Computing and Informatics, Vol. 40, 2021, 249–276, doi: 10.31577/cai_2021_2_249

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, 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, 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, also known as 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 a 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 Figure 4), joining the lines, Drag & Drop images (see the right side of Figure 5), and 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 to software validation difficult to manage.

[1] `https://www.facebook.com/LexiwayLearning`

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.

[2] `https://www.facebook.com/LexiwayLearning`

[3] `https://www.duolingo.com/`

[4] `https://www.babbel.com/`

[5] `https://www.busuu.com/`

[6] `http://www.omg.org/mda/`

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].

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 make 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, 2, 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, 3].

An interesting approach that defines a MDA to develop music learning applications is exposed in [26]. 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

[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`

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 [1]. 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 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 [4], 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 [23]. 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 to create 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 [22]. The 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

[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`

with expressions in OCL[14]. As a 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 "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.

Figure 1 shows the proposed metamodel exposing the packages of metaclasses that represent the language learning methodology common elements. The pack-

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

age 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. Figure 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 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

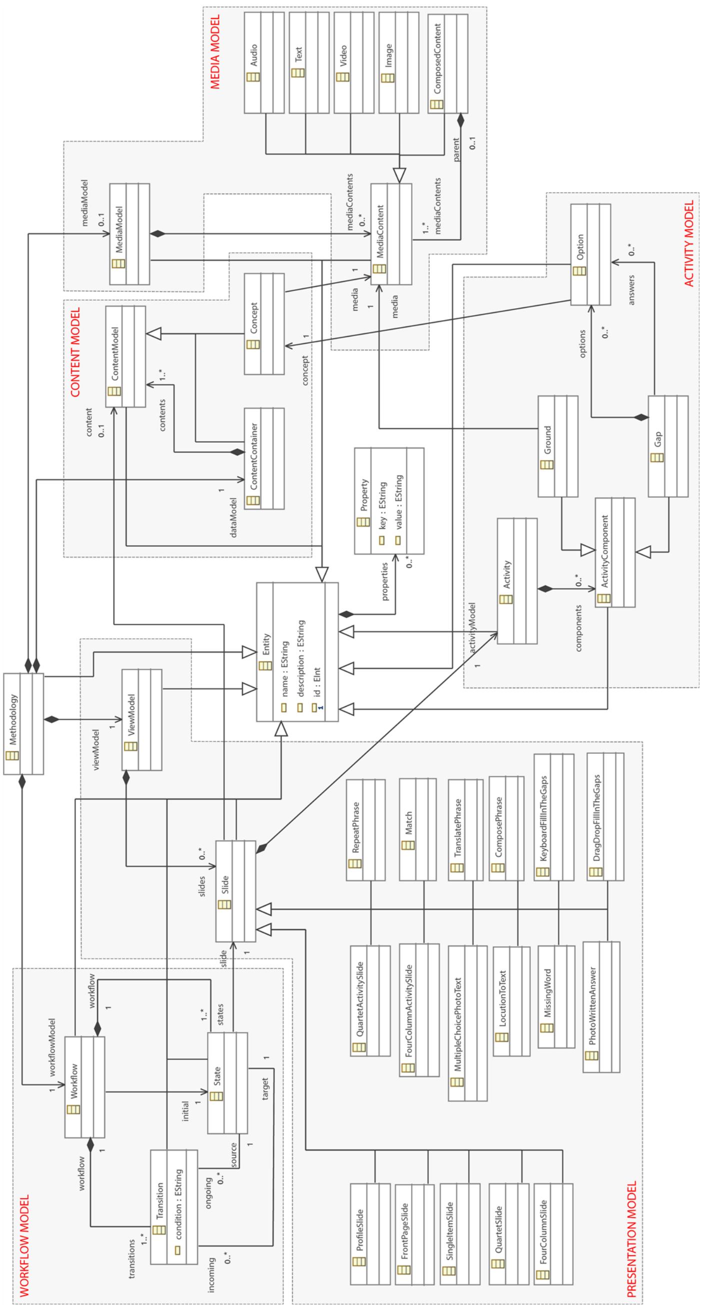


Figure 1. Language learning methodology metamodel

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 define 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.

