Journal of Theoretical and Applied Information Technology
10th February 2015. Vol.72 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

MODELING AND GENERATING THE USER INTERFACE OF MOBILE DEVICES AND WEB DEVELOPMENT WITH DSL

[1]MOHAMED LACHGAR, [2]ABDELMOUNAÏM ABDALI

[1]Laboratory of Applied Mathematics and Computer Science (LAMAI),
Faculty of Science and Technology (FSTG),
University Cadi Ayyad, Marrakech, Morocco
[1]Laboratory of Applied Mathematics and Computer Science (LAMAI),
Faculty of Science and Technology (FSTG),
University Cadi Ayyad, Marrakech, Morocco
E-mail: [1]lachgar.m@gmail.com, [1]aabdali5@gmail.com

ABSTRACT

Due to the large number and variety of mobile technologies (Android, iOS, Windows Phone, etc) and web (Java Server Faces, Asp.net, HTML 5, etc) based-components, developing the same application for these different platforms becomes a tedious task. The Model Driven Architecture (MDA) approach aims to provide an easy and efficient practical solution for developing a cross-platform application. In this work, we propose a new approach to the design of the user interface for mobile applications and web applications, which we apply to the android platform and Java Server Faces Framework. This approach is later generalized for all mobile platforms and web based-components, by defining a language for the development of graphical interfaces, the Technology Neutral DSL (Domain-specific language) intended to be cross-compiled to generate native code for a diversity of platforms.

Keywords: *Model-driven engineering, Domain-specific language, Cross-Platforms, Code Generation, Templates*

1. INTRODUCTION

Considering the variety of mobiles technologies (Android, IOS, Windows phone etc) and Web (Java server Faces, Asp.net, HTML5 etc) based on the component, developing the same application for these different platforms becomes an exhausting task. In view of the fact that each platform uses different tools, different programming languages and user interface declarations. This heterogeneity development tools and languages makes difficult to develop multi-platform applications. Thus, it requires developers to make a choice on the platform, while ensuring the widest possible dissemination.

The Model Driven Architecture (MDA) [1] approach provides significant advances in term of controlling the development of software applications and allows productivity gains, increased reliability, significant improvement in sustainability and better ways to deal changing constraints.

The MDA [1] approach is highly focused around models. The objective is to switch from mainly documentary models to productive models, by defining a number of operations on these models in order to produce software applications. Such operations include model transformation, the reverse engineering, and the models checking. Its aim is to perform automatic transformations of models until the generation of the code that implements the software. MDA expects to replace the slogan "Write once, Run anywhere" by "Model once, Generate anywhere".

In order to clarify the concepts, the Object Management Group (OMG) has defined a number of terms around the models named meta-meta-model, meta-model, model, business model (CIM) which specifies system requirements of a particular problem domain, functional model (PIM) which is independent from the technique and technical model (PSM) illustrated in "Figure1" and "Figure 2".

Journal of Theoretical and Applied Information Technology
10th February 2015. Vol.72 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

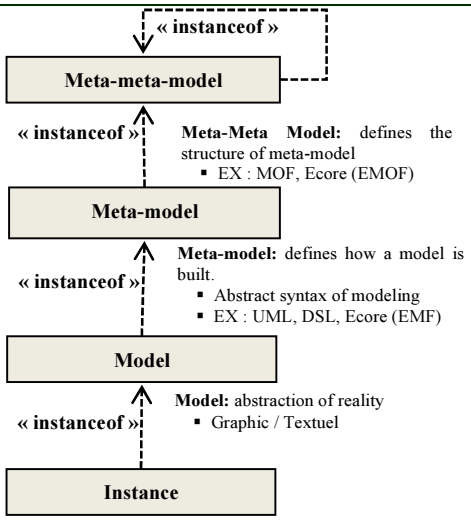


Figure 1: Four-level Meta model Architecture

Thus, the MDA approach allows for the same model to be implemented on multiple platforms through standardized projections. It allows applications to interoperate with models and support the development of new platforms and techniques. The implementation of the MDA is entirely based on the models and their transformations, as illustrated in "Figure 2".

