<[aTag.name/][aTag.generateAttributes()/]>
[for (tagContent : TagContent | aTag.content)]
  [tagContent.generateContent()/][/for][/if]
[if (not aTag.content->isEmpty())]</[aTag.name
    /]>[/if][/template]
```

Fig. 9. Template for generating XML tags in ACCELEO.

```
[template public generateAttributes(aTag : Tag)]
[for (attr : Attribute | aTag.attribute)][attr.
    generateAttribute()/][/for][/template]
[template public generateAttribute(attr :
    Attribute)]
 [if (not attr.value.oclIsUndefined())][self.name
     /]='[attr.value/]'[else][self.name/][/if]
 [/template]
```

Fig. 10. Template for generating attributes of an XML tag in ACCELEO.

```
[template public generateContent(aTagContent :
    TagContent) ? (aTagContent.oclIsTypeOf(Tag))
    post (trim())]
[aTagContent.oclAsType(Tag).generateTag()/][/
    template]
[template public generateContent(aTagContent :
    TagContent) ? (aTagContent.oclIsTypeOf(Text))
    post (trim())]
[aTagContent.oclAsType(Text).generateText()/][/
    template]
```

Fig. 11. Template for generating the content of an XML tag in ACCELEO.

```
[template public generateText(aText : Text) post (
    trim())][self.content/][/template]
```

Fig. 12. Plain text generation template in ACCELEO.

with a label, the attributes associated with it and the recursive call to the nested labels.

To generate the code of the attributes, the code shown in Fig. 10.

The recursive call to generate the nested tags is carried out by calling the polymorphic function generateContent in FolderContent. This function which distributes its execution according to the type of element that is treated. If it is a label, then the function generateTag shown in Fig. 9 is executed. Otherwise, the function generateText that can be seen in Fig. 12 is executed. This function generates the content associated with a plain text.

C. Implementation issues

Both the TagML model editor and the M2T transformation have been developed as Eclipse plugins. These plugins were developed with the Eclipse Modeling Framework (EMF) technology. The use of EMF ensures that these tools follow the standards of the Object Management Group (OMG) ensuring their interoperability with third-party tools.

Developing tools such as Eclipse plugins has certain advantages with respect to the development of ad-hoc environments. The first one is to have a multi-platform development environment. In addition, we can mention the inherited functionality of the environment, which, for example, provides management of workspaces, projects, files, configuration for the user, custom text editors, integration with third-party tools, management of plugins configuration, management of tasks, model management, etc.

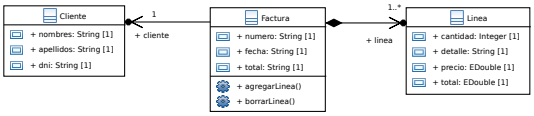


Fig. 13. UML class diagram.

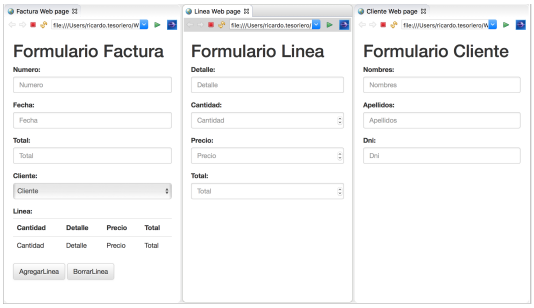


Fig. 14. XHTML generated documents.

IV. Case study

Once defined the meta-model that allows to model XML documents, and the transformation that allows to generate the code associated with these models, we proceed to show a case study to demonstrate the versatility of the proposed generation method.

The case study described in this section, is a generator of Web documents in XHTML from a UML class model [33]. Figure 13 presents an example of an input model that defines 3 classes. This model has been generated with the Papyrus tool [34]. Figure 14 shows the result of applying the transformation method proposed in the section III.

As seen in the section III, the process defines two phases. While the first phase requires the definition of an M2M transformation of the PSM to a TagML model, the second one does not require any intervention from the developer being fully automatic and independent of the PSM. Therefore, we will focus on the definition of the first transformation.

The first step in defining the first transformation is to identify the meta-input and output models. In this case they are UML and TagML.

To define this transformation we used ATL [22] which is a declarative language to define model-to-model transformations. However, you can use any other, eg. QVT [21]. The reason for this is that the model-to-text transformation is carried out from the TagML model.

The code shown in Fig. 15 converts the instance of Model into an Archive instance and the Package instance into Folder instances.

Figure 16 shows the most relevant lines of code of how classes are converted to documents.

IEEE SOFTWARE, VOL. 14, NO. 8, AUGUST 2015

```
rule Model2Archive {
from model: UML!Model
to
folder: TagML!Folder (name <- model.name.
    firstToLower(),
    archive <- archive),
archive: TagML!Archive (name <- model.name.
    firstToLower())}

rule Package2Folder {
from model: UML!Package (model.oclIsTypeOf(UML!
    Package))
to
folder: TagML!Folder (name <- model.name.
    firstToLower(),
    parent <- model.namespace)}
```

Fig. 15. Transformation rules from Model to Archive and from Package to Folder in ATL.

```
rule Class2Document {
from model: UML!Class
to
form: TagML!Tag (name <- 'FORM', parent <-
    divContainerCol),
document: TagML!Document (
    name <- model.name.firstToLower() + '.html',
    parent <- model.namespace),
html: TagML!Tag (name <- 'HTML',document <-
    document),
head: TagML!Tag (name <- 'HEAD',parent <- html),
body: TagML!Tag (name <- 'BODY',    parent <- html
    ),
p: TagML!Tag (name <- 'H1',parent <- form),
pText: TagML!Text (
    content <- 'Formulario '.concat(model.name.
        firstToUpper()),
    parent <- p)
}
```

Fig. 16. Rule of transformation of Class a Document in ATL.

```
abstract rule Attribute2Field {
from
property: UML!Property (property.association.
    oclIsUndefined())
using { inputIdString: String = property.name.
    concat('_ID');    }
to
label: TagML!Tag (name <- 'LABEL',parent <-
    formGroup   ),
labelText: TagML!Text (
    content <- property.name.firstToUpper().concat
        (': '),
    parent <- label),
labelFor: TagML!Attribute (name <- 'for',
    value <- inputIdString, tag <- label),
input: TagML!Tag (name <- 'INPUT', parent <-
    formGroup  ),
}
```

Fig. 17. Property to Tag transformation rule in ATL.

The code in Fig. 17 shows the rule that transforms the properties of a class into the fields of the form associated with the class.

The model resulting from executing the transformations can be seen in Fig. 18. The result of the M2M transformation can be seen in the left half of Fig. 18. In it, you can see that the root node is an instance of Archive. This node contains the model folder, which is an instance of Folder. Inside the folder model you can see three documents (Document instances): invoice.html, linea.html and client.html. The content of the

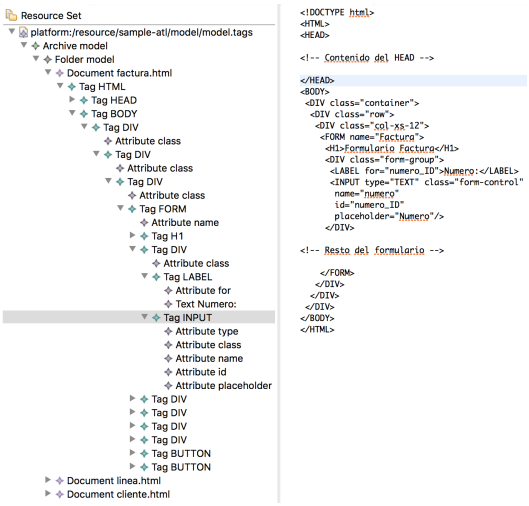


Fig. 18. Result of the M2M and M2T transformations.

document factura.html defines a set of labels that make up its structure. In this case, we can see the labels that present the branch of the tree that contain the form that corresponds to the class Invoice. An extract of the result of the M2T transformation is shown in the left half of Fig. 18. In this extract you can see the correspondence of the Tag instances and the generated XML tags.

For instance, an example of the benefit of employing this approach from the maintainability perspective is the simplicity of changing tag delimiters (i.e. "<", ">", "/>" by "[", "]" and "/]") by duplicating the proposed transformation and performing a small modification of 4 characters in lines 2, 3 and 6 of the code presented in Figure 8. This fact reduces maintainability costs because all models represented with the proposed metamodel can be reused to generate the same documents suing a different format without changing XML document models at all.

V. Evaluation of the Proposed Method

This section discusses the evaluation of the proposed method to tackle the generation of XML documents using M2M transformations. This evaluation compares the use of the traditional mechanism (archetype based M2T transformation languages), against declarative M2M transformation languages for the definition of model transformations that generate XML documents.

The evaluation is performed in two steps: first a quantitative evaluation is described, then a qualitative evaluation is presented, both to compare the two techniques.

A. Quantitative evaluation

The quantitative evaluation focuses on the amount of code required to generate XML documents that is not part of the transformation language. This type of code is referred in this section as *unhandled* code. The unhandled code is not validated in any way; therefore, we assume that the more code

is unhandled by the transformation language, the more error prone it is.

Before defining the metric to measure the unhandled code, we define the set of well-formed XML documents using the equations 1, 2, 3, 4, and 5 where:

- I represents the set of valid XML Strings to define XML identifiers (e.g. do not contain any spaces)
- S represents the set of valid XML Strings to define XML text (e.g. representation of accented characters in ASCII)
- A represents the set of valid XML attributes where n and v represent attribute names and values respectively
- T represents the set of valid XML tags where A' represents the set of attributes of the tag t and T' represents the set of tags that are nested in the tag named t
- D represents the set of valid XML documents where d represents an XML document name and T'' represents the set of elements defined at the top level of the XML document named d

$$I = \{\text{valid XML Strings to represent identifiers}\} \quad (1)$$

$$S = \{\text{valid XML Strings to represent text}\} \quad (2)$$

$$A = \{(n, v); n \in I \wedge v \in S\} \quad (3)$$

$$T = \{(t, A', T'); t \in I \wedge \forall a' \in A', a' \in A \wedge \forall t' \in T', t' \in T\} \quad (4)$$

$$D = \{(d, T''); d \in I \wedge \forall t'' \in T'', t'' \in T\} \quad (5)$$

From Equations 1, 2, 3, 4, and 5, we derived the 3 transformation cases covering the XML document structure:

1) Attribute generation
2) Empty tag generation
3) Composed tag generation

1) Case 1: Attribute generation: The Equation 3 defines the set of valid attributes in XML. The code required to generate attributes in an archetype-based M2T transformation language (i.e. Acceleo) is `␣[n/]="[v/]"`. And the amount of unhandled code to generate these tags is 4 (i.e. `␣=""`).

If $f_A(d)$ represents the set of valid XML attributes for the XML document d, $C_A(d) = 4 \times \#f_A(d)$ is the amount of unhandled code.

Alternatively, the code required to generate attributes in a declarative M2M transformation language (i.e. ATL) is `attr : TagML!Attribute (name <- n, value <-v)`. In this case, all the code is part of the transformation language; therefore, the amount of unhandled code is 0.

2) Case 2: Empty tag generation: According 4, $T_{C2} = \{(t, \emptyset, \emptyset); t \in I\}$ defines the set of empty tags without attributes.

The code required to generate empty tags without attributes in an archetype-based M2T transformation language (i.e. Acceleo) is `<[t/]/>`. And the amount of unhandled code to generate these tags is 3 (i.e. `</>`).

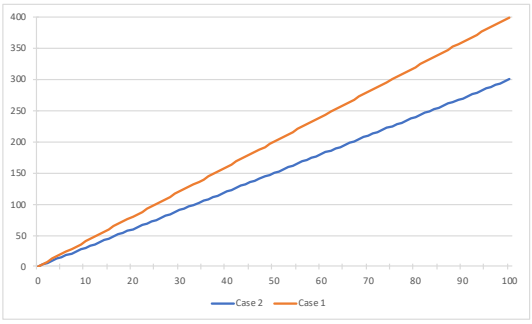


Fig. 19. The evolution of the amount of unhandled code in Case 1 and Case 2.

If $f_{ET}(d)$ represents the set of empty tags without attributes of an XML document d, $C_{ET}(d) = 3 \times \#f_{ET}(d)$ is the amount of unhandled code.

On the other hand, the code required to generate empty tags without attributes in a declarative M2M transformation language (i.e. ATL) is `tag : TagML!Tag (name <- t)`. Again, all the code is part of the transformation language; therefore, the amount of unhandled code is 0.

3) Case 3: Composed tag generation: According 4, $T_{C3} = \{(t, \emptyset, T'); t \in I \wedge \forall t' \in T'(t' \in T) \wedge T' \neq \emptyset\}$ defines the set of composed tags without attributes. The code required to generate empty tags without attributes in an archetype-based M2T transformation language (i.e. Acceleo) is `<[t]>[for t' in T'] [/for]</[t/]>`. And the amount of unhandled code to generate these tags is 5 (i.e. `<></>`).

If $f_{CT}(d)$ represents the set of composed tags without attributes of an XML document d, $C_{CT}(d) = 5 \times \#f_{CT}(d)$ is the amount of unhandled code.

However, if the tag name (i.e. t) is not generated in a local variable, and it is a constant value, t should be written twice. Therefore, $C'_{CT}(d) = 5 \times (\#f_{CT}(d) + l(t))$ where $t \in f_{CT}(d)$ and $l(x)$ represents the length of x.