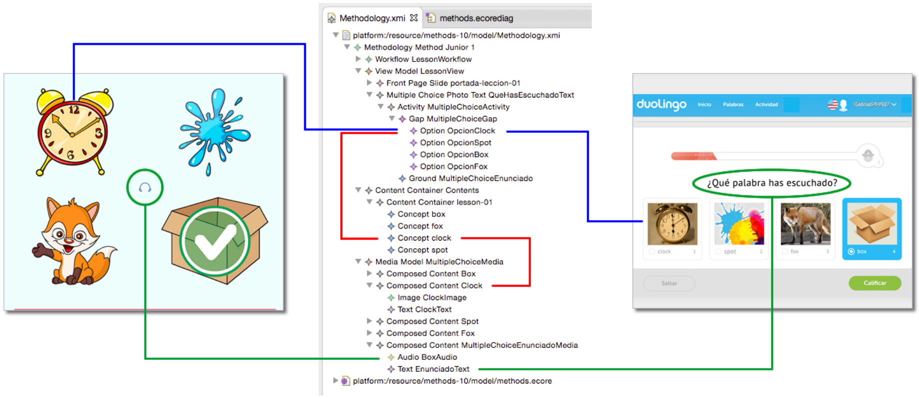
All the metaclasses in this metamodel inherit from the *Entity* metaclass defined in the *Commons* package which provides identification (i.e. *Entity* metaclass) and extension (i.e. *Property* metaclass) features to the rest of the metaclasses. This package is designed to take into account variable 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.

3.1 Analysis of Language Learning Activity Modelling

In [23] we illustrate the flexibility and adaptability of the proposed metamodel to represent different language learning methodologies. Figure 2 shows the correspondence among the different elements of the model of a multiple-choice learning activity Lexiway as well as Duolingo. Thus, Figure 2 illustrates the expressiveness power of

[15] `http://www.eclipse.org/modeling/emf/`
[16] `https://wiki.eclipse.org/OCL/OCLinEcore`



```
Methodology.xmi   methods.ecorediag
▼ platform:/resource/methods-10/model/Methodology.xmi
  ▼ Methodology Method Junior 1
    ▶ Workflow LessonWorkflow
    ▼ View Model LessonView
      ▶ Front Page Slide portada-leccion-01
      ▼ Multiple Choice Photo Text QueHasEscuchadoText
        ▼ Activity MultipleChoiceActivity
          ▼ Gap MultipleChoiceGap
              Option OpcionClock
              Option OpcionSpot
              Option OpcionBox
              Option OpcionFox
            Ground MultipleChoiceEnunciado
    ▼ Content Container Contents
      ▼ Content Container lesson-01
          Concept box
          Concept fox
          Concept clock
          Concept spot
    ▼ Media Model MultipleChoiceMedia
      ▶ Composed Content Box
      ▼ Composed Content Clock
          Image ClockImage
          Text ClockText
      ▶ Composed Content Spot
      ▶ Composed Content Fox
      ▼ Composed Content MultipleChoiceEnunciadoMedia
          Audio BoxAudio
          Text EnunciadoText
▶ platform:/resource/methods-10/model/methods.ecore
```

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

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. Figure 3 depicts the *Activity* package which contains the "Fill-in the Gaps" Activity metamodel used to model language learning activities.