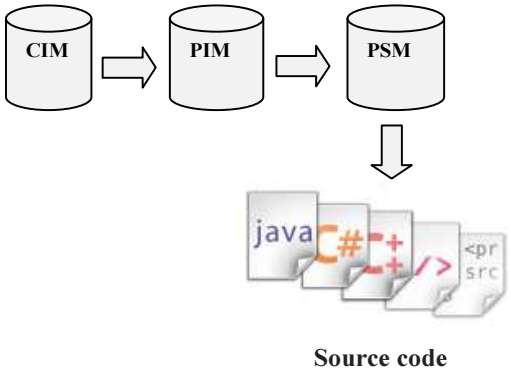


Figure 2 : Key Models in MDA

The transformations between the different models are produced with tools compatible with the OMG standard called QVT (Query / View / Transformation) [2].

The transformation of the entities of the model source involves two steps [3]. The first step to identify the correspondences between the concepts of source and target models in their meta-models, which indicate the existence of a transformation function applicable to all instances of the source meta-model. Next, the second step is to apply the transformation of the source model to automatically generate the target model by a program called transformation engine or execution, "Figure 3" illustrates these two steps of a model transformation.

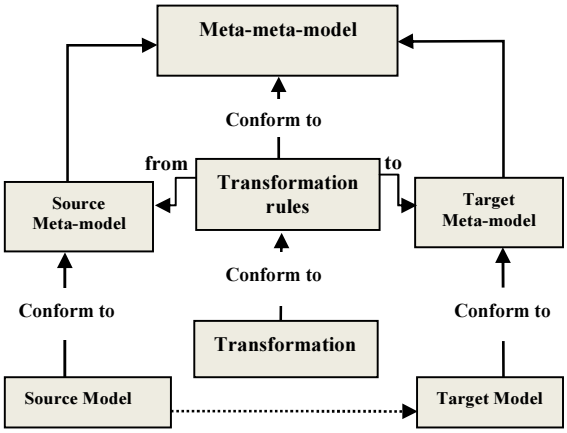


Figure 3: Transformation Process in the MDA Approach

In this work, we propose a new approach for the design of the user interface for mobile applications (using the example of Android) and web applications (using the example of Java Server Faces). This approach is later generalized for all mobile platforms and web based-components, by defining a fully share language for the development of graphical interfaces, the Technology Neutral DSL (Domain-specific language) intended to be cross-compiled to produce native GUI code for a variety of platforms.

This paper is organized as follows. The first section presents some related work. The approach to the development of graphical interface is described in the second part. The third section shows the applicability of our approach through a case study. The last section concludes the paper and presents future work.

2. RELATED WORK

The MDA approach has proven itself for the development of enterprise applications and can also bring a lot for mobile applications. The MDA approach can help us ensure the sustainability of expertise, and gain productivity while addressing the issues of fragmentation of mobile platforms. In recent years, several studies have been done in this direction. Juliano de Almeida Monte Mor et al [4] contributed to the improvement of the generation of graphical user interfaces for a variety of platforms such as JSF and JSTL by using the AndroMDA

Journal of Theoretical and Applied Information Technology
10th February 2015. Vol.72 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

open source Framework. The approach described in their paper is based on the analysis and design of the PIM model using UML diagrams, and then enriched by stereotypes to obtain the model PSM, once the PIM to PSM transformation is achieved. Then, AndroMDA generates the specific code of the target platform. Other research works have already focused on the subject. Stefan Link et al [5] propose UML profiles for the modelling of user interfaces. These researchers exploit the MDA approach to define a model independent from the platform, then enriched by stereotypes in order to obtain a specific model to a platform by using M2M transformations. The latter model is transformed to source code further to M2C (Model to code) transformations. In order to do this, the authors use UML diagrams to define the PIM and the QVT to realize the various transformations. In our paper, we present a new approach to the design of the user interface of mobile applications. We use the MDA approach to provide a platform independent model under textual format and the M2M M2T transformations are applied to generate the GUI for a specific platform. In order to do this, we use Xtext to define a DSL and Xtend 2 to perform different transformations.