On the other hand, the code required to generate empty tags without attributes in a declarative M2M transformation language (i.e. ATL) is `tag : TagML!Tag (name <- t)`. Again, all the code is part of the transformation language; therefore, the amount of unhandled code is 0.

Instead, the code required to generate composed tags without attributes in a declarative M2M transformation language (i.e. ATL) is `tag : TagML!Tag (name <- t, contents <- T'.collect(t'))`|. Again, all the code is part of the transformation language; therefore, the amount of unhandled code tis 0.

4) Analysis: The Fig. 19 shows the lineal evolution of the amount of unhandled code using a M2T transformation language in terms of the amount of attributes and empty tags.

The Fig. 20 shows the evolution of the amount of unhandled code using a M2T transformation language in terms of the amount of composed tags (x axis) and the length of tag names (y axis).

IEEE SOFTWARE, VOL. 14, NO. 8, AUGUST 2015

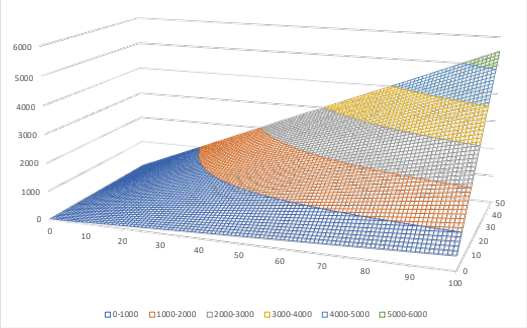


Fig. 20. The evolution of the amount of unhandled code in Case 3.

B. Qualitative evaluation

This section describes the validation of the proposed method based on the five general dimensions of Model Driven Engineering (MDE) [35] that a good model should cover.

In [35] it is explained that a good model has to cover the following five characteristics: 1) abstraction, 2) understandability, 3) accuracy, 4) predictiveness, and 5) inexpensiveness.

Abstraction. A model of a system should reduce the details of the system it represents.

Understandability. A model should be easier to understand than the system it models.

Accuracy. While abstraction may summarize or hide important details, the meaning of these details should not be altered by a model.

Predictiveness. A model should be executable in order to help developers predict specific system behaviour.

Inexpensiveness. A model must be inexpensive to produce.

1) Abstraction: The definition of TagML models hides XML document syntax from developers while keeping model XML document structure. To carry out this task, this proposal defines an Implementation Specific Models (ISMs) to generate XML documents.

The definition of ISMs facilitate the definition of vertical transformations (to perform model abstraction and refinement), horizontal transformations (to perform model migrations and representations), and mixed transformations. For instance, an ISM enables developers to define abstraction transformations using ATL; instead of using a traditional parser that reads text and creates models manually. Moreover, the use of a declarative language, instead of an archetype-based language, enables developers to execute inverse transformations.

The use of ISMs also maximizes the interoperability at design time delaying the generation process to later stages of the development enabling developers to introduce changes in the TagML model that do not affect the rest of the models. For instance, developers can modify or add tags to XML documents in the TagML model avoiding changes in the transformation function definition improving its maintainability.

2) Understandability: This proposal decouples XML document syntax from its structure defining TagML models. The use of TagML models improves transformation model understandability because it enables developers to focus on the XML document structure; instead of the XML syntax (terminals, closing tags, etc.) that is a key issue when using M2T archetype-based transformation approaches.

3) Accuracy: As we mentioned in Section V-B1, the definition of TagML models hides XML document syntax from developers while keeping model XML document structure. The proofs of accuracy are explained in Section III-B and depicted in Fig. 8, Fig. 9, Fig. 10, Fig. 11, and Fig. 12 where we present most relevant excerpts of code that define M2T transformation functions that transform TagML metamodel entities depicted in Fig. 5 into XML document source code.

4) Predictiveness: The definition of TagML models improves the prediction of specific system behaviour introducing XML document validation at metamodel level;for instance, to check that XML tag names are valid (e.g. do not contain any spaces). Thus, the XML document validation is defined once for all TagML models, improving model reliability, reuse and reducing maintenance costs; for instance, new invariants can be easily introduced without modifying the transformation function definition. It improves model predictiveness because model validation ensures that generated XML documents are syntactically valid.

Conversely, the use of a traditional approaches using M2T archetype-based transformation languages implies the definition of a validation function (e.g. query in Acceleo) that must be called explicitly in all transformations whenever XML elements are generated. This practice interweaves the validation code with the text generation, which MDA practitioners should avoid for the sake of the separation of concerns (i.e. XML document generation and validation).

5) Inexpensiveness: The definition of the TagML metamodel also enables developers to use the same transformation language to manipulate models and generate source code; instead of the two different languages that are employed in traditional approaches (archetype-based and declarative). It makes the generation process less expensive compared to traditional approaches where developers require skills in two transformation languages following different programming paradigms.

VI. Conclusions and Future Work

The first conclusion we can observe is that the proposed M2T transformation definition method facilitates a more readable, compact definition, with lower maintenance costs and less prone to errors than traditional methods.

In addition, it provides a complete development environment based on Eclipse plugins that allows easy integration with third-party tools. Also, it provides a set of services inherent to the platform that are available in the tool. Regarding the standards, when developing a plugin with EMF technology, all the devices comply with the OMG standards. Last but not least, this environment provides a system for updating and managing versions of the plugins that allow easy maintenance and extension of the tool.

As future work, we are working on extending the environment to customize and optimize M2T transformations that have a TagML model as input. We are also working on a

new version of TagML to support meta-labels and markup languages that allow customizing transformations.

ACKNOWLEDGMENT

This work was supported partially by the Junta de Comunidades de Castilla-La Mancha under Grant SBPLY/17/180501/000495 and the Spanish Ministerio de Ciencia, Innovacion y Universidades under grant RTI2018-099942-B-I00.

REFERENCES

[1] Object Management Group, "XML Metadata Interchange," Object Management Group, June 2015. [Online]. Available: http://www.omg.org/spec/XMI/2.5.1

[2] World Wide Web Consortium, "XHTML 1.0 The Extensible HyperText Markup Language (Second Edition)," World Wide Web Consortium, August 2002. [Online]. Available: https://www.w3.org/TR/xhtml1/

[3] ——, "HTML 5.1," World Wide Web Consortium, November 2006. [Online]. Available: https://www.w3.org/TR/html51/

[4] J. C. Process, "JSR-245 : JavaServer Pages (JSF 2.1) Specification," Java Community Process, May 2013. [Online]. Available: https://jcp.org/aboutJava/communityprocess/final/jsr245/index.html

[5] ——, "JSR-372 : JavaServer Faces (JSF 2.3) Specification. final draft," Java Community Process, March 2017. [Online]. Available: https://jcp.org/aboutJava/communityprocess/pfd/jsr372/index.html

[6] ——, "JSR-52 : A Standard Tag Library for JavaServer Pages," Java Community Process, November 2006. [Online]. Available: https://jcp.org/aboutJava/communityprocess/maintenance/jsr052/index3.html

[7] Microsoft, "ASP .NET web site," Microsoft, 2017. [Online]. Available: https://www.asp.net/

[8] ——, "XAML documentation," Microsoft, 2012. [Online]. Available: https://msdn.microsoft.com/en-us/library/gg134030.aspx

[9] Google, "Polymer project," Google, 2017. [Online]. Available: https://www.polymer-project.org/1.0/

[10] ——, "Angularjs," Google, 2017. [Online]. Available: https://angularjs.org/

[11] ISO, "X3D V3.3 Abstract Specification," International Organization for Standardization, Geneva, Switzerland, ISO ISO/IEC IS 19775-1:2013, 2013.

[12] Android, "Android Web Site," February 2017. [Online]. Available: https://www.android.com/

[13] World Wide Web Consortium, "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)," World Wide Web Consortium, April 2007. [Online]. Available: https://www.w3.org/TR/soap12-part1/

[14] Object Management Group, "OMG web site," Object Management Group, 2017. [Online]. Available: http://www.omg.org/

[15] ——, "MDA - The Architecture Of Choice For A Changing World," Object Management Group, 2017. [Online]. Available: http://www.omg.org/mda/

[16] World Wide Web Consortium, "W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures," World Wide Web Consortium, April 2012. [Online]. Available: https://www.w3.org/TR/xmlschema11-1/

[17] ——, "XSL Transformations (XSLT) Version 1.0," World Wide Web Consortium, November 1999. [Online]. Available: https://www.w3.org/TR/xslt

[18] Eclipse Foundation and Obeo, "Acceleo," Obeo, February 2017. [Online]. Available: http://www.eclipse.org/acceleo/

[19] Eclipse Foundation, "MOFScript Home Page," Eclipse Foundation, February 2011. [Online]. Available: https://eclipse.org/gmt/mofscript/

[20] ——, "Java Emitter Templates (JET2)," Eclipse Foundation, June 2011. [Online]. Available: https://projects.eclipse.org/projects/modeling.m2t.jet

[21] Object Management Group, "Meta Object Facility (MOF) 2.0 Query/View/Transformation (QVT)," Object Management Group, June 2016. [Online]. Available: http://www.omg.org/spec/QVT/1.3

[22] F. Jouault, F. Allilaire, J. Bézivin, and I. Kurtev, "Atl: A model transformation tool. science of computer programming," pp. 31–39, 2008.

[23] S. Mellor, *MDA Distilled, Principles of Model Driven Architecture*. Addison-Wesley Professional, 2004.

[24] D. S. Kolovos, L. M. Rose, J. Williams, N. Matragkas, and R. F. Paige, "A lightweight approach for managing xml documents with mde languages," in *Proceedings of the 8th European Conference on Modelling Foundations and Applications*, ser. ECMFA'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 118–132. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-31491-9_11

[25] D. S. Kolovos, R. F. Paige, and F. A. C. Polack, "The epsilon object language (eol)," in *Model Driven Architecture – Foundations and Applications*, A. Rensink and J. Warmer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 128–142.

[26] T. J. Parr, "Enforcing strict model-view separation in template engines," in *Proceedings of the 13th International Conference on World Wide Web*, ser. WWW '04. New York, NY, USA: ACM, 2004, pp. 224–233. [Online]. Available: http://doi.acm.org/10.1145/988672.988703

[27] A. Anjorin, K. Saller, S. Rose, and A. Schürr, "A framework for bidirectional model-to-platform transformations," in *Software Language Engineering*, K. Czarnecki and G. Hedin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 124–143.

[28] J. Bézivin, H. Bruneliere, F. Jouault, and I. Kurtev, "Model engineering support for tool interoperability," in *Conference: 4th Workshop in Software Model Engineering (WiSME 2005)*, 10 2005.

[29] E. D. Willink, "A text model - use your favourite m2m for m2t." in *18th International Workshop in OCL and Textual Modeling (OCL 2018)*, 2018.

[30] Object Management Group, "Meta Object Facility (MOF) 2.5.1," Object Management Group, November 2016. [Online]. Available: http://www.omg.org/spec/MOF/2.5.1/

[31] Eclipse Foundation, "EMFModelingFramework (EMF)," Eclipse Foundation, February 2017. [Online]. Available: https://www.eclipse.org/modeling/emf/

[32] Gamma, Helm, Johnson, and Vlissides, *Design Patterns Elements of Reusable Object-Oriented Software*. Massachusetts: Addison-Wesley, 2000.

[33] Object Management Group, "Unified Modeling Language (UML)," Object Management Group, June 2015. [Online]. Available: http://www.omg.org/spec/UML/2.5

[34] Eclipse Foundation, "Papyrus Modeling environment," Eclipse Foundation, February 2017. [Online]. Available: https://eclipse.org/papyrus/

[35] B. Selic, "The pragmatics of model-driven development," *IEEE Software*, vol. 20, no. 5, p. 19–25, 2003.

Ricardo Tesoriero received his PhD from the University of Castilla-La Mancha (UCLM), Spain. He performed a post-doc stay in the Université Catholoque de Louvain (UCL) in Louvain-La Neuve, Belgium where he conducted research activities in the Model-driven development of User Interfaces research field. His main research interests are focused on the model-driven development of user interfaces and context-aware applications in ubiquitous computing environments. He published more than 70 research articles in International Congresses and Journals. He participated in several scientific committees of international conferences and workshops, including DUI (Distributed User Interfaces), INTERACCION, ISEC, IADIS/WWW (La Web), and so on. He is professor of the Computing System Department of the UCLM and he teaches Human-Computer Interaction, Advanced Methods of Software Development and Web Engineering and Services subjects at the Faculty of Computer Science Engineering since 2008.

IEEE SOFTWARE, VOL. 14, NO. 8, AUGUST 2015

Gabriel Sebastián received his BSc degree from Polytechnic University of Valencia (UPV), and his MSc degree from University of Castilla-La Mancha (UCLM). 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 researcher in the Interactive Systems Everywhere research group in the Albacete Research Institute of Informatics (I3A) in Albacete, Spain.

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.

Capítulo 8

Automatic Code Generation for Language-Learning Applications

Citación: G. Sebastián, R. Tesoriero, J. A. Gallud, Automatic Code Generation for Language-Learning Applications, IEEE Latin America Transactions (2019).

Tipo de publicación: Revista Internacional (IF: 0.804, Q4)

Esta publicación avala esta disertación por compendio de publicaciones.

IEEE LATIN AMERICA TRANSACTIONS, VOL. 17 NO. 12, NOVEMBER 2019 1

Automatic Code Generation for Language-Learning Applications

Gabriel Sebastián, Ricardo Tesoriero, Jose A. 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.