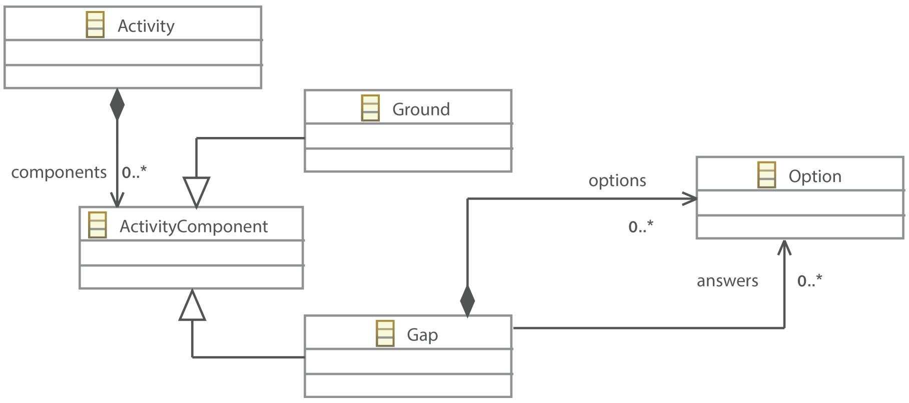
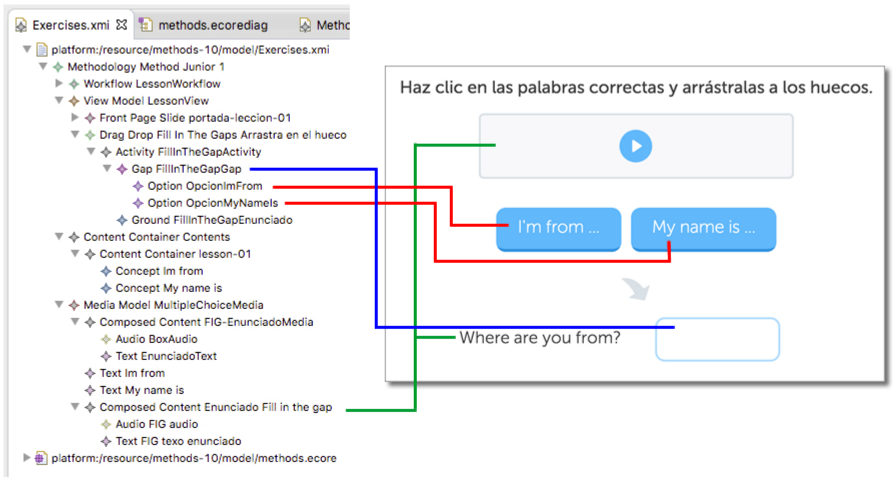


Figure 3. Activity metamodel

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. Figure 4 depicts the model and presentation of a fill-in the gaps activity in the Busuu 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.



```
Exercises.xmi   methods.ecorediag   Metho
▼ platform:/resource/methods-10/model/Exercises.xmi
  ▼ Methodology Method Junior 1
    ▶ Workflow LessonWorkflow
    ▼ View Model LessonView
      ▶ Front Page Slide portada-leccion-01
      ▼ Drag Drop Fill In The Gaps Arrastra en el hueco
        ▼ Activity FillInTheGapActivity
          ▼ Gap FillInTheGapGap
              Option OpcionImFrom
              Option OpcionMyNameIs
            Ground FillInTheGapEnunciado
    ▼ Content Container Contents
      ▼ Content Container lesson-01
          Concept Im from
          Concept My name is
    ▼ Media Model MultipleChoiceMedia
      ▼ Composed Content FIG-EnunciadoMedia
          Audio BoxAudio
          Text EnunciadoText
        Text Im from
        Text My name is
      ▼ Composed Content Enunciado Fill in the gap
          Audio FIG audio
          Text FIG texo enunciado
▶ platform:/resource/methods-10/model/methods.ecore
```

Haz clic en las palabras correctas y arrástralas a los huecos.
I'm from ...
My name is ...
Where are you from?

Figure 4. Fill-in the gaps activity in Busuu

4.2 Word Ordering Learning Activity

The word ordering learning activity asks students to order a set of words to compose a meaningful sentence. Figure 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 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.).

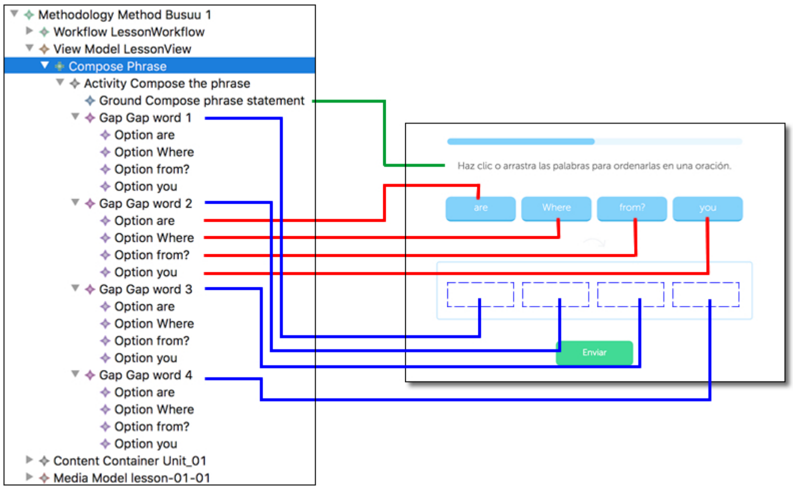


Figure 5. Word ordering learning activity in Busuu

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. Figure 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.