An MDA approach [6] has been implemented in order to model and generate graphical interfaces mobile platforms. This approach consists of three main steps:

- Modelling of the graphical interface under UML,
- Transformation of diagrams obtained in a simple plan XML by using JDOM API,
- Generation of the graphical interface on the basis of the approach MDA.

This method presents the advantage of automatically generating graphical interfaces for several mobile platforms from a UML model. Our approach is based on a textual model allowing a simplification of the way models are represented, and generation of graphical interfaces for platforms based on components.

A study of a variety of meta-model of mobile platforms such as Symbian, JavaME and .NET platforms and contribution to the improvement of the common Meta model for these mobile platforms had been investigated by Madari et al [7]. Moreover, projection towards the definition of an abstract syntax of the target platforms is also discussed in this work.

In [8] a Web DSL for integration and data validation that allows unified syntax, error handling mechanisms, and semantics for data validation checks. This enables the Web application developers to adopt a model-driven, focusing on the logical design of an application rather than the accidental complexity of technical implementation approach at low altitude. Thus, our approach is relatively simple and has taken into account the web and mobile application.

3. DESIGN OF THE GRAPHICAL USER INTERFACE FOR AN APPLICATION BASED ON COMPONENTS

To cover the structural aspect, we propose a DSL to describe and conceive the graphic user interfaces of the applications based on components. Our approach proposes a meta-model GUI to design a model of the graphic user interface of an application based on components. Then, M2M (Model To Model) and M2T (Model to Text) transformations are applied to generate the code of the graphic user interface targeting a specific platform the "Figure 4" presents the various stages which characterize our method.

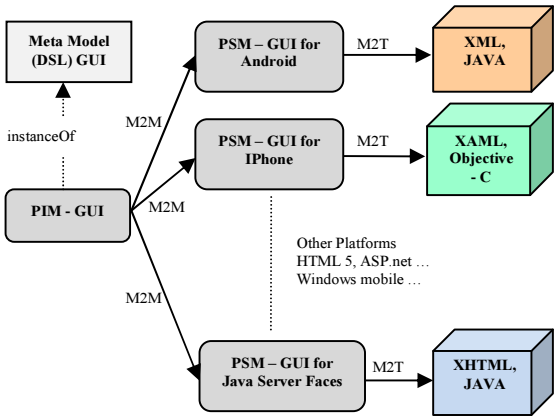


Figure 4: Architecture Proposed for the Generation of Graphical User Interfaces from a Platform Independent Model (PIM-GUI)

3.1 Meta-Model GUI DSL

Our Meta-model is described by the following class diagram:

Journal of Theoretical and Applied Information Technology
10th February 2015. Vol.72 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