Enviado a revisión: 14/11/2019.

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.

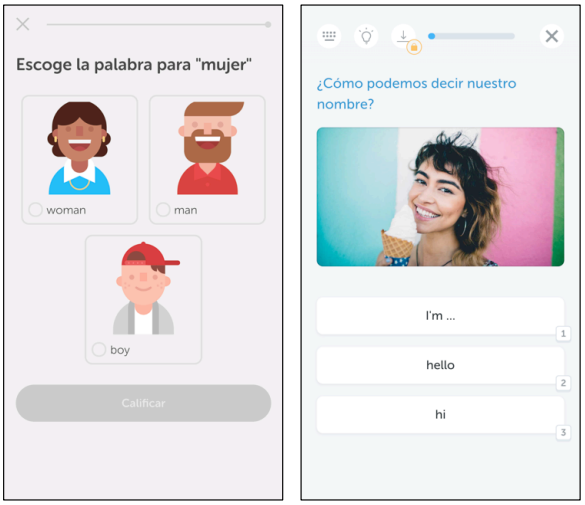
II. Planteamiento del Problema

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).

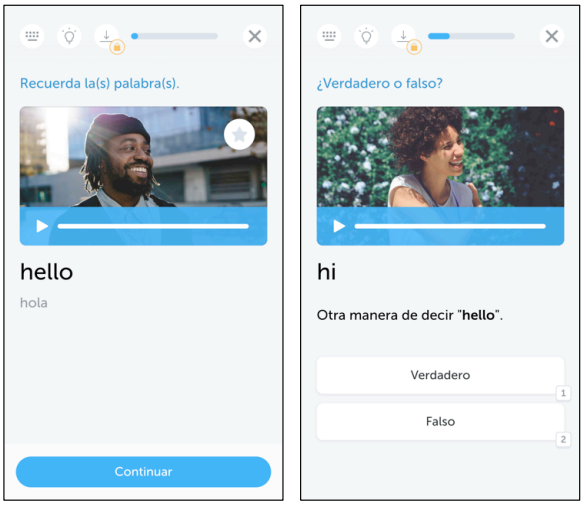
Los siguientes párrafos se centran en algunas características de la metodología de aprendizaje de *Busuu* como base para el caso de estudio. Esta descripción asume que el español es el idioma nativo del estudiante (SNL) y que el inglés es el idioma a aprender (LtL).

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)

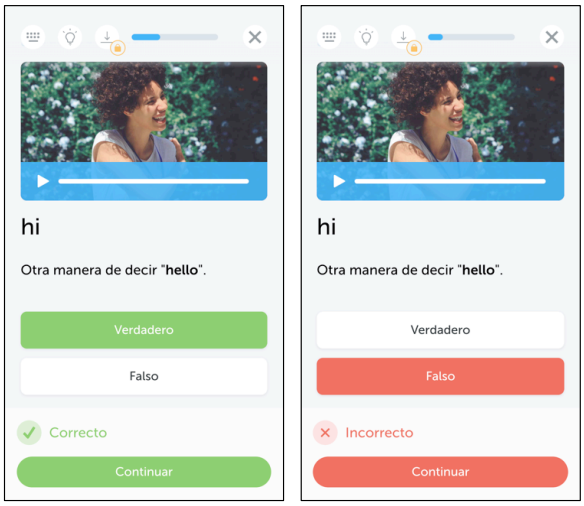
Fig. 2. Dos Mecanismos de Actividades de Aprendizajes en *Busuu*.

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*.

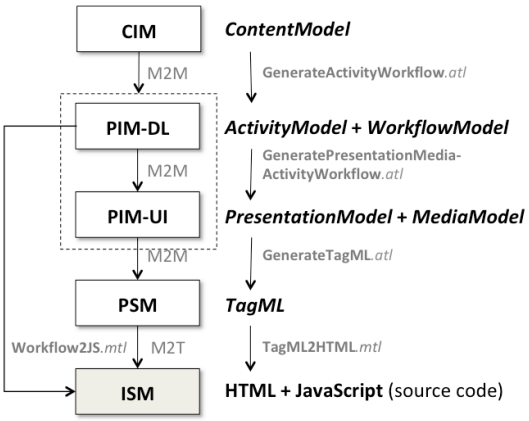


Fig. 4. Capas y transformaciones de la arquitectura basada en modelos de la propuesta.

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.

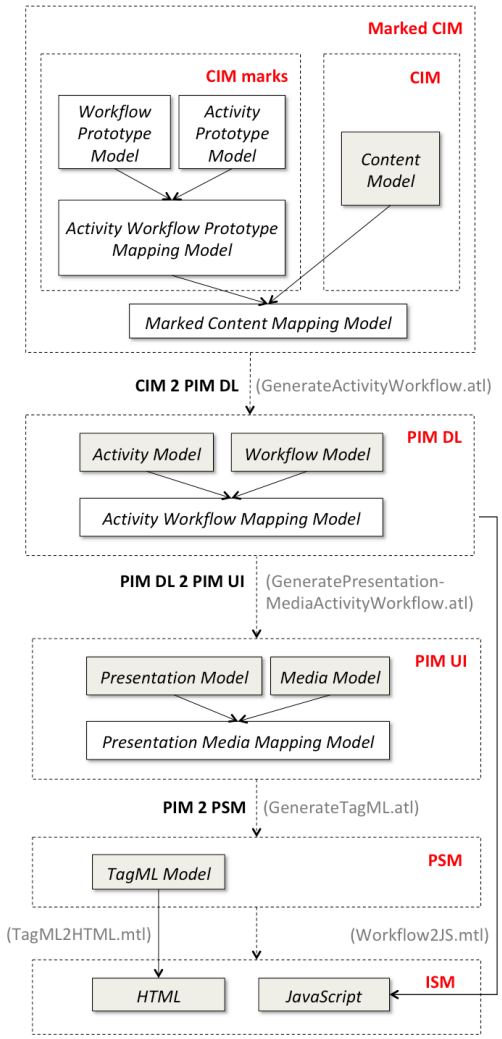


Fig. 5. Proceso de desarrollo basado en modelos de aplicaciones de aprendizaje de idiomas.

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

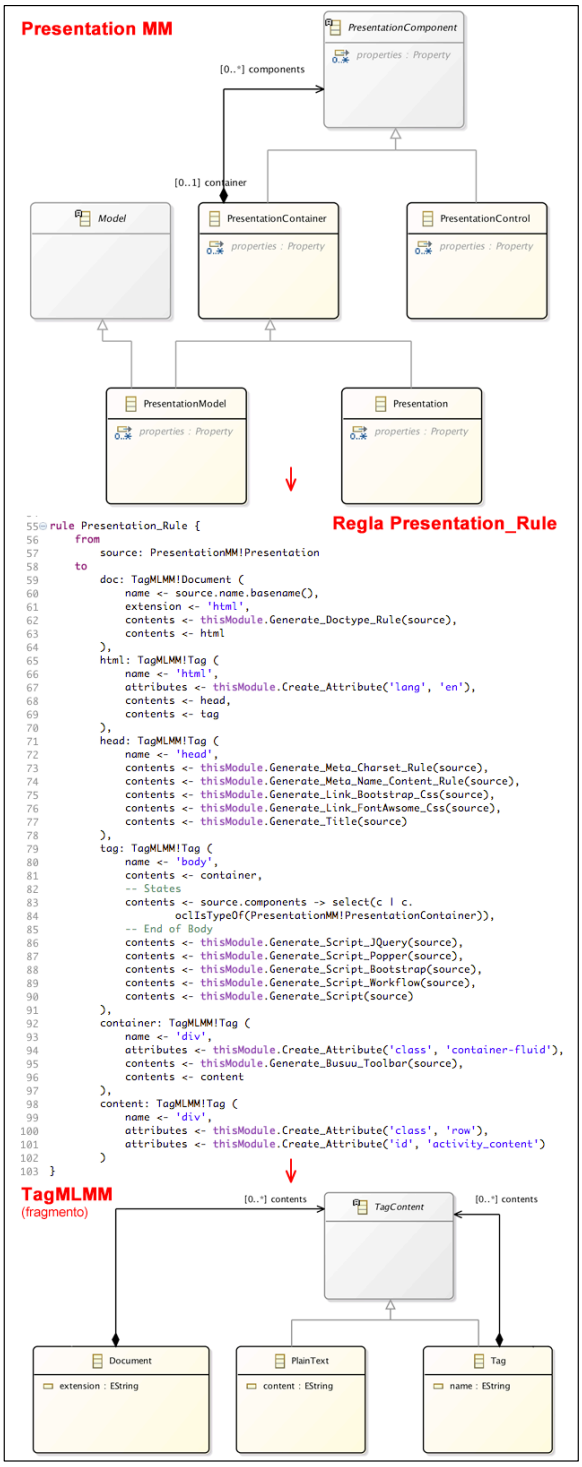


Fig. 6. Ilustración de parte de la generación del PSM en TagML (regla Presentation_Rule contenida en GenerateTagML.atl).

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.

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.

Capítulo 9

A Visual DSL for automatic generation of Language-learning Applications

Citación: G. Sebastián, R. Tesoriero, J. A. Gallud, A Visual Domain Specific Language for automatic generation of Language-learning Applications, PLOS ONE.

Tipo de publicación: Revista Internacional (IF: 2.441, Q2)

(Enviado)

A Visual Domain Specific Language for automatic generation of Language-learning Applications

Gabriel Sebastián[1*], Ricardo Tesoriero[2], Jose A. Gallud[2], Daniyal Alghazzawi[3]

1 Albacete Research Institute of Informatics, University of Castilla-La Mancha, Albacete, Spain
2 Faculty of Computer Science Engineering, University of Castilla-La Mancha, Albacete, Spain
3 Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia

* gabriel.sebastian@uclm.es

Abstract

This paper presents a graphical notation for a domain specific language to represent language learning activities. It describes how this notation enables developers to represent language learning activity characteristics using workflow, presentation, content, media and activity model conforming a metamodel that defines the abstract syntax of the domain specific language. This notation is implemented as part of an integrated development environment to build model-based applications. Finally, this proposal is evaluated with a framework that uses the cognitive dimensions of notations for notational systems. The proposed graphic diagram editor exceeds the experience that the user has with the reflexive model editor. In relation to the creation and editing of workflow models and presentation/activity models, the proposed graphical notation its more intuitive and easy to maintain visually than the traditional reflexive tree notation used by many model-based development frameworks.

1 Introduction

Globalization has increased the need for society to master new languages. This need has encouraged the launch of many applications dedicated to language learning. These applications are usually developed for the various work environments that are the product of this globalization. The most popular platform to develop applications of these characteristics is undoubtedly the Web. The Web not only allows easy access to resources, but also allows interaction through learning communities.

There are different learning methods. Formal learning is the strictly academic one that was traditionally applied in classrooms, although there are currently on-line alternatives (i.e. e-learning and b-learning). On the other hand, informal learning has the main characteristics of not being structured and support different learning rythms. Finally, unlike informal learning, non-formal learning is structured; and unlike the formal one, it is not strictly academic. As in formal learning, learning resources are produced professionally [1]. These types of non-formal materials are often designed for educational purposes with a structured progression. While this form of learning can be

offered as in-class, it is more common to find it on-line (eg Busuu [1], Duolingo [2], Babbel [3], etc.).

The development of such applications is complex due to the diversity of language learning methodologies, the variety of execution environments (desktop, mobile and web) and the amount of different technologies that can be used. The preparation of learning exercises to implement a language learning application is a complex, tedious and repetitive process, which involves the repetition of resource management tasks, the generation of duplicate code that is difficult to maintain and prone to errors, etc. These problems are further aggravated by the large number of deployment platforms in which applications must be available (e.g. iOS, Linux, Windows, Web, etc.).

A survey of the published literature [2] revealed that much of it justified the cognitive advantages of visual languages in terms of concepts from popular psychology such as the left-brain right-brain dichotomy. This was in spite of published research into the usability implications of diagrammatic notations, dating back to the 1970s [3]. Empirical investigations of notation usability have tended to find that while diagrams are useful in some situations, they are not efficient in others (e.g. [4]).

Given these circumstances, languages, tools and techniques have emerged in recent years to assist designers to support instructional design [4], either in Spanish or English, of learning scenarios [5] that include the configuration of Learning Management Systems (LMS) or Technology Enhanced Learning (TEL) systems.

As we will see later, there are many approaches that support models and visual languages through editors that allow designers to express the mental and conceptual models to communicate the work to be developed [6]. Thus, the complexity of creating learning exercises is increased by the complexity of creating tools that allow manipulating visual artefacts through specific notations associated with specific domain languages capable of representing them, also known as Visual Instructional Design Languages (VIDL) .

These tools should be designed to help professionals specify learning scenarios with specific terminology and graphic formalism. In addition, these specifications must be interpretable by computers.

Therefore, the objective of this proposal is the definition and implementation of a graphic language capable of representing activities to learn languages. In particular, the goal is the development of a language learning methodology editor capable of constructing representations of language learning activities that be interpretable by computers regardless of the execution platform.

The potential end users of this editor would be both, language teaching academics and people who are familiar with TEL systems. The editor will assist them in modelling and specifying methodologies to increase the reuse of learning resources (e.g. workflow , multimedia resources, exercises, etc.).

To ensure that the specifications are interpretable by computers supporting different execution platforms, we will apply an approach based on Domain Specific Modelling (DSM).