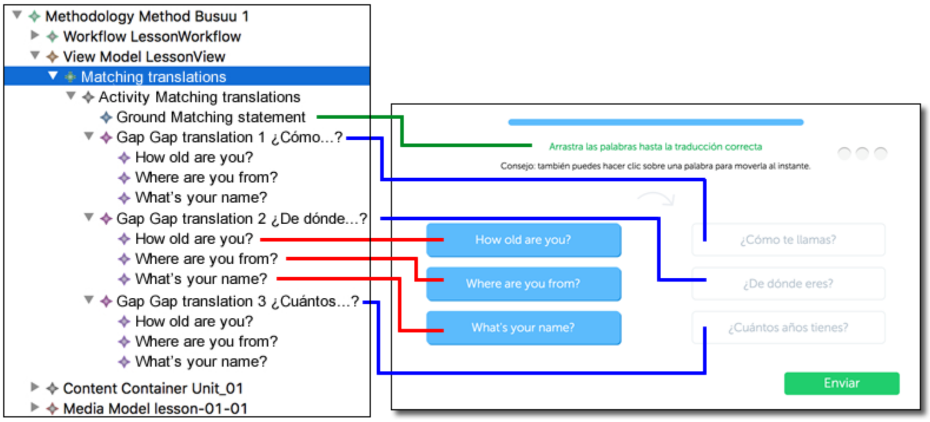


Figure 6. Match-up learning activity in Busuu

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?*, *What's 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.

4.4 Locution to Text, Multiple Choice and Translate Phrase Learning Activities in Duolingo

Figures 7, 8 and 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.

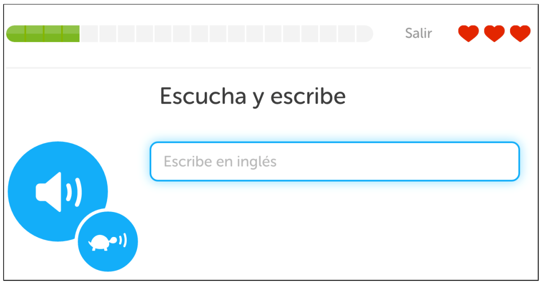
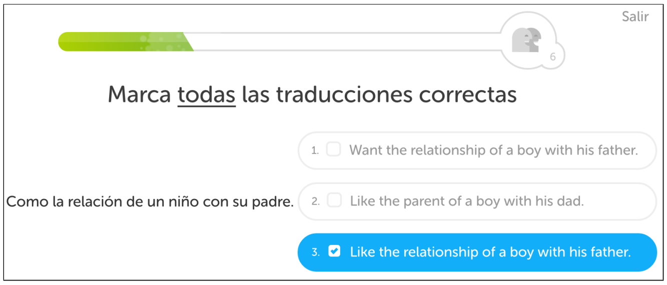


Figure 7. Locution to text learning activity in Duolingo using one gap



Como la relación de un niño con su padre.

1. ☐ Want the relationship of a boy with his father.
2. ☐ Like the parent of a boy with his dad.
3. ☑ Like the relationship of a boy with his father.

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

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

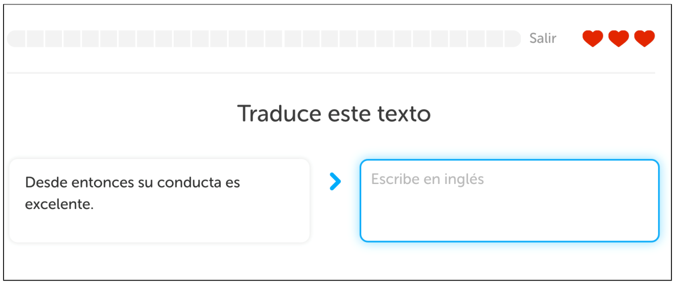


Figure 9. Translate phrase learning activity in Duolingo using one gap

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, Busuu and Duolingo

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

Figure 10, Figure 11 and Figure 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 Figure 2.