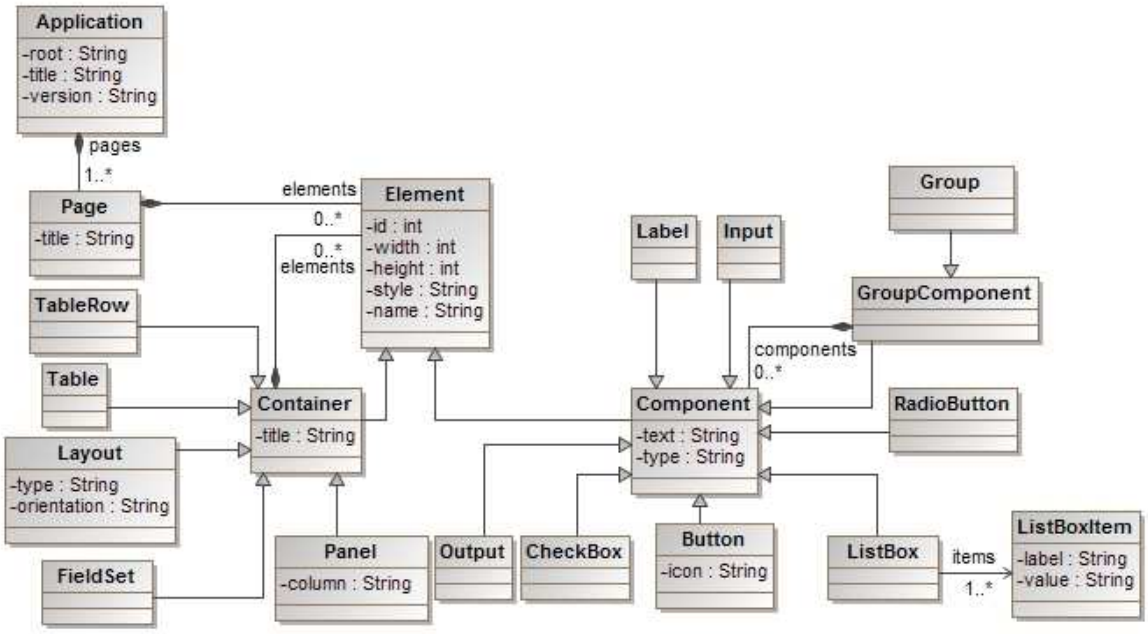


Figure 5 : Proposed Meta-Model GUI DSL

3.2 GUI DSL

Keep the code "clean" is important for software development (and that includes modularity, readability and maintainability). Xtext [9] offers many features to keep DSL clean and modular, thanks to its decomposition into many diverse aspects. Xtext is the solution envisaged for the creation of our DSL.

3.2.1 Description

An application is composed of several pages and each page contains a collection of elements. An element can represent groups of elements or a single component. The type attribute is able to distinguish the type of elements (date, email, text, image, etc) during the transformation.
A page is a collection of graphic components, which can be buttons, text, but also groupings of the other graphic components, for which we can define common attributes (size, positioning, etc.).

An extract of our DSL is represented below.

```
Application:
        'Application'
        'root' root=STRING
        'title' title=STRING
        'version' version=STRING
        NEXT
        pages+=Page*;
Page:
        'Page'
        'title' titre=STRING
        OPEN
        (elements+=Element*)
        CLOSE;
Element:
        Container | Component;
Container:
        FieldSet | Table | Panel;

Component:
        Input | Label | RadioButton | CheckBox |
GroupComponent |
        ListBox;
FiledSet:
        OPENATT 'filedset'
        'legend' title=STRING
        attribut=Attribut CLOSEATT
        OPEN
        (elements+=Element (NEXT elements+=Element)*)?
        CLOSE;
Panel:
        'panel'
        OPENATT attribut=Attribut
        ('header' header=STRING)?
        ('columns' column=STRING)? CLOSEATT
        OPEN
        (elements+=Element (NEXT elements+=Element)*)?
        CLOSE;
```

