Journal of Software Engineering and Applications, 2014, 7, 906-919
Published Online October 2014 in SciRes. http://www.scirp.org/journal/jsea
http://dx.doi.org/10.4236/jsea.2014.711081

Scientific Research

Evaluation of XIS-Mobile, a Domain Specific Language for Mobile Application Development

André Ribeiro, Alberto Rodrigues da Silva

INESC-ID/Instituto Superior Técnico, Lisbon, Portugal
Email: andre.ribeiro@tecnico.ulisboa.pt, alberto.silva@tecnico.ulisboa.pt

Received 15 August 2014; revised 10 September 2014; accepted 3 October 2014

Abstract

The great evolution of the mobile market during the last years caused some fragmentation of the mobile platforms namely through the existence of different programming languages and software development tools for each platform. This fact can be an obstacle and increases the development complexity and costs when we want to develop mobile applications for multiple platforms. The XIS-Mobile domain specific language (defined as a UML profile) and its MDD-based framework address this problem by proposing platform-independent models to describe mobile applications and from them automatically generate the application's source code. Many issues arose during an iterative process of evaluation which originated changes and the evolution of XIS-Mobile. This paper presents the results of the evaluation of XIS-Mobile (both the language and the companion framework) obtained through the implementation of a case study and by conducting a user session, and discusses its benefits and challenges.

Keywords

Model-Driven Development, Cross-Platform, Mobile Applications, Domain Specific Languages

1. Introduction

The mobile computing industry has suffered a great evolution, during the last decade, with the emergence of a great variety of mobile devices, as well as the operating systems that support them [1]. Due to this fact, mobile devices and applications are becoming more present than ever in our lives facilitating our daily tasks. All of this resulted primarily from the efforts made by the major mobile industry companies (e.g. Google, Apple or Microsoft) to popularize their respective products [2]. However, a certain fragmentation of the mobile platforms has

How to cite this paper: Ribeiro, A. and da Silva, A.R. (2014) Evaluation of XIS-Mobile, a Domain Specific Language for Mobile Application Development. *Journal of Software Engineering and Applications*, **7**, 906-919.
http://dx.doi.org/10.4236/jsea.2014.711081

appeared, since each company provides its own platform with specific language, tools and application market. This fragmentation can be an obstacle particularly when someone wants to develop a mobile application for multiple platforms. Moreover, due to the differences between each platform, the development complexity and consequently the costs of producing a cross-platform mobile application have increased, because simply rewriting a mobile application for each target mobile platform is often impractical due to budget and time limitations. Thus, given these problems, the ideal scenario would be to develop the application once and then run it in as many platforms as desired. Fortunately, some work has been conducted over the last years to tackle both problems presented previously: the software development complexity and the mobile platform fragmentation. The use of web technologies (e.g. HTML5, CSS3 and JavaScript libraries), cross-platform tools and frameworks, or approaches based on Model-Driven Development (MDD) [3] [4], like the one presented in this paper, are examples of solutions focused on solving these problems.

XIS-Mobile [5] has been created precisely with the goal of mitigating these problems. It materializes an idea for the extension of the XIS UML profile [6] [7], used to model desktop or web interactive applications and generate code for multiple platforms (e.g. Windows Forms.NET and Microsoft ASP.NET). However, XIS reveals several limitations to model mobile applications, like proper support for the specification of gestures, internet connection, localization and other context-aware issues common in mobile applications. Furthermore, from what have been researched and our knowledge, the smart approach proposed by XIS was never actually implemented, while in XIS-Mobile is implemented, and the support of the XIS language was achieved by a proprietary tool that has not been maintained. XIS-Mobile proposes the creation of platform-independent models (PIM) to design mobile applications, using a UML profile, and from them generate native source code for multiple mobile platforms through model-to-model and model-to-text transformations. It is important to emphasize that XIS-Mobile does not intend to replace the role of a developer, but instead represent a helpful tool that generates the skeleton of mobile applications. Namely, XIS-Mobile helps the developer by generating the repetitive and boilerplate code which represents a great percentage of the application's code. Then, the developer can modify the generated code if it does not fulfill all his needs. For instance, he could need to improve the GUI or implement some custom actions attached to certain widgets. XIS-Mobile is suitable for a multitude of mobile apps that are typically business (and form) oriented, e.g., that allow consulting and submitting data; but are not intended for very specific domains such as games. XIS-Mobile has been developed over the last eighteen months following the well-known Action Research Methodology [8]. Over time, it was necessary to evaluate the usefulness and adequacy of XIS-Mobile to the purpose of modeling mobile applications. This evaluation process was done in an iterative and gradual way. Initially, this was done testing subsets of the language, checking the corresponding models and code generated, and then successively adding new language concepts and respective functionalities. Throughout this period, some case study applications were defined and used to analyze the full development process. Furthermore, it was also developed a first version of a code generator for iOS. From this iterative evaluation, several issues and needs were identified and justified the need of a new XIS-Mobile version, more clean and up to date when compared to the one presented in [5].