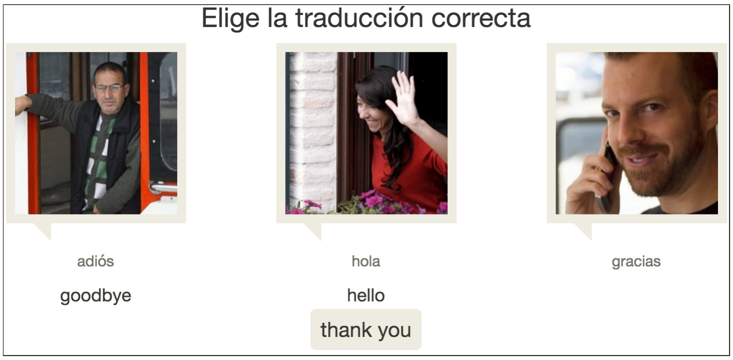


Figure 10. Multiple choice learning activity in Babbel

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 Figure 10 and Figure 12 depict Babbel and Doulingo

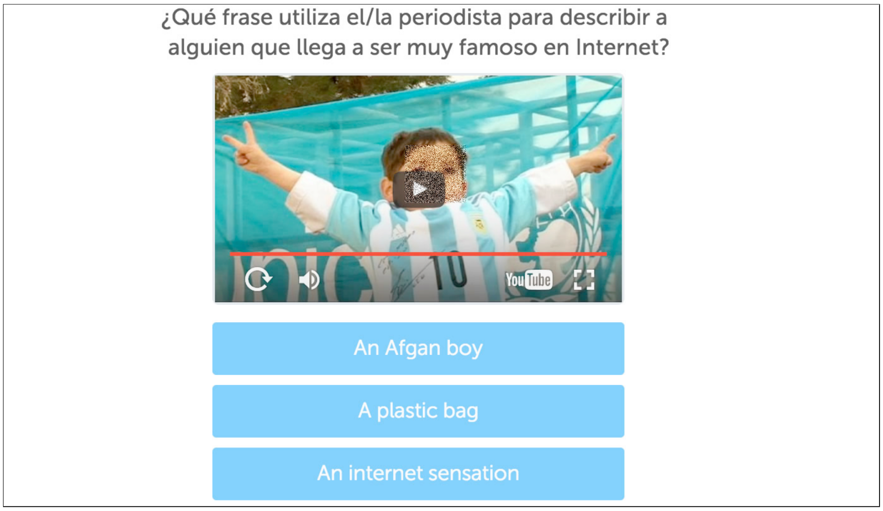

¿Qué frase utiliza el/la periodista para describir a alguien que llega a ser muy famoso en Internet?
An Afgan boy
A plastic bag
An internet sensation

Figure 11. Multiple choice learning activity in Busuu

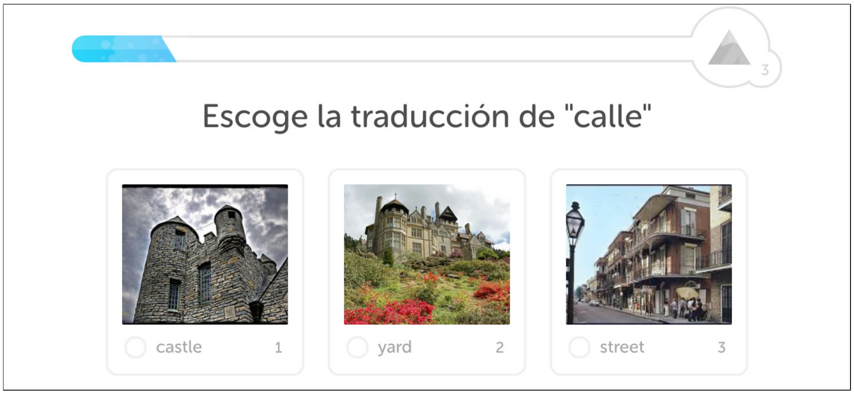


Figure 12. Multiple choice learning activity in Duolingo

version of a multiple choice activity using 3 pictures to represent gap options; the Busuu version of the activity depicted in Figure 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.

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 proposed 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, Ph.D. 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 Figure 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.

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 the left side of Figure 2 generated with the traditional and

T	Description
1	Define the graphic design of the user interface for the learning activity presented on the left side of Figure 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 Figure 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 Figure 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 Figure 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).

Table 1. Common tasks performed by the participants

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 sub-characteristic 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.

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

Table 2. Participant profiles

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 a 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 Figure 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.

[17] `https://validator.w3.org/`
[18] `http://jshint.com/about/`

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	

Table 3. Heuristic evaluation results

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 all evaluated software attributes.

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 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 an 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. an 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).

We 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 definition of customized OCL expressions that are loaded dynamically (using the Complete OCL plugin [19]). These extensions are defined based on the sub-classification of meta-classes, which allow introducing new features in a flexible and robust way.