Journal of Theoretical and Applied Information Technology
10th February 2015. Vol.72 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
Input:
        'input'
        OPENATT attribut=Attribut CLOSEATT;
Label:
        'label'
        OPENATT attribut=Attribut CLOSEATT;
RadioButton:
        'radio'
        OPENATT attribut=Attribut CLOSEATT;
CheckBox:
        'checkBox'
        OPENATT attribut=Attribut CLOSEATT;
Button:
        'button'
        OPENATT attribut=Attribut CLOSEATT;
Groupe:
        'groupe'
        OPENATT attribut=Attribut CLOSEATT
        OPEN
        (components+=Component (NEXT components+=Component)*)?
        CLOSE;
ListBoxItem:
        'item'
        OPENATT
        'label' label=STRING
        'value' value=STRING
        CLOSEATT;
ListBox:
        'list'
        OPENATT attribut=Attribut CLOSEATT
        OPEN
        (items+=ListBoxItem (NEXT items+=ListBoxItem)*)?
        CLOSE;
```

3.2.2 Model validation

Most controls for DSL is usually implemented, according to the semantics which we wish that our DSL respects. These additional controls are implemented using the Xtend 2 class that Xtext has generated. The validation takes place in the background while the user of the DSL is typing in the editor, so that an immediate feedback is provided. An example is illustrated in the following figure:

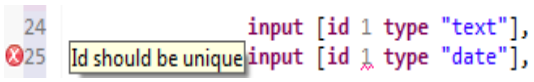


Figure 6: Model validation

3.2.3 Transformation Model to Model (M2M)

The transformation from the PIM towards the PSM is realized with Xtend 2 [10]. For every element of our PIM which associated with one or several elements of the PSM, often this distinction is assured by the type attribute. Some transformations are represented in following table:

PIM	PSM Android
Label	TextView
Input ▪ type = text ▪ type = date ▪ type = multiline ▪ type = email	EditText ▪ inputType = text ▪ inputType = date ▪ inputType = textMultiLine ▪ inputType = textEmailAddress
Output ▪ type = text	TextView
Button ▪ type = image ▪ type = simple	ImageButton Button
CheckBox	CheckBox
RadioButton	RadioButton
Group ▪ type = radio	RadioGroupe
ListBox	Spinner
TableRow	TableRow

Table 1: Some Transformation Rule for Android Platform

Table 2: Some Transformation Rule for Java Server Faces

PIM	PSM JSF
Label	outPutLabel
Input ▪ type = text ▪ type = date ▪ type = multiline ▪ type = email	inputText calendar inputTextarea inputText[1]
Output ▪ type = text	outputText
Button ▪ type = image ▪ type = simple	commandButton[2] commandButton
CheckBox	selectBooleanCheckbox
RadioButton	selectItem
Group ▪ type = radio	selectOneRadio
ListBox	selectOneMenu
TableRow	columns

3.2.4 Projection in the templates

When the transformation M2M is realized. A 2nd stage which consists in generating the code from the obtained PSM, what we call the projection. The

[1] inputText with validator

[2] commandButton with icon attribute

Journal of Theoretical and Applied Information Technology
10th February 2015. Vol.72 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Template of targets pages is developed with Xtend 2. A part of template is represented below.