The implementation of the DSM will be based on the specification of an abstract syntax based on the metamodel presented in [7] and explained in detail in [8], which supports the development of language learning applications that separates the definition of language learning methodologies in 5 concerns. These 5 concerns structure the learning methodologies in terms of concepts, multimedia resources, user interface, activities and their workflow.

[1] Busuu home page. URL = `https://www.busuu.com/` (last access 09/26/19)

[2] Duolingo home page. URL = `https://www.duolingo.com/` (last access 09/26/19)

[3] Babbel home page. URL = `https://www.babbel.com/` (last access 09/26/19)

[4] Instructional design. URL = `https://es.wikipedia.org/wiki/Instructional_Design` (last access 09/20/2019)

Thus, the specific syntax associated with this metamodel is implemented based on the graphic editor proposed in this work. Models built with the editor are used to generate the source code of the language learning application modeled with the editor using transformation rules expressed in the ATL and ACCELEO languages, maximizing code reuse and minimizing maintenance costs.

The structure of this article is as follows. In Section 2 we will see how this work relates to MDA/MDE technologies, and we will discuss about work related to our proposal. The aim of Subsection 2.3 is introducing the reader to the domain of language learning activities which are implemented as part of the language learning methodologies deployed in the Internet. In Section 3 we will present the graphical notation for the abstract syntax defined in the metamodel shown in Fig. 1 and presented in [8].

Section 4 presents the evaluation of the notation described in Section 3. Finally, in Section 5 we will show the conclusions of this work and future work.

2 Related work

To address the solution to the aforementioned problem, in this article we will use DSM [9] as a development methodology, applying the principles and techniques defined in the standard of Model Driven Architectures (MDA) within the context of Model Driven Engineering (MDE) [10].

In this section we will explain how this work is related to these technologies (subsection 2.1) and we will discuss some works that are related to our proposal (subsection 2.2). Besides, in Subsection 2.3 we will introduce to the domain of language learning activities which are implemented as part of the language learning methodologies.

2.1 How this work relates to MDA/MDE

The most important principle of the MDE is the creation of non-contemplative productive models (i.e. interpretable by computers); therefore, they must be well defined, that is, they have to correspond to a specific metamodel. In this way, productive models can be managed and interpreted with MDE [11] tools.

The DSM is a flexible model-based approach that is based on the definition of different Domain Specific Modelling Languages (DSML) defined by experts in the different domains covered by the system; instead of a single modelling language like Unified Modelling Language (UML) [12].

A DSML is defined in terms of three basic components: *abstract syntax*, *concrete syntax* and *semantics*.

The *abstract syntax* is specified by a metamodel that describes the concepts of language, the relationships between them and the structuring rules that restrict the elements of the model and their combinations to represent the domain rules. In our case, the metamodel is the one shown in Fig. 1, and presented in detail in [7] and [8].

The *concrete syntax* of a DSML is defined as a mapping between the concepts of the metamodel and its textual or graphic representation providing users with a more intuitive notation for the representation of models. In this paper we present the definition of a specific syntax to represent language learning methodologies.

Finally, the *semantics* of a DSML describes the precise meaning of its models through model transformations that support the update *in situ* [13] in terms of the allowed actions. UML profiles are the mechanism that UML provides for the definition of DSMLs. Some proposals that use this mechanism for the definition of learning systems are presented in [14] and [15].

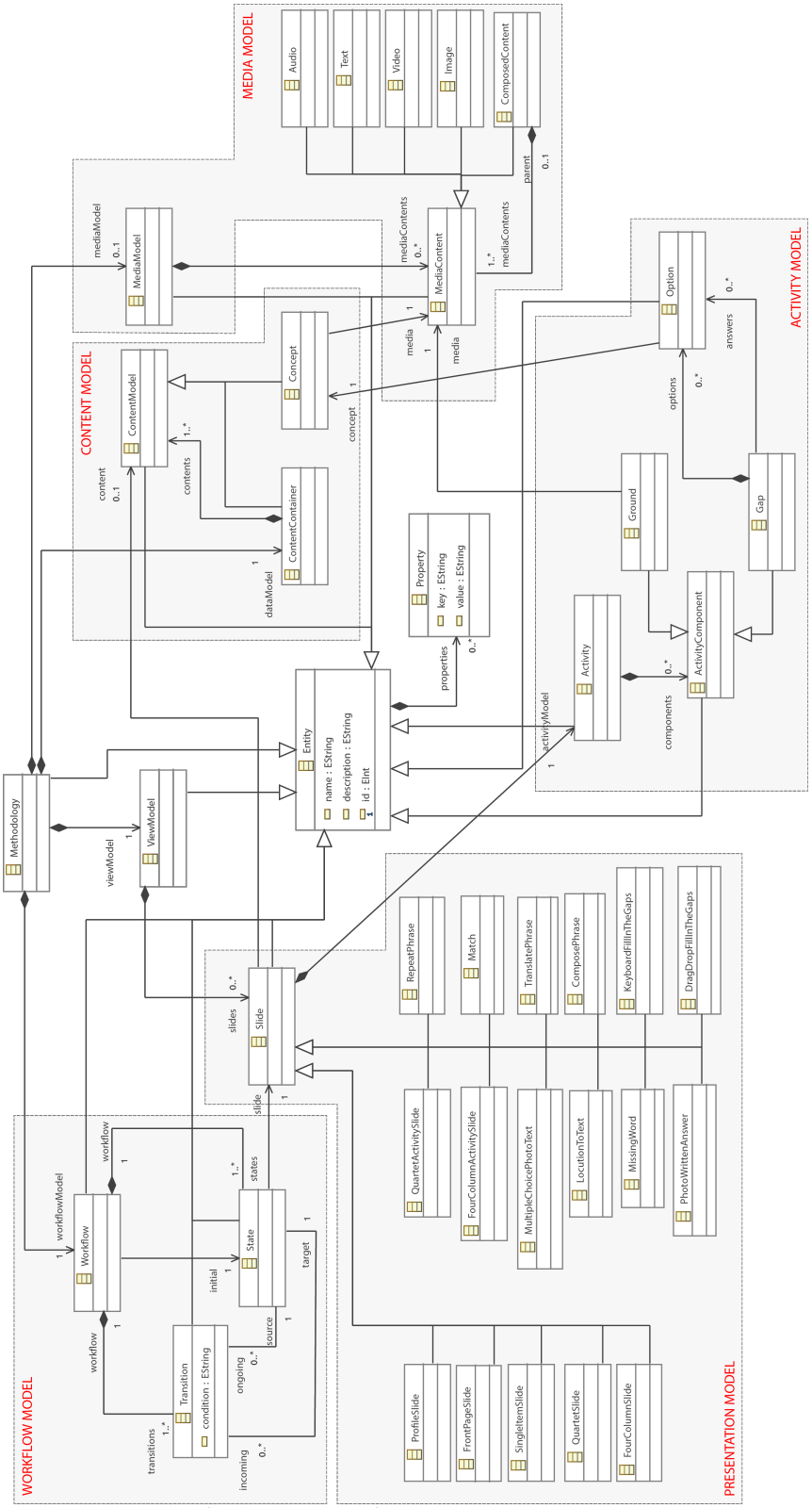


Fig 1. The metamodel (abstract syntax)

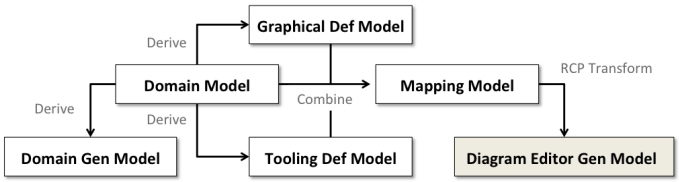


Fig 2. Model editor development workflow

However, the use of UML profiles implies that DSML users must know UML, which determines the number of end users of the language. This limitation can be saved by using a specific editor for the DSML, instead of defining an extension to UML.

Several technical approaches coexist to develop DSMLs [16]: MetaCase/MetaEdit + [5], Microsoft Visual Studio DSL Tools [6], EMF and GMF [7].

All these tools allow to define both the abstract and the concrete syntax of the DSMLs as well as facilities for the persistence of the built models. In addition, some of them provide support for the execution of transformations, model-to-model, and model-to-text.

In our case we have decided to work with the tools provided by the Eclipse Modeling Project (EMP) [8] with the objective of creating plug-ins for Eclipse IDE that implement the DSML. The EMP includes the Eclipse Modeling Framework (EMF) that allows the generation of modifiable code to specify metamodels, edit instances of models through a basic reflective editor (in the form of a tree, table or properties). It also provides runtime support that allows XMI notification, persistence and serialization of the models. However, it is noted that the basic reflective editor is insufficient in terms of usability, at least for complex metamodels, such as those used in the creation of language-learning methodologies.

The EMP also includes the Graphical Editing Framework (GEF) that allows developers to create a graphic editor from a metamodel generated with EMF. Another framework included in the EMP is the ATLAS Transformation Language (ATL) [9] that allows you to define model-to-model transformations.

From the point of view of VIDL, the Graphical Modelling Framework (GMF) [10] provides a generative component and a runtime infrastructure to develop graphic editors of EMF-based diagrams and GEF using an MDE approach. In addition, GMF allows us to define restrictions in OCL [11] to define rules to build well-formed models. Fig. 2 illustrates the workflow and domain models (ECORE), used during GMF-based development: the reflective editor code generator (*Domain Gen Model*), the Graphic Definition model (*Graphical Def Model*), the Tool Definition Model (Tooling Def Model), the Mapping Model (Mapping Model) and the code generator of the diagram graphic editor (*Diagram Editor Gen Model*).

The *Domain Model* is specified when defining an Ecore model (meta-model name for EMF/GMF, in reference to the Eclipse meta-meta-model, called Ecore, to define

[5]MetaCase/MetaEdit +. URL =`https://www.metacase.com/products.html` (last access 11/23/19)

[6]Microsoft Visual Studio DSL tools. URL = `https://docs.microsoft.com/en-us/visualstudio/modeling/overview-of-domain-specific-language-tools?view=vs-2019` (last access 11/23/19)

[7]GMF. Eclipse Graphical Framework. URL = `https://www.eclipse.org/gmf-tooling/` (last access 11/23/19)

[8]EMP. Eclipse Modeling Project. URL = `https://www.eclipse.org/modeling/` (last access 11/23/19)

[9]ATLAS Transformation Language. URL = `https://www.eclipse.org/atl/` (last access 11/26/19)

[10]GMP. Graphical Modeling Project. URL =`https://www.eclipse.org/modeling/gmp/` (last access 11/23/19)

[11]Object Constraint Language (OCL) specification. URL = `https://www.omg.org/spec/OCL/` (last access 11/23/19)

meta-models) with the editor from EMF or importing the model from other Ecore providers. The Graphic Definition Model is formalized through the use of a formula-based wizard, which guides and generates a graphic definition model of the previously specified domain model. The Graphic Definition Model contains information related to graphic elements (figures, nodes, compartments, links, etc.) without direct connection to the elements of the domain model for which they will provide representation and editing. The Tool Definition Model is specified using another wizard; It is used to design the palette and other visual controls, such as menus, actions and toolbars. The Mapping Model is specified, with the help of another assistant, linking the three previous models: the domain, the graphic definition and the definition of tools. In this way, GMF allows you to reuse the graphic definition for several domains. A separate Mapping Model is used to link the graphic and tool definitions to the selected domain models. Once the appropriate assignments are developed, a Generator Model is created to allow the implementation details or the code generation phase to be defined. Finally, the Diagram Plug-in is generated and then tested in a new Eclipse Application at runtime. When a user is working with a diagram, at runtime, the notation and domain models are joined providing both persistence and synchronization. Instead of generating workbench plug-ins, GMF can be used to generate diagram editors as Rich Client Platform (RCP) [12], that is, independent applications.

2.2 Related work on visual learning designs

E-learning is one of the mostly address fields of application of the MDA papers [17]. Complex Learning Processes (CLPs) are represented using Educational Modeling Languages (EMLs). IMS Learning Design (IMS-LD) [13]) is a commonly used EML for which some visual editors are being created that help the authoring process of learning scenarios (learning designs).

In [18] authors describe the FLEXO language, a Domain Specific Language (DSL) based upon the definition of a generic IMS-LD. A visual representation of FLEXO can be provided using an editing tool that abstracts the text-based specification of courses. The FLEXO language delivers the course in a variety of EMLs, as required by the execution environment or LMS. It hides most of the technical details from the designer, and can be easily extended to deliver other formats. The designer must only be aware of the possibilities of the target language, but not of how to use them. If the target platform in which the learning design has to be run is changed, the course can be generated from the FLEXO specification with scarce modifications.

COLLAGE [19] is too a high-level creation tool compatible with IMS-LD specialized in Computer-Supported Collaborative Learning (CSCL). COLLAGE helps teachers in the process of Create your own potentially effective collaborative learning designs by reusing and customizing patterns, according to the requirements of a particular learning situation. These patterns, called Collaborative Learning Flow Patterns (CLFP), represent best practices that are reused by professionals when structuring the flow of learning activities (collaborative).

In [20], the authors also present a visual authoring tool that complies in this case with the LPCEL (Learning Process Composition and Execution Language) specification proposed in [21]. The objective of this tool is also to facilitate the process of creating learning scenarios, with the advantage that the level of expressiveness of LD-LPCEL (as objective EML) is broader than IMS-LD. So, the LPCEL Editor provides a broad level of expressiveness and facilitates the authoring process with an editor that includes: (1)

[12] Rich Client Platform applications. URL = `https://wiki.eclipse.org/Rich_Client_Platform` (last access 11/26/19)

[13] Learning Design Specification. URL=`http://www.imsglobal.org/learningdesign` (last access 10/30/19)

Visual Elements, (2) Intermediate Representation, (3) Learning Patterns, (4) Collaboration tools and (5) Web Services

In [22] authors proposes an authoring system, refered to as ASK Learning Design Tool (ASK-LDT), that utilizes the LD principles to provide the means for designing activity-based learning services and systems. The ASK-LDT relies on user-defined visual representations of IMS-LD level A formalisms from which an XML-based description of the target Unit-of-Learning (UoL) can be generated. Visual shapes, icons, connectors and all source elements are kept close to the core IMS-LD level A model.