The outline of this paper is as follows. Section 2 discusses the related work. Section 3 describes the case study application used throughout the paper to explain the XIS-Mobile language and also used in the session with users. Section 4 presents the XIS-Mobile language, namely its multi-view organization, supported design approaches and main concepts. Section 5 describes the MDD-based framework that supports the XIS-Mobile language focusing on the usual development process and the technologies used. Section 6 discusses the preliminary results obtained from the creation of the user session. Finally, Section 7 concludes the paper, summarizing its key points and giving directions for future work.

2. Related Work

Similarly to XIS-Mobile, some works propose and justify the relevance of using UML profiles and MDD to abstract the application development, and to achieve platform independency. Unfortunately, none of them represent a real alternative to XIS-Mobile. For example [9] is more focused on mobile context-aware applications, while MAM-UML [10] targets mobile-agent applications addressing concepts like code mobility.

Phone Gap [11] or Appcelerator Titanium [12] are examples of cross-platform tools for mobile application development, which have appeared in the last years and widely used to overcome the mobile platform fragmentation. Surveys like the ones presented in [13]-[15] evaluate several of these tools. Mostly, this kind of tools

does not reduce the complexity of implementation and in some cases does not produce truly native applications.

MobiCloud [16] is a textual DSL purposely created to generate mobile applications leveraging the Cloud computing paradigm. Besides supporting the automatic generation of mobile applications, it also creates applications that will function as back-end, through the use of Cloud Computing platforms (e.g. Amazon EC2 and Google App Engine). MobiCloud is based on the Model-View-Controller (MVC) design pattern and so has as main components: models, views and controllers. Despite being a textual DSL, MobiCloud is complemented with a tool called MobiCloud Composer that enables the generation of MobiCloud scripts using graphical components. These components can be dragged-and-dropped and interconnected to create the desired configuration. Its Ruby-based syntax represents a shortcoming, because only allows limited constructs. This fact results in generic applications with restrictions in the UI customization and also a limited set of actions supported (only CRUD). While XIS-Mobile only generates code for mobile applications, MobiCloud also generates back-end applications and takes advantage of Cloud Computing.

Unlike XIS-Mobile, MobDSL [17] proposes to achieve portability through the use of a virtual machine (VM) and thus, does not generate native source code, instead interprets the code through the VM. This can represent a drawback, namely on Apple devices, since MobDSL requires the VM to be installed on the device. Moreover, because of being new and textual, it can be harder to develop a mobile application using MobDSL than with XIS-Mobile.

MobiA (Mobile Applications modeler), similarly to XIS-Mobile, proposes a MDD approach to decrease the complexity of developing mobile applications and suggests the use of a high-level model to achieve platform independence. Other points are the use of a visual editor to design the system and the existence of multiple views, like a navigation model and a screen description model, equivalents to XIS-Mobile Navigation Space and Interaction Space views, respectively. On the other hand, MobiA does not use a standard language like UML, but a specific one based on XML. This fact could be a disadvantage not just for introducing less flexibility and expressiveness than a UML-based language, but also, for example, in the rigor of the documentation. In addition, MobiA states is focus on the development of mobile health monitoring applications for non-expert users [18].