```
override doGenerate(Resource input, IFileSystemAccess fsa) {
    var application = input.contents.head as Application
    pages = application.pages
    title = application.title
    for (Page page : pages) {
        id = pages.indexOf(page) + 1
        elements = page.elements
        titlePage = page.title
        for (Element e : elements) {
            fsa.generateFile (application.root + "android/" + titlePage +
            ".xml", e.compile)
            fsa.generateFile (application.root + "jsf/" + titlePage +
            ".xhtml",    e.compileJsf)
        }
    }
    fsa.generateFile (application.root + 'values/string' + ".xml", gen)
}
def gen()'''
    <?xml version="1.0" encoding="utf-8"?>
    <resources>
      <string name="app_name">«titre»</string>
      <string name="action_settings">Settings</string>
      «FOR m : pages»
          «var int id = pages.indexOf(m) + 1»
          «FOR v : m.elements»
              «genString(v, id)»
          «ENDFOR»
      «ENDFOR»
    </resources>
'''
```

3.2.5 Synthesis

To sum up, our approach is based on four main steps:

- Analysis and modeling of graphical interfaces with a textual model in compliance with our DSL,
- Validation of model by means of Xtend 2,
- Transformation PIM to PSM by means of Xtend 2,
- Generation of code by projecting in the Templates of the target platforms.

In the following, we present the diagrams for the generation of the graphic interfaces for android platfom and Java Server Faces (JSF) Framework.

a) Android Platform

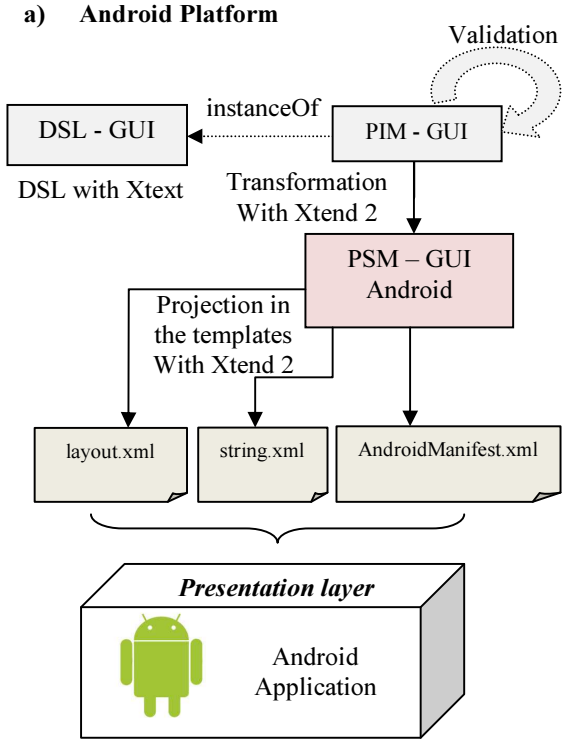


Figure 7: Diagram of Transformation for Android Platform

b) Java Server Faces Framework

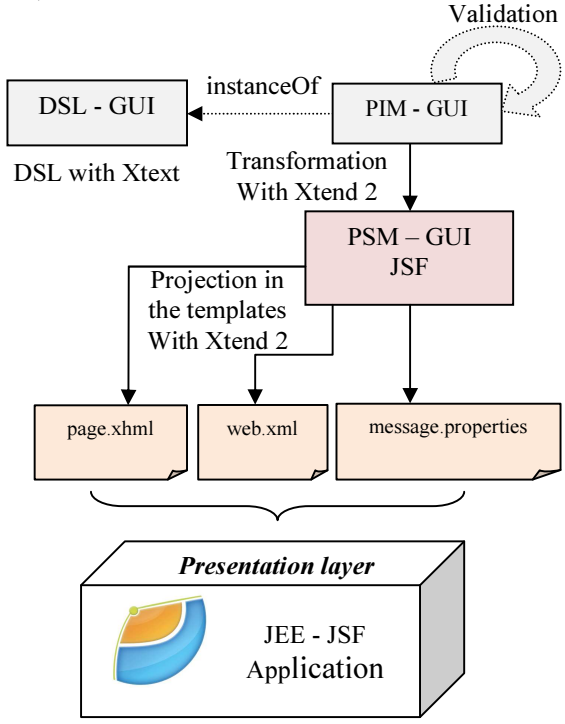


Figure 8: Diagram of Transformation for JSF

Journal of Theoretical and Applied Information Technology
10th February 2015. Vol.72 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4. CASE STUDY: ENGLISH GRAMMAR QUIZ APPLICATION

In this section, we present an example that allows us to illustrate our approach which automatically generates the graphical interface for an application Android and Java Server Faces. We consider a test application with multiple choices named English Grammar quiz (EGQ), is an easy way to learn English Grammar with in quiz as games.

The application contains a huge number of questions in English grammar and vocabulary that are selected randomly to improve your English which are presented in different windows. The application automatically grades the quiz, and provides immediate feedback to the user.

4.1 Analysis and model creation

The objective of this stage is to design the various graphic interfaces for our application by using a model corresponding to our DSL, illustrated in "Figure 9".