The Model-Driven Learning Design (MDLD) [23] proposes a set of visual abstractions for the authoring process and the mechanisms to generate XML files compliant with an EML. In [24], the core of the approach is the structural and operational specification of a DSL used to describe key aspects of the final learning application. On the other hand, in [25] a language is proposed for the high-level design of interactive applications created in accordance with the model-view-controller pattern. This approach is especially suitable for applications that incorporate content with sophisticated structures and whose interactive behavior is driven by these structures. In both [24] and in [25], the resulting designs are compatible with rapid prototyping, exploration and early discovery of application features, and systematic implementation using technologies web based standard. Also, these two approaches facilitate active collaboration between instructors and developers throughout the process of developing and maintaining e-learning applications.

Gamification has been proven to increase engagement and motivation in multiple and different non-game contexts such as education. In [26] authors describe a graphical modeling editor for gamification domain definition and a graphical modeling editor for gamification strategy design, monitoring and code generation in event-based systems. The solution makes use of Model-Driven Engineering (MDE) and Complex Event Processing (CEP) technology. This work also shows how the proposal can be used to design and automate the implementation and monitoring of a gamification strategy in an educational domain supported by a well-known Learning Management System (LMS) such as Moodle.

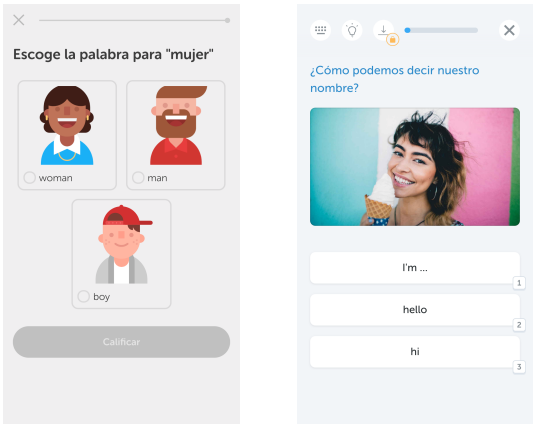
Finally, the following research works [27], [28], [29], and [30], although they do not apply to the field of e-learning, are also related to VIDL and are related to our work by their approach or by the tools they use for their development.

2.3 Analyzing language-learning methodologies

The aim of this section is introducing the reader to the domain of language learning activities which are implemented as part of the language learning methodologies deployed in the Internet. Although the implementation of most of these methodologies has different look and feel; they share common characteristics such as the Concepts to Learn (CtL), the structural organization of contents (i.e. levels, blocks, lessons, activities, etc.), multimedia resources (i.e. audios, texts, images, etc.), Learning Activity Mechanisms (LAMs), and so on.

For instance, the Fig.3 depicts the multiple-choice LAM in two different learning methodologies. While Fig.3 (a) depicts the Duolingo[14] version of the mechanism using radio buttons associated to images to display available options, and a button to confirm the option selection; the Fig. 3 (b) depicts the Bussu[15]) version the employs a set of buttons labelled with the available options instead of the radio buttons. Moreover, both versions share common characteristics in the presentation too (e.g. providing student lesson progress at the top of the language learning activity UI).

[14]Duolingo. URL = `https://www.duolingo.com` (last access 11/19/2018)
[15]Busuu. URL = `https://www.busuu.com` (last access 11/19/2018)



(a) (b)

Fig 3. The multiple-choice learning activity in Duolingo (a) and Busuu (b)

This is just one example of the similarity among language learning methodologies, since it is not the aim of this paper to perform a review of language learning methodologies.

3 Graphical Notation Description

The graphical notation for the abstract syntax defined in [8] was implemented as an Eclipse IDE plugin using the Eclipse Graphical Framework (GMF)[16] that is part of the Eclipse Modeling Project (GMP)[17].

The purpose of the specific syntax of a DSL is to provide users with an intuitive and friendly notation similar to others they normally use. In the case of this work, this notation makes easier for users to specify the models of a language learning methodology. The concrete syntax is usually defined as a mapping between the concepts of the meta-model and its textual or graphic representation. Thus, to define these visual representations, it is necessary to establish links between these concepts and the visual symbols that represent them. For all this, we have chosen visual symbols for our graphic editor so that they are as intuitive as possible. A complete example illustrating the concrete syntax adopted can be seen in Fig. 5.

The description of the graphical notation will be carried out following the description tables presented in [31], which are described in the tables 1 and 2.

The Graphical User Interface (GUI) of our editor consists of the following components: a title bar, a properties panel, a tool palette and the workspace. Fig. 4 shows the distribution of these elements in the GUI.

Title bar. In this bar, both the name of the application and the name of the current job file are displayed.

Properties Panel. In this panel, the user can view and edit the properties of the selected elements in the workspace.

Tools palette. The elements of the tool palette are organized into five categories: Contents, Resources, Presentation, Activities and Control Flow.

The **Workspace** is the canvas in which we place all the elements that will define the methodology that we want to model with the graphic editor. In the Fig. 5, an

[16]GMF. Eclipse Graphical Framework. URL=`https://www.eclipse.org/gmf-tooling/` (last access 09/23/19)

[17]GMP. Graphical Modelling Project. URL=`https://www.eclipse.org/modeling/gmp/` (last access 09/23/19)

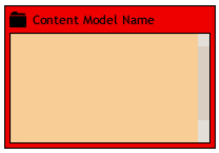
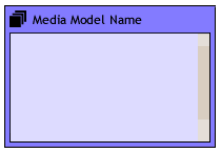
Concept & Graphical representation	**Rationale** (shape/icon & color)
Content model 	**Shape**: Rectangular window that will contain each of the levels and sub-levels of tree-shaped content, as well as the concepts or words (content tree sheets). In the header of the window we show a representative icon and the model name (normally it corresponds to the methodology or level of the methodology we are modelling; for example, Level 11 of Duolingo). See an example in Fig. 5 (a). In Fig. 6 (b) we visually show the association of contents of the previous example, and in Fig. 6 (a) another example (in this case of Busuu methodology). **Color**: The head of this window is painted red (and the canvas in faint red).
Content levels 	**Shape**: We represent them with the metaphor of a folder, because they contain concepts (or sub-levels of content). **Color**: Red icon
Concepts or words 	**Shape**: We represent them with an icon of a concept, because they represent the new concepts or words that are studied in a lesson. **Color**: Red icon
Media model 	**Shape**: Rectangular window that will contain each of the levels and sub-levels of average resources. In the header we show a representative icon and the model name. See an example in Fig. 5 (b). On the right side of Fig. 7 the resource association of the previous example is shown. **Color**: This window is painted violet (and the canvas in a faint violet).
Media levels 	**Shape**: We represent them with the metaphor of a folder, because they contain resources made up of several resources (or resource sub-levels). In the one in Fig. 7 we show the media resources of a multiple-choice activity in the Duolingo methodology, to illustrate what we mean by "compound resources". **Color**: Violet icon.
Media 	**Shape**: We represent them with a representative icon of the type of resource in question: text, audio, video or image. **Color**: Violet icon.

Table 1. Description of the graphic notation (Content model and resource model)

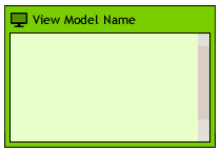
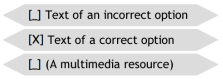
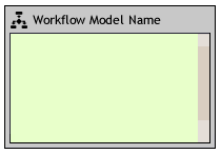
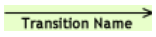
Concept & **Graphical representation**	**Rationale** (shape/icon & color)
Presentation and activity model 	**Shape**: Rectangular window that will contain each of the slides and activities. In the header we show a representative icon and the model name. See an example in Fig. 5 (c). In Fig. 9 we illustrate the modelling of a Busuu FillInTheGaps activity. **Color**: We paint this window green (and the canvas with a very faint green).
Statements and passive contents 	**Shape**: Icon of a generic media resource. **Color**: Green icon.
Gap 	**Shape**: We represent them with an icon of a gap. **Color**: Green icon.
Option 	**Shape**: We represent them with a rhomboid rectangle with a checkbox (checked if it is the correct option), and with the text of the option. If the option is of type text, the corresponding text is partially shown; and if the option is a multimedia resource, the name of the multimedia resource is shown in parentheses. **Color**: Gray rhomboid rectangle.
Workflow Model 	**Shape**: Rectangular window that will contain the states, transitions and sub-workflows. In its header we show a representative icon and the name specified for the model. An example can be seen in Fig. 5 (d). In Fig. 8 another workflow that also has a sub-workflow is shown. **Color**: This window is painted gray with a faint green background.
State 	**Shape**: Rectangle with a representative icon, the name of the state and the description of the state. **Color**: Green rectangle. The states (and the background of the Workflow Model) are painted green because they are closely related to the slides and activities of the Activity Model.
Transition 	**Shape**: We represent them with an arrow (which connects the source state with the destination state) together with a descriptive label. **Color**: Black arrow.

Table 2. Description of the graphical notation (Presentation and workflow models)

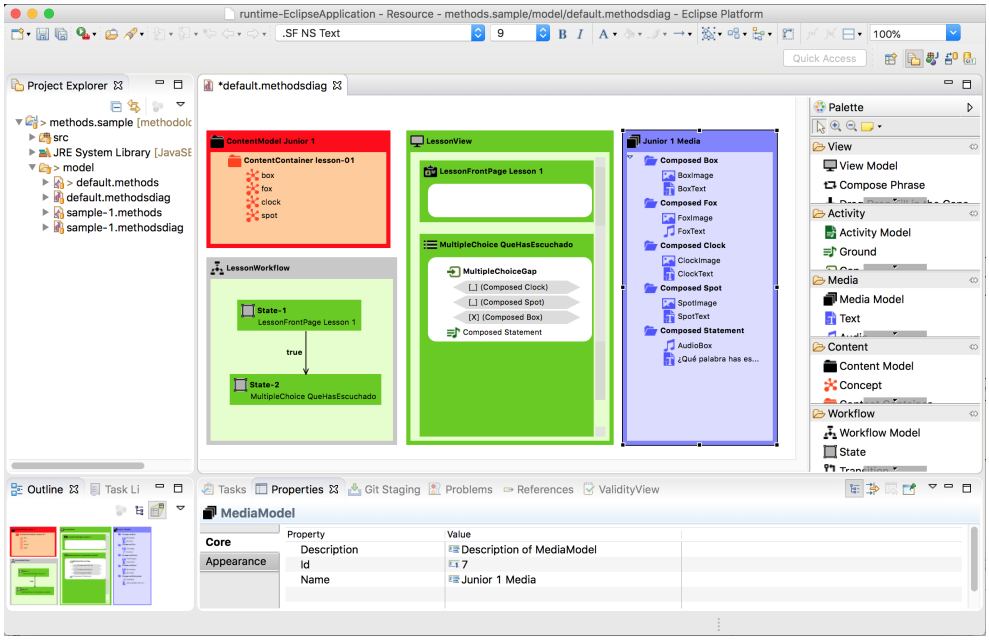


Fig 4. Editor User Inteface

example of the content of the "Workspace" is shown.

As we have already mentioned, the diagram of any methodology consists of 4 sub-diagrams within it: the sub-diagram of the **Contents**, the sub-diagram of the **Resources**, the sub-diagram of the **Slides**, and the sub-diagram of the **Control flow**. In the 4 cases we will use 4 containers of graphic elements, each of which is distinguished by an identifying icon, a description in its header bar, and in general by a color associated with the mentioned sub-diagram. To illustrate this, in Fig. 5, the 4 sub-definitions of a small part of Duolingo's methodology and activities are shown.

For the Contents we will use the paradigm of a tree-like structure to express the different levels and sub-levels of the contents of a level of the language learning method that we are defining. For example, the Duolingo methodology is structured in Levels, which in turn are configured by Units, which consist of Lessons, and in which a series of Concepts or Words are used to learn. See this illustrated in Fig. 6 (b).

For Resources we will also use the paradigm of a tree-like structure to define the library of multimedia resources that will be used in interactive activities. Basically, folders or resource containers are used, which may contain sub-folders. These resources or media can be of 4 types (audio, text, video and image), which can be combined to represent complex media (eg text and speech or eg image and text, etc.). See this illustrated in Fig. 7.

To model the control flow with the graphic editor, we have defined a state diagram. This can be seen in Fig. 8.

Finally, to model the activities, like all of them are modeled according to the paradigm of the exercises to fill gaps, they are always modeled with resources for the statement (Ground) and with gaps with possible answers, valid or not (see Fig. 9).