Moreover, we are addressing the process of model-to-model (M2M) and model-to-text (M2T) transformations required for source code automatic generation (i.e. HTML and JavaScript) of language learning applications. To carry out these transformations, the ATLAS Transformation Language[20] (ATL) and ACCELEO[21] transformation languages are used respectively.

This transformation process requires 3 M2M transformations: (1) a M2M transformation to generate the Activity Model and Workflow Model instances that are part of the PIM layer of the model architecture using a Content Model instance that is part of the CIM layer; (2) another M2M transformation at PIM layer to generate Presentation Model and Media Model instances from Activity Model and Workflow Model instances; finally, (3) the last M2M transformation generates the TagML [25] PSM layer model from Activity Model and Workflow Model PIM layer instances.

In addition, the transformation process also involves 2 M2T transformations. While the first one generates HTML source code from TagML [25] PSM model to define the application UI structure; the second one generates JavaScript source code from Workflow Model PIM layer instance to define the application UI behavior. Thus, the whole process generates the Implementation Specific Model (ISM) of a fully functional application.

Finally, we are also exploring the adaptation of this model architecture to generate a wide variety of interactive multimedia applications including gamification features.

Acknowledgements

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

[19] `https://marketplace.eclipse.org/content/eclipse-ocl`
[20] `https://eclipse.org/atl/`
[21] `https://www.eclipse.org/acceleo/`

the regional government (JCCM) and the European Regional Development Funds (FEDER).

REFERENCES

[1] Bárcena, E.—Read, T.—Underwood, J. et al.: State of the Art of Language Learning Design Using Mobile Technology: Sample Apps and Some Critical Reflection. In: Helm, F., Bradley, L., Guarda, M., Thouësny, S. (Eds.): Critical CALL – Proceedings of the 2015 EUROCALL Conference. Padova, Italy, 2015, pp. 36–43, doi: 10.14705/rpnet.2015.000307.

[2] Bizonova, Z.: Model Driven E-Learning Platform Integration. In: Maillet, K., Klobucar, T., Gillet, D., Klamma, R. (Eds.): Proceedings of the EC-TEL 2007 PROLEARN Doctoral Consortium. CEUR Workshop Proceedings, Vol. 288, 2007.

[3] Blumschein, P.—Hung, W.—Jonassen, D.—Strobel, J. (Eds.): Model-Based Approaches to Learning: Using Systems Models and Simulations to Improve Understanding and Problem Solving in Complex Domains. Brill, Leiden, The Netherlands, Modeling and Simulation for Learning and Instruction, Vol. 4, 2009, doi: 10.1163/9789087907112.

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

[5] Calvary, G.—Coutaz, J.—Thevenin, D.—Limbourg, Q.—Bouillon, L.—Vanderdonckt, J.: A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers, Vol. 15, 2003, No. 3, pp. 289–308, doi: 10.1016/s0953-5438(03)00010-9.

[6] Ceri, S.—Fraternali, P.—Bongio, A.: Web Modeling Language (WebML): A Modeling Language for Designing Web Sites. Computer Networks, Vol. 33, 2000, No. 1-6, pp. 137–157. doi: 10.1016/S1389-1286(00)00040-2.

[7] Conallen, J.: Building Web Applications with UML. 2nd Edition. Addison Wesley, Reading, Massachusetts, October 2002.

[8] Conn, S.—Forrester, L.: Model Driven Architecture: A Research Review for Information Systems Educators Teaching Software Development. Information Systems Education Journal, Vol. 4, 2006, No. 43, pp. 3–11.

[9] Dodero, J. M.—García-Peñalvo, F.-J.—Gonzãlez, C.—Moreno-Ger, P.—Redondo, M.-A.—Sarasa-Cabezuelo, A.—Sierra, J.-L.: Development of E-Learning Solutions: Different Approaches, a Common Mission. IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, Vol. 9, 2014, No. 2, pp. 72–80, doi: 10.1109/RITA.2014.2317532.

[10] Fardoun, H.—Montero, F.—López Jaquero, V.: eLearniXML: Towards a Model-Based Approach for the Development of E-Learning Systems Considering Quality. Advances in Engineering Software, Vol. 40, 2009, No. 12, pp. 1297–1305, doi: 10.1016/j.advengsoft.2009.01.019.