```
Application root "C:\\" titre "English Grammar Quiz" version "V 1.0",

Page title "page_1" {
   panel [ id 1 type "simple" orientation 'horizontal' header "Question 1 from 2" ] {
        panel [ id 2 type "grid" orientation 'vertical'  columns "1"] {
           label [ id 1  text 'I_____a new car last month. ' ] ,
           groupe [ id 2 type "radio"  orientation "vertical" ] {
              radio [ id 1  text 'buyed' ] ,
              radio [ id 2  text 'bought' ],
              radio [ id 3  text 'have bought' ],
              radio [ id 4  text 'did bought' ]
           } ,
           panel [ id 3 type "simple" orientation 'horizontal' columns "2" ] {
             button [ id 1  text 'Next Question >>' ]
           }
        }
   }
}


Page title "page_2" {
   panel [ id 1 type "simple" orientation 'horizontal' header "Question 2 from 2" ] {
        panel [ id 2 type "grid" orientation 'vertical'  columns "1"] {
           label [ id 1  text '_____ students attended the meting  ? ' ],
           groupe [ id 2 type "check" orientation "vertical" ] {
              checkBox [ id 1  text 'How many' ] ,
              checkBox [ id 2  text 'How much' ],
              checkBox [ id 3  text 'How long' ],
              checkBox [ id 4  text 'How was' ]
           } ,
           panel [ id 3 type "simple" orientation 'horizontal' columns "2" ] {
             button [ id 1  text 'Submit' ]
           }
        }
   }
}
```

Figure 9: Model of the English Grammar Quiz

In this case, our application is constituted by two pages named "page_1" and "page_2" which contain a collection of elements.

4.2 GUI generated targeting the Android platform

We can automatically generate the source files in an Android project from a model consistent with our DSL. The "Figure 10" illustrates the result obtained.

Screen_1

Screen_2

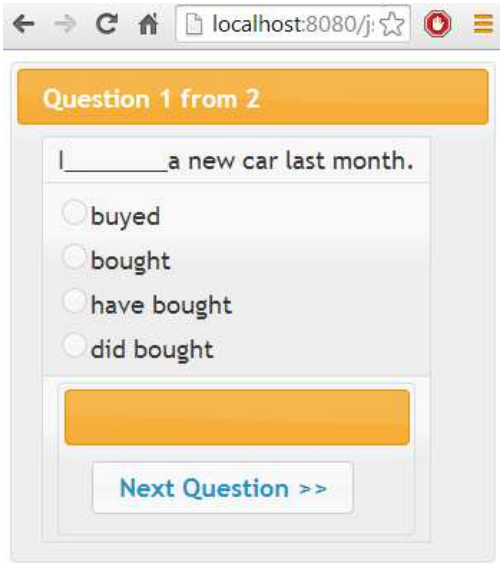
Figure 10: EGQ with Android

Journal of Theoretical and Applied Information Technology
10th February 2015. Vol.72 No.1

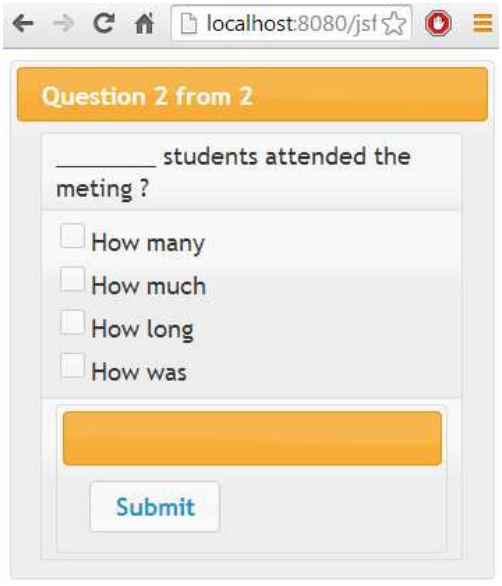
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4.3 GUI generated targeting the Java Server Faces Framework

We can automatically generate the source files into a Java Enterprise Edition Web project from a model consistent with our DSL. It develops a quality application at effective time without technical concerns. The "Figure 11" shows the result obtained.



Page_1



Page_2

Figure 11: EGQ with JSF

5. CONCLUSION AND FUTURE WORK

Due to the large number and variety of mobile technologies and web based-components, developing the same application for these different platforms becomes a tedious task. In view of the fact that each platform uses diverse tools, programming languages and user interface declarations. This heterogeneity development tools and languages makes hard to develop multi-platform applications. Thus, it requires developers to make a choice on the platform, while ensuring the widest possible dissemination.

In this paper we presented a study of the mobile Android platform and the Java Server Faces Framework, and we have adopted the MDA approach to automatically generate the Graphical User Interface (GUI code) for the two platforms by defining a new GUI DSL. This approach is later generalized for all mobile platforms (iPhone, Windows mobile etc) and web (Asp.net, HTML5 etc) based-components.