4 Notation Evaluation

This section presents the evaluation of the notation described in Section 3. This evaluation uses the Cognitive Dimensions of Notations framework for Notational

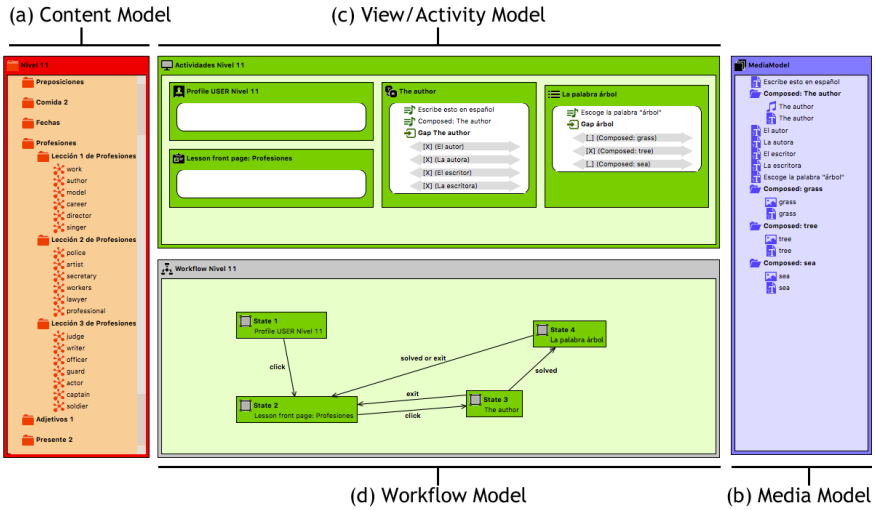
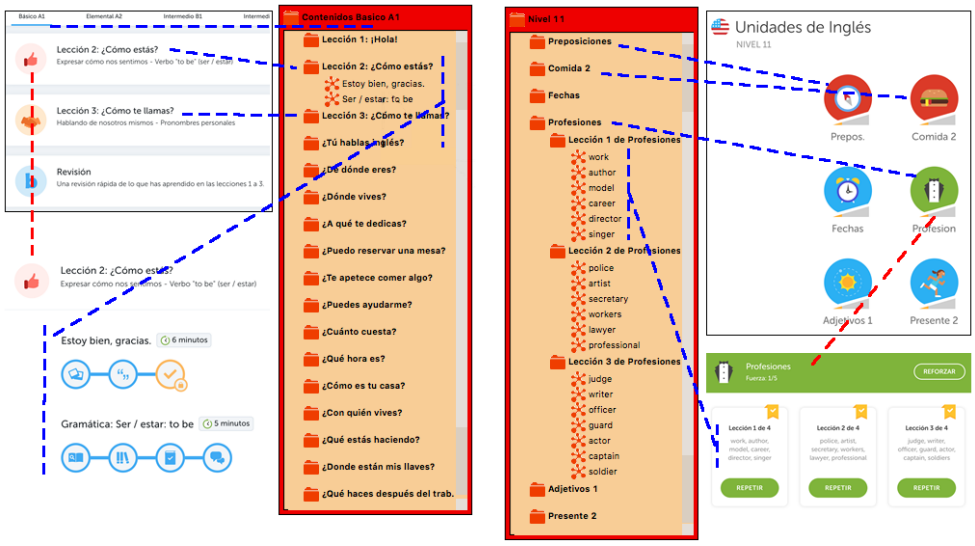


Fig 5. The 4 sub-diagrams that make up the diagram of a methodology (as Duolingo)



(a) (b)

Fig 6. Tree structure to represent content in Busuu (a) and Duolingo (b)

Fig 7. The media resources of a multiple-choice activity in the Duolingo methodology

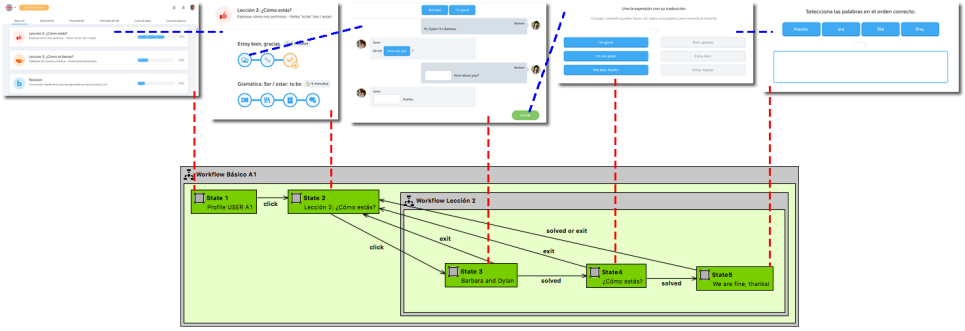


Fig 8. The workflow of 5 slides in the Busuu methodology

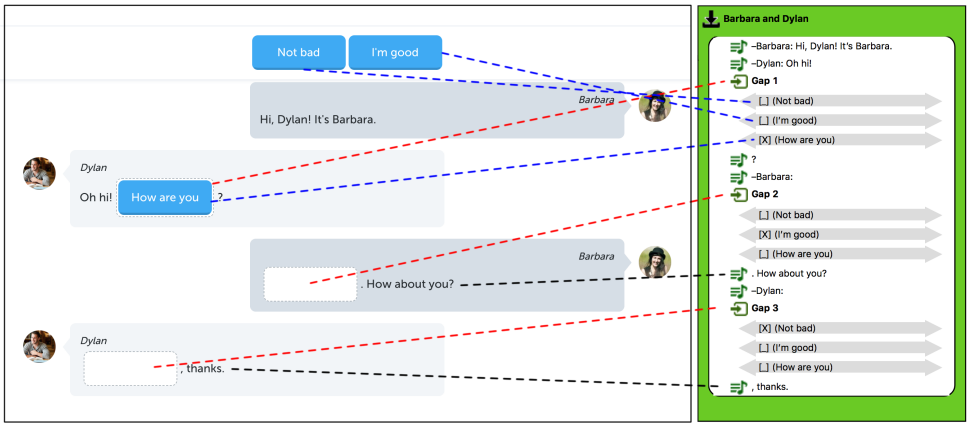


Fig 9. Graphical model of a Busuu FillInTheGaps activity

systems presented in [32] and extended in [33] to provide a common vocabulary for use by user interface designers to conduct comprehensible evaluations. It is solely based on structural properties; it does not representational issues (i.e. effectiveness and aesthetics are outside its purview). Therefore, the question addressed by the Cognitive Dimensions (CDs) framework is: *are the users' intended activities adequately supported by the structure of the information artifact?* and if not, *what design maneuver would fix it, and what trade-offs would be entailed?*.

This evaluation, and the conclusions of this work (Section 5), also offers the comparison of the reflexive model editor with the diagram model editor supporting the proposed notation. The reflexive model editor is the *de facto* default model editor used in the Eclipse Modeling Tools (EMT)[18] (Eclipse IDE distribution to edit models). It supports a tree-based notation that employs reflexive properties of metamodels to build models conforming these metamodels.

The procedure to carry out this evaluation starts with the classification of intended activities; it continues with the analysis of the cognitive dimensions; and it decides if activities' requirements are met. Note that the term "activity" refers to a notational activity in cognitive dimensions as is a manner of interacting with a notation and it is not related to Activity Theory, or Active User Theory.

4.1 Activities

The CDs framework contrasts the following 6 generic types of notation use activity on artifacts based on contributions from a variety of researchers [32]: **incrementation** (e.g. add a media resource to a repository), **transcription** (e.g. set a question text using a media resource), **modification** (e.g. change the media resource to render a multiple-choice option from text to image), **exploratory design** (e.g. define a multiple-choice using audio resources as options instead of text), **searching** (e.g. look for the set of learning activities related to a CtL), and **exploratory understanding** (e.g. discover next learning activity given a starting activity and the result of a user interaction).

In the Table 3, we list a series of generic activities and show to what extent they satisfy or not each of the 6 types of usage notations mentioned above and exemplified.

Each activity imposes different demands on the system; thus, nothing marked in Table 3 is properly good or bad, unless an activity does not operate as is expected to do.

Some comments regarding the Table 3 of Activities and types of notation use, are the following:

- All activities, except those of pure editing of properties, involve the use of "Incrementation".
- All activities offer and require a use of "Transcription".
- The 4 general submodels are designed to be created only once, and do not require any "Modification". The ones that will have a type of "Modification" notation will be the Management activities of the mentioned submodels.
- "Exploratory design": Only the states of a Workflow cannot be predicted directly (visually) and must be ' 'discovered" using the properties window.
- "Searching": Only the Media, the Concepts, are easily searchable in the corresponding Resources or Content trees. Also the creation and editing of Activities, because their elements are always visible.

[18] EMT. Eclipse Modeling Tools. URL=`https://www.eclipse.org/downloads/packages/release/juno/sr1/eclipse-modeling-tools` (last access 02/06/2020)

Activity	Incrementation	Transcription	Modification	Exploratory design	Searching	Exploratory understanding
Create View Model	X	X		X		X
Add a View or Activity	X	X	X	X		X
Add a Ground	X	X	X	X		X
Add a Gap	X	X	X	X		X
Add a Option	X	X	X	X		X
Create Media Model	X	X		X		X
Add a Media resource	X	X	X	X	X	X
Create Content Model	X	X		X		X
Add a Concept	X	X	X	X	X	X
Add a Content Container	X	X	X	X		X
Create Workflow Model	X	X				X
Add a State	X	X	X			X
Add a Transition	X	X	X	X		X
Add a Workflow Model	X	X	X			X
Manage View Model	X	X	X	X		X
Edit a View		X	X	X		
Edit a Ground		X	X	X	X	X
Edit a Gap	X	X	X	X	X	X
Edit a Option		X	X	X	X	X
Manage Media Model	X	X	X	X	X	X
Edit a Media resource		X	X	X	X	X
Manage Content Model	X	X	X	X	X	X
Edit a Concept		X	X	X	X	X
Edit a Content Container		X	X	X		X
Manage a Workflow Model	X	X	X	X		X
Edit a State		X	X			X
Edit a Transition		X	X	X		X

Table 3. Activities and types of notation use

- The activity of Editing a View is the only one that does not properly have an "Exploratory understanding".

So, we can affirm that the proposed graphic editor meets all the requirements of the activities.

4.2 Components

The CDs framework also defines 4 types of components: an **interaction language** or **notation**, an **environment** for editing the notation, a **medium** of interaction, and possibly **sub-devices**.

4.2.1 Interaction language or Notation

The **interaction language** or **notation** is defined as what the user sees and edits (i.e. the graphical notation presented in Section 3).

Everything related to the definition of a methodology, in our graphic editor is visually represented on the canvas. The distribution and sizes of the graphic notation, is editable by the user. Based on this general graphic representation, the user can configure each of the elements thanks to properties windows and pop-up windows. Some of these options or configuration properties will be visible on the canvas, but others will not.

Additionally, there are many interaction resources related to what the user sees and edits; Some of those resources are:

- In all buttons, the user is shown a tooltip as a feedback that tells him what that button does.
- When a window or container, due to its current size, does not reveal its contents, the interface offers the user the possibility of interacting on a horizontal and vertical scroll-view, which allows him to visualize any part of the sub-canvas that initially appeared hidden.
- The editor interface allows the user to configure many aspects of the environment.

4.2.2 Editing environment

The **editing environment** and other interaction resources (e.g. for digram editors: drag and drop support, inline text edition support, zoom support, and so on) that facilitate editing activities. As it has been mentioned before, users visually manage certain visual diagrams with freedom and great versatility. We now list some interface aspects related to the editing environment:

- Drag&drop interaction to add slides, activities, states, and so on.
- Depending on which object is selected on the canvas, one or another context-sensitive sub-palette will appear, showing the user what operations he can do on that object at that time.
- To help the user, when moving the mouse on the canvas or on the objects contained in it, the cursor changes its aspect to let you know what the user can do.
- Once a transition is created, the arrow can be edited freely, moving it, changing the start and end position, and even creating new vertices.
- There are user operations that might trigger a feedback of violation of the integrity of the model.

#	Dimension	Aspect
1	Viscosity	Resistance to change
2	Visibility	Ability to view components easily
3	Premature commitment	Constraints on the order of doing things
4	Hidden dependencies	Important links between entities are not visible
5	Role-expressiveness	The purpose of an entity is readily inferred
6	Error-proneness	The notation invites mistakes and the system gives little protection
7	Abstraction (redefinitions)	Types and availability of abstraction mechanisms
8	Secondary notation	Extra information in means other than formal syntax
9	Closeness of mapping	Closeness of representation to domain
10	Consistency	Similar semantics are expressed in similar syntactic forms
11	Diffuseness	Verbosity of language
12	Hard mental operations	High demand on cognitive resources
13	Provisionality	Degree of commitment to actions or marks
14	Progressive evaluation	Work-to-date can be checked at any time

Table 4. Summary of notational dimensions

- During the specification of an activity, it is easy to manage options, correct answers, and so on, through property windows, drop-down, parallel lists with elements bidirectionally transferable from one to another to specify, among the possible answers, which we establish as correct answers, and so on.
- When creating activities, states, or other types of entities, the interface facilitates the user's work by making certain auto-fill of default values (e.g. automatic generation of object names).

4.2.3 Medium of interaction

The **medium of interaction** is the medium used to interact with the notation language (e.g. touch screens, displays, and so on).

In our case, being a graphic editor managed with a desktop computer, the means of interaction are monitors (as an output device) and the mouse and keyboard (as input devices). It is a persistent medium preceded by many fully editable transitional actions. Although the order of the actions is free, to guarantee the integrity of the model, in some cases the editor restricts the actions that can be performed at any time or according to the context.

4.2.4 Sub-devices

We now mention the two **sub-devices** supported by the main device. While the **helper** sub-device offers cross-references (e.g. references from *Workflow model* to *Presentation model*, diagram outline view, and so on); the **redefinition** sub-device enables the main notation changes (e.g. collapsing model views to get a high level view).