A lot of work has been done based on the Cameleon Reference Framework [19], being UsiXML4ALL [20] or MARIA [21] some examples. UsiXML4ALL acts as a UI renderer for multiple platforms and connects the UI to application logic code, but requires the manual development of all the business logic code, while XIS-Mobile mainly through its concept of XisAction can generate a considerable part of it. MARIA focuses on service-oriented applications in ubiquitous environments. Like XIS-Mobile, MARIA takes into account notions like gestures detection and web service call, but it goes further in ubiquitous environments exploitation, since it supports the migration of UIs from devices (either desktop or mobile) by maintaining their state while the user is moving. Additionally, other initiatives, namely the Google App Inventor [22], highly abstracted the mobile application development through the use of building blocks that doesn't require the user to write code and proved its usefulness in introductory programming courses. However, Google App Inventor only supports the generation for Android and has been discontinued by Google and its support was transferred to the MIT Center for Mobile Learning.

3. Case Study—The Flight Reservation App

A case study application named "Flight Reservation App" was defined in order to better illustrate and explain the XIS-Mobile language concepts and views throughout Section 4. Additionally, this application was used in the user session, described in Section 6, where the participants had the task of modeling this application using XIS-Mobile.

The Flight Reservation App is an application that allows a user to perform and manage flight reservations. A flight reservation has a destination airport associated. Similarly, it has a departure date and, if it is not "One Way", also a return date. A flight reservation has a class type that can be for instance: Economy, Business or First. A flight reservation must have one or more passengers associated. In turn, a passenger has an ID (e.g., Passport ID), name, date of birth, and country. When a user enters the application he may perform one of the following tasks:

- Manage his own list of passengers (typically himself, his relatives and friends), which allow him to add, edit or remove items from that list;
- Manage his own flight reservations, which allow him to add new ones, view and edit the details of an existing reservation or remove it from that list;
- Update the list of airports from an international external service.

4. XIS-Mobile Language

This new version of the XIS-Mobile language introduced new concepts and corrected some others in comparison to the version presented in [5], maintaining the multi-view organization and the design approaches. The next sub-sections describe the multi-view organization of the XIS-Mobile language, the dependencies between its views and the design approaches it supports.

4.1. Multi-View Organization

XIS-Mobile is organized in four major packages of views: Entities, Architectural, UseCases and User-Interfaces Views. In turn, the Entities View contains the Domain and BusinessEntities Views, while the User-Interfaces View contains the NavigationSpace and InteractionSpace Views. These views depend on each other and that fact influence the design approaches and the transformation stages (see **Figure 1**). The Domain View does not depend on any other view, since it is the starting point to define the problem domain. In turn, the BusinessEntities View depends on the Domain View, because business entities aggregate domain entities. The UseCases View depends on the BusinessEntities View and the Architectural View, since each use case must be connected to a business entity and/or provider. The Architectural View does not depend on any other view. The InteractionSpace and the NavigationSpace Views depend on each other. The InteractionSpace View also depends on the BusinessEntities View, because an interaction space can be associated to a business Entity; and on the Architectural View, because some widgets can trigger WebService actions defined on that view.

4.2. Design Approaches

XIS-Mobile supports two design approaches: the dummy approach and the smart approach. Essentially, the main difference between them is that the smart approach uses model-to-model transformations. By leveraging this