The potential benefits of the MDA comes from the cost reduction in having only one code to write and maintain, and the time reduction being able to write one code and target multiple devices and platforms, making researching cross-platform applications development with one's effort and findings. Our work falls into this category of research that aims to automate GUI code generation for cross-platform applications from a textual model in accordance with our DSL. This approach is based on four main steps:

- Analysis and modeling of graphical interfaces with a textual model in compliance with our DSL,
- Validation of model by means of Xtend 2,
- Transformation PIM-GUI to PSM by means of Xtend 2,
- Generation of GUI code by projecting in the the adequate Templates of the target platforms.

For future studies we intend to complete this code generator to generate source code, such as the structure of classes, the management of transitions between the screens of the GUI and system events, avoiding the developer to write repetitive code.

Furthermore, we will be going to work on the support of all the platforms by analyzing and defining the characteristics of each one.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

REFRENCES:

[1] Object Management Group, "MDA Guide Version 1.0.1", omg/2003-06-01, June 2003.

[2] Object Management Group, "Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification", Version 1.1, January 2011, http://www.omg.org/spec/QVT/1.1/.

[3] S. Diaw, R. Lbath, and B. Coulette, "Etat de l'art sur le développement logiciel basé sur les transformations de modèles". TSI, Hermès Sciences Publications, Vol. 29, N. 4-5/2010, p. 505-536, juin 2010.

[4] J. A. Monte-Mor, E. O. Ferreira, H. F. Campos, A. M. da Cunha, and L. A. V. Dias, "Applying MDA Approach to Create Graphical User Interfaces", *Eighth International Conference on Information Technology*: New Generations, Las Vegas, NV, IEEE, 11-13 April 2011, p.p. 766-771.

[5] S. Link, T. Schuster, P. Hoyer, and S. Abeck, "Focusing Graphical User Interfaces in Model-Driven Software Development", First International Conference on Advances in Computer-Human Interaction", Sainte Luce, 10-15 Feb. 2008, pp. 3-8.

[6] A. Sabraoui, M.E. Koutbi and I. Khriss, "GUI Code Generation for Android Applications Using a MDA Approach", Mobile Intell. Syst. Team (MIS), Ecole Nat. Super. d'Inf. et d'Anal. Des Syst. (ENSIAS), Rabat, Morocco, IEEE, 5-6 Nov. 2012.

[7] I. Madari, L. Lengyel, and T. Levendovszky, "Modeling the User Interface of Mobile Devices with DSLs", *8th International Symposium of Hungarian Researchers on Computational Intelligence and Informatics*, Budapest, Hungary, November 15-17 2007, pp. 583-589.

[8] D. M. Groenewegen and E. Visser, "Integration of Data Validation and User Interface Concerns in a DSL for Web Applications", In Mark G. J. van den Brand, Jeff Gray, editors, *Software Language Engineering*, *Second International Conference, SLE 2009*, Denver, USA, October, 2009, Lecture Notes in Computer Science, Springer, 2009, pp 164-173.

[9] Eclipse, Xtext "LANGUAGE DEVELOPMENT MADE EASY", https://www.eclipse.org/Xtext/.

[10] Eclipse, Xtend "JAVA 10, TODAY", https://www.eclipse.org/xtend/.