4.3 Notational dimensions

In this evaluation we have also considered the 14 notational dimensions defined by the CDs framework presented in Table 4, where each dimension describes an aspect of an information structure that is reasonably general.

4.3.1 Viscosity

Cross references are automatically redrawn in the diagram. For example, if we change the name of a resource, it is automatically modified in all the places in the diagram where it is referenced.

4.3.2 Visibility

As we have already mentioned in Subsection 4.2, everything related to the definition of a methodology, in our graphic editor, is visually represented in a diagram on the canvas. Logically, there are properties that remain hidden, but these are easily accessible, as appropriate, in the properties window.

4.3.3 Premature commitment

The editor, depending on the context, provides restrictions on the order of the operations. For example, depending on which object is selected on the canvas, one or another context-sensitive sub-palette will appear, depending on the operations I can do on that object at that time; In addition, operations that could be performed in other contexts are disabled.

4.3.4 Hidden dependencies

Initially the diagram did not graphically show the dependencies of the elements of the Content Model (concepts/words) with the slides/activities in which these concepts/words are worked on or studied. Later, we implemented that these dependencies are shown graphically, but, as many lines are generated on the diagram and the canvas, we have left the default option for these dependencies to remain hidden.

4.3.5 Role-expressiveness

In general, the purpose of all the entities that make up the diagram of a methodology is easily deduced. Certainly, in the case of the graphic expression of the operation (contents) of some activities, it is not easy to discover what the author wants to model. When all the activities are based on the fill-in the gaps activities paradigm, all of them are seen in the diagram in a very similar way. It is therefore necessary for the user to understand the dynamics - for example - of the creation of an activity of *Match* based on the paradigm of the exercises of fill-in the gaps. Once you understand this dynamic, it is easier and more obvious to understand what another author has modelled.

4.3.6 Error-proneness

The diagram can be validated at any time. In addition, when the user tries to add an entity inside a container, thanks to the dynamic change of the appearance of the cursor (while mouse over the different models, activities, etc.), the user knows whether or not such operation is allowed. If it is not allowed, the user will not be able to make the mistake of doing it.

4.3.7 Abstraction

As we saw in [8], the "*fill-in the gaps*" abstraction is at the base of the modelling of all types of activities supported by the editor. This is the main abstraction mechanism available in the editor, but other new abstraction mechanisms cannot be managed or created.

4.3.8 Secondary notation

In our editor 3 types of secondary annotations that users can freely use are supported. We refer to: (1) the text boxes for making annotations on the canvas of the diagram, (2) the adhesive notes, as well as (3) the relationship between an adhesive note and an entity of the diagram.

4.3.9 Closeness of mapping

In general the diagram of a methodology, that is, the *Content model*, the *Media model* and the *Workflow model*, is closely related to what it describes. It is not so obvious in the case of *View model* (*Activity model*). As we have already mentioned, being the "*fill-in the gaps* "abstraction at the base of the modelling of all types of activities that the editor supports, the modelling is not always visually evident and is closely related to the result that it describes. However, it is in the case of most activities, such as those of type *Image Audio Text Options*, *Image Text Options*, *Question Answer*, *Gap*, *Listening*, and *Selection*.

4.3.10 Consistency

In relation to the *Consistency*, two aspects stand out:

1. We identify the containers with the icon of a folder, of one colour or another, as appropriate, both in the context of a *Content Model* or a *Media Model*.
2. The housing of the diagrams of all the Models (packages) are represented in a similar way (with a window, an identifying icon, and a descriptive name), but with different colours for better identification and differentiation.

4.3.11 Diffuseness

There are no long annotations, nor they occupy valuable spaces within a viewing area. In any case, if this happened, the user can at any time resize and relocate any entity of the diagram that is generating, if he considers it appropriate to facilitate the readability of the diagram. In addition, you can minimize some parts of the diagram that may no longer need to be displayed, to make room for other more valuable parts of the diagram at any given time.

4.3.12 Hard mental operations

As already mentioned in the Sub-subsections 4.3.7 and 4.3.9, the readability of modelling of some types of activities requires more cognitive resources from the user. This is the case of activities of type *Compose Phrase*, *Match*, and so on.

4.3.13 Provisionality

Users at any times (thanks to the restrictions that avoid the most obvious errors of inconsistency of the model) can take provisional actions or *playing "what-if" games*. In addition, at all times, after such provisional or trial actions, the user, by validating the diagram, can know whether or not it is consistent.

4.3.14 Progressive evaluation

Users, as just mentioned in the Sub-subsection 4.3.13, can verify their work at any time. Our editor allows you to validate any part of the diagram (as well as its entirety), at any time, progressively verifying the stage of the work you are doing or where you are up to that moment. In the context menu, which can be accessed by having selected any entity in the diagram, the user always has the option *Validate* available, which provides the necessary feedback to locate and resolve errors.

#	Dimension	PE score	RE score
1	Viscosity	X X X X X	X X X X
2	Visibility	X X X X	X X X
3	Premature commitment	X X X X	X X X
4	Hidden dependencies	X X X X	X X X
5	Role-expressiveness	X X X	X
6	Error-proneness	X X X X	X X X
7	Abstraction	X X X	X
8	Secondary notation	X X X X X	X
9	Closeness of mapping	X X X X	X X
10	Consistency	X X X X	X X
11	Diffuseness	X X X X X	X X
12	Hard mental operations	X X X	X
13	Provisionality	X X X X	X X X
14	Progressive evaluation	X X X X	X X X

Table 5. Score of notational dimensions for the proposed editor (PE) and for the reflective editor (RE)

4.4 Discussion

In this section, the results of the evaluation and the relationships between some of the notational dimensions are analysed.

First of all, each of the notational dimensions are rated from 1 to 5 (for the proposed editor and for the reflective editor), according to Subsection 4.3 (See Table 5). Although is an internal assessment of this work, it illustrates the graphic and usability power of the graphic notation of the developed editor. We have scored with the criterion that 5 means the maximum level of satisfaction by the user in relation to the notational dimension evaluated, and 1 the minimum.

First, in the Table 5 we can see that there are a number of outstanding aspects of the proposed editor: An adequate management and resistance to change (*Viscosity*), a complete offer of mechanisms non-formal to capture extra information (*Secondary notation*), and the absence of language verbosity (*Diffuseness*).

Secondly, it is also deduced from the Table 5, that the proposed editor also stands out for the right solutions (graphical and interaction) achieved in relation to other 8 important notational dimensions: *Visibility*, *Premature commitment*, *Hidden dependencies*, *Error-proneness*, *Closeness of mapping*, *Consistency*, *Provisionality* and *Progressive evaluation*.

Of course, some of these last aspects are related to each other: For example, the good score in *Premature commitment* and *Error-proneness* is due to the fact that the editor effectively offers restrictions in the order of making things, which in turn provides the user with adequate protection against possible errors. Likewise, the good score in the aspects of the *Provisionality* and the *Progressive evaluation*, indicate us that the editor has an adequate commitment to the actions, allowing the users to carry out provisional actions, and facilitating them to at the same time validating - at any time - what they are doing.

Finally, the aspects *Role-expressiveness*, *Abstraction* and *Hard mental operations*, which, although adequate, they received less score, are also related to each other (but significantly better than in the reflective editor). The adopted score responds to the fact that, as we have already mentioned in the corresponding subsections: (1) in the diagrams generated with the editor, the purpose of some entities is not always easily deduced, (2) the editor does not allow the definition of new mechanisms of abstraction, and (3) sometimes the user may not find it obvious what another author has modelled.

5 Conclusions

In this paper we have presented a graphical notation for a domain specific language to represent language learning activities. We have described how this notation enables developers to represent language learning activity characteristics using workflow, presentation, content, media and activity model conforming a metamodel that defines the abstract syntax of the domain specific language. This notation is implemented as part of an integrated development environment to build model-based applications. In addition, we have evaluated this proposal with a framework that uses the cognitive dimensions of notations for notational systems. Finally, in this work we have compared the proposed graphical notation to the traditional reflexive tree notation used by many model-based development frameworks.

We can conclude that the proposed graphic diagram editor exceeds the experience that the user has with the reflexive model editor. Although there is not much difference in the modeling of content models and resource models of methodologies, in relation to the creation and editing of workflow models and presentation/activity models, the solution provided -unlike the one offered by the reflective editor - it is intuitive and easy to maintain visually. We list some other more prominent conclusions:

1. The maintanment of the workflow and activity submodels are very tedious with the reflective editor, among other things because many elements and properties remain hidden ... With the proposed graphic editor you can easily and visually create and maintain the states, transitions between them and sub-workflows.

2. In the reflexive editor, the editing operations using *copy&paste*, are delicate, and sometimes, to find errors you have to go to the text editor, with all the difficulty that entails.

3. The proposed editor correctly generates the models that we then use as input in a process of transformations that generate source code (fully functional) in HTML and JavaScript.

As future work, we think we have to improve certain aspects of the "notation dimension" that we have commented and scored in Section 4.3 and in Section 4.4 respectively. In particular we will look at these aspects:

1. "Role-expressiveness ": Show more graphically what the author wants to model when designing activities that are not as obvious as an exercise in filling in gaps. This is the case of the activities of type *Word Ordering* and *Join* , because its design is not evident using the paradigm on which we always rely, " fill-in the gaps. "Actually this dimension (and the improvement outlined) is related to these other two " Abstraction" and "Hard mental operations."

2. "Visibility": We think it may be interesting to make it easier for users to preview multimedia resources. We refer to the Media subdiagram files, such as image, audio and video.

3. "Hidden dependencies": To facilitate the maintenance of workflow diagrams, we find it interesting to show or make more evident the relationship-dependence between states and views.

In addition, some additional work likes could be the following:

1. Incorporate in the editor the possibility of specifying and validating certain restrictions of each method (Duolingo, Busuu, etc.) in relation to for example: types of activities that it supports, restrictions of its content structure, types of media resources that it supports , etc.

2. Continuing with the previous aspect, although the different methods use the same types of activities, in some cases, they do not work visually exactly in the same way, as they may unequally require Ground resources to compose and paint the statements. In this sense, the current graphic editor is limited and should guide the user more when it comes to, for example, modeling activities of type *Dialog*, *Word Ordering*, or *Join*.

3. Improve the graphic aspect in general, eliminating aspects of the interface that distract the user with functionality provided by GMF, but which are not necessary in our case.

Acknowledgments

This work has been partially supported by the national project granted by the Ministry of Science, Innovation and Universities (Spain) with reference RTI2018-099942-B-I00 and by the project TecnoCRA (ref: SBPLY/17/180501/000495) granted by the regional government (JCCM) and the European Regional Development Funds (FEDER).

References

1. Sockett G. In: The online informal learning of English. Palgrave Macmillan UK; 2014. p. 11.

2. Blackwell AF. Metacognitive theories of visual programming: what do we think we are doing? In: Proceedings 1996 IEEE Symposium on Visual Languages; 1996. p. 240–246.

3. Fitter M, Green TRG. When do diagrams make good computer languages? International Journal of Man-Machine Studies. 1979;11(2):235 – 261. doi:https://doi.org/10.1016/S0020-7373(79)80019-X.

4. Green TRG, Petre M. When Visual Programs are Harder to Read than Textual Programs. In: In; 1992. p. 167–180.

5. Koper R. Editorial: Current Research in Learning Design. Journal of Educational Technology & Society. 2006;9(1):13–22.

6. Kopp G. Handbook of Visual Languages for Instructional Design: Theories and Practices. Canadian Journal of Learning and Technology. 2009;34. doi:10.21432/T2N30X.

7. Sebastian G, Tesoriero R, Gallud JA. Modeling Language-Learning Applications. IEEE Latin America Transactions. 2017;15(9):1771–1776. doi:10.1109/TLA.2017.8015084.

8. Sebastian G, Tesoriero R, Gallud JA. Model-based approach to develop learning exercises in language-learning applications. IET Software. 2018;18(3):206–2014. doi:10.1049/iet-sen.2017.0085.

9. Kelly S, Tolvanen J. Domain-Specific Modeling: Enabling Full Code Generation. Wiley-IEEE Computer Society Press; 2008.

10. Schmidt DC. Guest Editor's Introduction: Model-Driven Engineering. IEEE Computer. 2006;39(2):25–31. doi:10.1109/MC.2006.58.

11. Bézivin J, Gérard S, Muller PA, Rioux L. MDA components: Challenges and Opportunities. In: Workshop on Metamodelling for MDA. York, England, United Kingdom; 2003. p. 23–41. Available from: https://hal.archives-ouvertes.fr/hal-00448057.

12. Ráth I, Schmidt A, Vago D. Automated Model Transformations in Domain Specific Visual Languages. Scientific Students' Associations Report; 2005.

13. Czarnecki K, Helsen S. Classification of Model Transformation Approaches. In: OOPSLA03 Workshop on Generative Techniques in the Context of MDA. California, USA; 2003.

14. Laforcade P. Towards a UML-based educational modeling language. In: Fifth IEEE International Conference on Advanced Learning Technologies (ICALT'05); 2005. p. 855–859.

15. Laforcade P. Visualization of Learning Scenarios with UML4LD. Journal of Learning Design. 2007;2(2):31–42.

16. Jouault F, Bézivin J, Consel C, Kurtev I, Latry F. Building DSLs with AMMA/ATL, a Case Study on SPL and CPL Telephony Languages. In: ECOOP Workshop on Domain-Specific Program Development. Nantes, France; 2006.Available from: https://hal.inria.fr/inria-00353580.