[11] Feuerstack, S.—Pizzolato, E. B.: Engineering Device-Spanning, Multimodal Web Applications Using a Model-Based Design Approach. In: Bressan, G., Sil-

veira, R. M., Munson, E. V., Santanchà, A., da Graça Campos Pimentel, M. (Eds.): Proceedings of the 18th Brazilian Symposium on Multimedia and the Web (WebMedia '12). Association for Computing Machinery, 2012, pp. 29–38, doi: 10.1145/2382636.2382646.

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

[13] García-Peñalvo, F. J.: Advances in E-Learning: Experiences and Methodologies. Information Science Reference, 2008, doi: 10.4018/978-1-59904-756-0.

[14] Isakowitz, T.—Stohr, E. A.—Balasubramanian, P.: RMM: A Methodology for Structured Hypermedia Design. Communications of the ACM, Vol. 38, 1995, No. 8, pp. 34–44, doi: 10.1145/208344.208346.

[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.—Bast, W.: MDA Explained: The Model Driven Architecture: Practice and Promise. Addison-Wesley Professional, 2003.

[17] Koch, N.—Kraus, A.: 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., Aguilar, L. J. (Eds.): Web Engineering (ICWE 2003). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 2722, 2003, pp. 497–506, doi: 10.1007/3-540-45068-8_92.

[18] Lahiani, N.—Bennouar, D.: 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), ACM, 2015, Art. No. 78, doi: 10.1145/2816839.2816850.

[19] Lanzilotti, R.—Ardito, C.—Costabile, M. F.—De Angeli, A.: eLSE Methodology: A Systematic Approach to the E-Learning Systems Evaluation. Educational Technology and Society, Vol. 9, 2006, No. 4, pp. 42–53.

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

[21] Dehbi, R.—Talea, M.—Tragha, A.: A Model Driven Methodology Approach for E-Learning Platform Development. International Journal of Information and Education Technology, Vol. 3, 2013, No. 1, pp. 10–15, doi: 10.7763/IJIET.2013.V3.225.

[22] Sebastián, G.—Tesoriero, R.—Gallud, J. A.: Modeling Language-Learning Applications. IEEE Latin America Transactions, Vol. 15, 2017, No. 9, pp. 1771–1776, doi: 10.1109/TLA.2017.8015084.

[23] Sebastián, G.—Tesoriero, R.—Gallud, J. A.: Model-Based Approach to Develop Learning Exercises in Language-Learning Applications. IET Software, Vol. 12, 2018, No. 3, pp. 206–214, doi: 10.1049/iet-sen.2017.0085.

[24] Tang, S.—Hanneghan, M.: A Model-Driven Framework to Support Development of Serious Games for Game-Based Learning. 2010 Developments in E-Systems Engineering, 2010, pp. 95–100, doi: 10.1109/DeSE.2010.23.

[25] TESORIERO, R.—SEBASTIÁN, G.—GALLUD, J. A.: TagML – An Implementation Specific Model to Generate Tag-Based Documents. Electronics, Vol. 9, 2020, No. 7, Art. No. 1097, doi: 10.3390/electronics9071097.

[26] TIAN, Y.—YANG, H.—LANDY, L.: MDA-Based Development of Music-Learning System. In: Zhang, S., Li, D. (Eds.): Proceedings of the 14th Chinese Automation and Computing Society Conference, 2008, pp. 97–102.

Gabriel Sebastián received his Ph.D. and M.Sc. degrees from the University of Castilla-La Mancha (UCLM) and B.Sc. degree from Polytechnic University of Valencia (UPV) – all the three 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 25 research articles and book chapters in journals and international congresses. Currently, he works as Project Manager in the Interactive Systems Engineering Research Group of the Computing System Department (Faculty of Computing Science Engineering) in UCLM, and he works as Researcher in the Albacete Research Institute of Informatics (I3A) in Albacete, Spain.

Ricardo Tesoriero received his Ph.D. and M.Sc. degrees from the University of Castilla-La Mancha (UCLM), Spain and a B.Sc. 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 Université Catholique 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 is Associate Professor of the Computing System Department at the Faculty of Computer Science Engineering of the UCLM teaching web engineering and services, and human-computer interaction subjects since 2008.

Jose A. Gallud received his Ph.D. degree from the University of Murcia and the M.Sc. and B.Sc. degrees from the Polytechnic University of Valencia – all the three degrees in computer science. His main research of interest focuses on human-computer interaction, development of interactive systems and distributed user interfaces. He 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 Professor at the University of Castilla-La Mancha.