17. Sebastián G, Gallud JA, Tesoriero R. Code generation using model driven architecture: A systematic mapping study. Journal of Computer Languages. 2020;56:100935. doi:https://doi.org/10.1016/j.cola.2019.100935.

18. Dodero JM, del Val ÁM, Torres J. An extensible approach to visually editing adaptive learning activities and designs based on services. Journal of Visual Languages & Computing. 2010;21(6):332 – 346. doi:https://doi.org/10.1016/j.jvlc.2010.08.007.

19. Leo DH, Villasclaras-Fernández ED, Asensio-Pérez JI, Dimitriadis YA, Jorrín-Abellán IM, Ruiz-Requies I, et al. COLLAGE: A collaborative Learning Design editor based on patterns. Educational Technology & Society. 2006;9:58–71.

20. Torres J, Resendiz J, Aedo I, Dodero JM. A model-driven development approach for learning design using the LPCEL Editor. Journal of King Saud University - Computer and Information Sciences. 2014;26(1, Supplement):17 – 27. doi:https://doi.org/10.1016/j.jksuci.2013.10.004.

21. Torres J, Dodero JM, Aedo I, Diaz P. Designing the Execution of Learning Activities in Complex Learning Processes Using LPCEL. In: Sixth IEEE International Conference on Advanced Learning Technologies (ICALT'06); 2006. p. 415–419.

22. Sampson D, Karampiperis P, Zervas P. ASK-LDT: A web-based learning scenarios authoring environment based on IMS learning design. Advanced Technology for Learning. 2005;2:207–215. doi:10.2316/Journal.208.2005.4.208-0863.

23. Dodero JM, Ruiz-Rube I, Palomo-Duarte M, Cabot J. Model-Driven Learning Design. Journal of Research and Practice in Information Technology. 2012;44(3):267–288.

24. Martínez-Ortiz I, Sierra JL, Fernández-Manjón B, Fernández-Valmayor A. Language engineering techniques for the development of e-learning applications. Journal of Network and Computer Applications. 2009;32(5):1092 – 1105. doi:https://doi.org/10.1016/j.jnca.2009.02.005.

25. Sierra JL, Fernández-Manjón B, Fernández-Valmayor A. A language-driven approach for the design of interactive applications. Interacting with Computers. 2008;20(1):112 – 127. doi:https://doi.org/10.1016/j.intcom.2007.09.001.

26. Calderón A, Boubeta-Puig J, Ruiz M. MEdit4CEP-Gam: A model-driven approach for user-friendly gamification design, monitoring and code generation in CEP-based systems. Information and Software Technology. 2018;95:238 – 264. doi:https://doi.org/10.1016/j.infsof.2017.11.009.

27. Caro MF, Josyula DP, Jiménez JA, Kennedy CM, Cox MT. A domain-specific visual language for modeling metacognition in intelligent systems. Biologically Inspired Cognitive Architectures. 2015;13:75 – 90. doi:https://doi.org/10.1016/j.bica.2015.06.004.

28. Troya J, Vallecillo A. Specification and simulation of queuing network models using Domain-Specific Languages. Computer Standards & Interfaces. 2014;36(5):863 – 879. doi:https://doi.org/10.1016/j.csi.2014.01.002.

29. Marand EA, Marand EA, Challenger M. DSML4CP: A Domain-specific Modeling Language for Concurrent Programming. Computer Languages, Systems & Structures. 2015;44:319 – 341. doi:https://doi.org/10.1016/j.cl.2015.09.002.

30. Kim CH, Grundy J, Hosking J. A Suite of Visual Languages for Model-driven Development of Statistical Surveys and Services. Journal of Visual Languages and Computing. 2015;26(C):99–125. doi:10.1016/j.jvlc.2014.11.005.

31. Fondement F, Baar T. Making Metamodels Aware of Concrete Syntax. In: Hartman A, Kreische D, editors. Model Driven Architecture – Foundations and Applications. Berlin, Heidelberg: Springer Berlin Heidelberg; 2005. p. 190–204.

32. Blackwell AF, Britton C, Cox A, Green TRG, Gurr C, Kadoda G, et al. Cognitive Dimensions of Notations: Design Tools for Cognitive Technology. In: Beynon M, Nehaniv CL, Dautenhahn K, editors. Cognitive Technology: Instruments of Mind. Berlin, Heidelberg: Springer Berlin Heidelberg; 2001. p. 325–341.

33. Blackwell AF, Green TRG. Notational Systems - the Cognitive Dimensions of Notations framework. In: Carroll JM, editor. HCI Models, Theories, and Frameworks: Toward a Multidisciplinary Science. San Francisco: Morgan Kaufmann; 2003. p. 103–134.

Capítulo 10

Conclusiones

En esta tesis se ha presentado una solución *MDA* que permite desarrollar aplicaciones para el aprendizaje de idiomas a partir de modelos. Desde una capa *Computational Independent Model* (CIM) hasta la capa *Implementation Specific Model* (ISM), se han definido los meta-modelos y se han desarrollado los editores gráficos para los distintos lenguajes específicos de dominio, que permiten modelar todo lo necesario para el desarrollo de metodologías de aprendizaje de idiomas. Por otra parte, también se han definido las transformaciones necesarias –y el proceso de transformación asociado– para la generación automática del código fuente (en HTML y JavaScript) de este tipo de aplicaciones.

Con todo, se han alcanzado los objetivos específicos –y por tanto también el principal– previstos en el planteamiento inicial de esta tesis (ver la Sección 1.3), y se han abordado todas las preguntas de investigación planteadas (ver la Sección 1.2).

Es importante destacar que cuatro publicaciones respaldan esta tesis ([22][23][21][24]), junto con otros artículos relevantes también presentados en detalle en esta memoria (ver los Capítulos que van desde el Capítulo 3 hasta el Capítulo 9). Así pues,

1. Se ha realizado la revisión de las características principales utilizadas en las metodologías de aprendizaje de idiomas. El estudio ha permitido obtener, mediante un proceso de abstracción, los elementos comunes a todas ellas.

2. Se ha realizado también una revisión de los trabajos relacionados con el desarrollo de este tipo de aplicaciones con un enfoque dirigido por modelos, en lo relativo a: meta-modelos, editores de modelos y generación de código fuente.

3. Se ha realizado un estudio sistemático (*Systematic Mapping Study*) de la generación de código utilizando el enfoque de arquitecturas dirigidas por modelos, y se ha llegado a la identificación de 50 estudios primarios a partir de 2.145 artículos relacionados con MDA publicados durante un periodo de 10 años (2008-2018).

4. Se ha definido una arquitectura dirigida por modelos para desarrollar aplicaciones de aprendizaje de idiomas, y se han desarrollado los editores de modelos necesarios para modelar este tipo de aplicaciones siguiendo la arquitectura definida. Así mismo, se han desarrollado las transformaciones *modelo-a-modelo* y *modelo-a-texto* necesarias, que –junto con el adecuado proceso de dichas transformaciones– conduce a la generación automática del código fuente (en HTML y JavaScript) de parte de este tipo de aplicaciones.

5. Se han llevado a cabo casos de estudio relativos al modelado y la generación automática del código fuente de las principales funcionalidades del método Lexiway, así como de algunos tipos de actividades utilizadas en la metodología Busuu. De forma parcial (modelado), se han realizado casos de estudio con los principales tipos de actividades de Duolingo, y algunos de Babbel y Busuu.

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 de idiomas, 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 más mecanismos de actividades de aprendizaje, utilizados de forma semejante en metodologías como Busuu, Babbel y Duolingo. Finalmente, se está trabajando en nuevas extensiones de los meta-modelos propuestos, que permitan mejorar la validación de los modelos mediante la definición de expresiones OCL personalizadas, y que permitan introducir nuevas características de una manera flexible y robusta.

Bibliografía

[1] Bárcena, E. (2015). State of the art of language learning design using mobile technology: sample apps and some critical reflection. In *Proceedings of the 2015 EUROCALL Conference, Padova, Italy (2015)*, pages 36–43.

[2] Bizonova, Z. (2007). Model driven e-learning platform integration. In Maillet, K., Klobucar, T., Gillet, D., and Klamma, R., editors, *Proceedings of the EC-TEL 2007 PROLEARN Doctoral Consortium, Crete, Greece, September 18, 2007*, volume 288 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[3] Blackwell, A. F., Britton, C., Cox, A., Green, T. R. G., Gurr, C., Kadoda, G., Kutar, M. S., Loomes, M., Nehaniv, C. L., Petre, M., Roast, C., Roe, C., Wong, A., and Young, R. M. (2001). Cognitive dimensions of notations: Design tools for cognitive technology. In Beynon, M., Nehaniv, C. L., and Dautenhahn, K., editors, *Cognitive Technology: Instruments of Mind*, pages 325–341, Berlin, Heidelberg. Springer Berlin Heidelberg.

[4] Blackwell, A. F. and Green, T. R. (2003). Notational systems – the cognitive dimensions of notations framework. In Carroll, J. M., editor, *HCI Models, Theories, and Frameworks: Toward a Multidisciplinary Science*, pages 103–134. Morgan Kaufmann, San Francisco.

[5] Blumschein, P., Hung, W., Jonassen, D., and Strobel, J. (2009). *Model-Based Approaches to Learning: Using Systems Models and Simulations to Improve Understanding and Problem Solving in Complex Domains*. Brill | Sense, Leiden, The Netherlands.

[6] Conn, S. and Forrester, L. (2006). Model driven architecture: A research review for information systems educators teaching software development. 4.

[7] Fardoun, H., Simarro, F. M., and López-Jaquero, V. (2009). eLearniXML: Towards a model-based approach for the development of e-learning systems considering quality. *Advances in Engineering Software*, 40(12):1297–1305.

[8] Fardoun, H. M. (2011). *eLearniXML: towards a model-based approach for the development of e-learning systems*. PhD thesis.

[9] Fondement, F. and Baar, T. (2005). Making metamodels aware of concrete syntax. In Hartman, A. and Kreische, D., editors, *Model Driven Architecture – Foundations and Applications*, pages 190–204, Berlin, Heidelberg. Springer Berlin Heidelberg.

[10] García, F. (2008). *Advances in E-Learning: Experiences and Methodologies*. Information Science Reference.

[11] García-Peñalvo, F. J., Sarasa-Cabezuelo, A., and Sierra-Rodríguez, J. L. (2014). Educational software: Case studies and development methods [guest editorial]. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, 9(2):41–42.

[12] ISO-1 (2006). ISO/IEC 25062. SQuaRE - Common Industry Format (CIF) for Usability Test Reports.

[13] ISO-2 (2006). ISO/IEC 25010. Systems and software engineering - Systems and software engineering Quality Requirements and Evaluation (SQuaRE) - Systems and software quality models.

[14] Koch, N. and Kraus, A. (2003). Towards a common metamodel for the development of web applications. In Lovelle, J. M. C., Rodríguez, B. M. G., Gayo, J. E. L., del Puerto Paule Ruiz, M., and Aguilar, L. J., editors, *Web Engineering*, pages 497–506, Berlin, Heidelberg. Springer Berlin Heidelberg.

[15] Landy, L., Tian, Y., and Yang, H. (2008). Mda-based development of music-learning system. In *Proceedings of the 14th Chinese Automation & Computing Society Conference, UK*, Brunel University, West London.

[16] Lanzilotti, R., Ardito, C., Costabile, M. F., and Angeli, A. D. (2006). eLSE methodology: A systematic approach to the e-learning systems evaluation. *Educational Technology & Society*, 9(4):42–53.

[17] Oye, N. D., Salleh, M., and Iahad, N. A. (2012). E-learning methodologies and tools. *International Journal of Advanced Computer Science and Applications(IJACSA)*, 3(2).

[18] Puerta, A. (1997). A model-based interface development environment. 14(4):40–47.

[19] Sanchis, A. and Sebastian, G. (1997). *Creación y mantenimiento de páginas Web con 'Visual HTML' para Windows'95*. Servicio de Publicaciones de la Universidad Politécnica de Valencia, Valencia, Spain.

[20] Sebastian, G., Gallud, J. A., and Tesoriero, R. (2010). A proposal for an interface for service creation in mobile devices based on natural written language. In *Proceedings of the 2010 Fifth International Multi-Conference on Computing in the Global Information Technology*, ICCGI '10, page 232–237, USA. IEEE Computer Society.

[21] Sebastian, G., Gallud, J. A., and Tesoriero, R. (2019a). Code generation using model driven architecture: A systematic mapping study. *Journal of Computer Languages*, page 100935.

[22] Sebastian, G., Tesoriero, R., and Gallud, J. A. (2017). Modeling language-learning applications. *IEEE Latin America Transactions*, 15(9):1771–1776.

[23] Sebastian, G., Tesoriero, R., and Gallud, J. A. (2018). Model-based approach to develop learning exercises in language-learning applications. *IET Software*, 18(3):206–2014.

[24] Sebastian, G., Tesoriero, R., and Gallud, J. A. (2019b). Automatic code generation for language-learning applications. *IEEE Latin America Transactions*.

[25] Sebastian, G., Villanueva, P. G., Tesoriero, R., and Gallud, J. A. (2011). *Multi-touch Collaborative DUI to Create Mobile Services*, pages 145–151. Springer London, London.

[26] Vesselinov, R. and Grego, J. (2012). Duolingo Effectiveness Study: FINAL REPORT. http://static.duolingo.com/s3/DuolingoReport_Final.pdf. Online; accessed 9 January 2020.