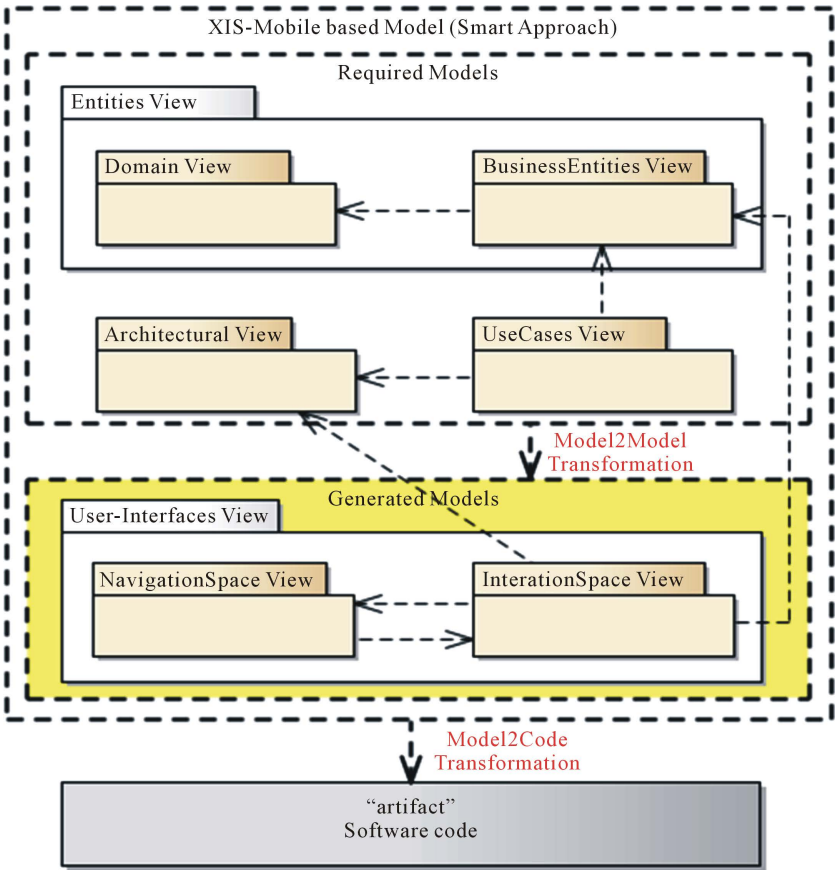

Figure 1. Multi-view organization of XIS-Mobile.

kind of transformations, XIS-Mobile avoids the manual creation of the User-Interfaces View package.

Using the *dummy approach*, the developer needs to define all views, except the Architectural View that is only required if the application interacts with external entities, like providers or web servers. On the other hand, using the *smart approach* (see **Figure 1**), the developer should define only the Domain, BusinessEntities and UseCases Views. Again, like in the dummy approach, the Architectural View is only necessary if the interaction with external entities is a requirement. Then, by leveraging the patterns represented by each use case and the information of the Domain and BusinessEntities Views is possible to perform model-to-model transformations and automatically generate the User-Interfaces View (InteractionSpace and NavigationSpace Views). Then the developer can customize them if he desires so. Therefore, this approach can be much less time-consuming than the dummy approach, since the more complex views are automatically generated. This approach should be used whenever possible, because it speeds up the modeling process. After defining all these view models either using the dummy or the smart approach, it is possible to generate native source code from them using model-to-text transformations.

4.3. Entities View

The Entities View defines the concepts relevant to the problem domain. It is subdivided in the Domain and BusinessEntities Views.

4.3.1. Domain View

The Domain View represents the entities that compose the problem domain as classes, using a traditional Class Diagram. These entities can contain one or more attributes and are related using associations, aggregations or inheritances. It is also possible to define enumerations and use them as entity attribute types.

The XIS-Mobile language provides the following stereotypes to be applied in this view: 1) XisEntity for classes; 2) XisEntityAttribute for attributes; 3) XisEntityAssociation for associations and aggregations; 4) XisEntityInheritance for inheritances; 5) XisEnumeration for enumerations; and 6) XisEnumerationValue for their values.

Considering the Flight Reservation App's Domain View (see **Figure 2**), there are three XisEntities: FlightReservation, Airport and Passenger. There is also a XisEnumeration named Class with three XisEnumerationValues: Economy, Business and First. The FlightReservation contains four XisAttributes that specify if the reservation is "One Way", its departure and return dates and its class type. The Airport has one XisAttribute representing its name. The Passenger has four XisAttributes for the passportID, name, date of birth and country.

4.3.2. BusinessEntities View

The BusinessEntities View represents high-level entities as classes, known as business entities, using a Class Diagram. A business entity is represented as a class with the stereotype XisBusinessEntity which aggregates one

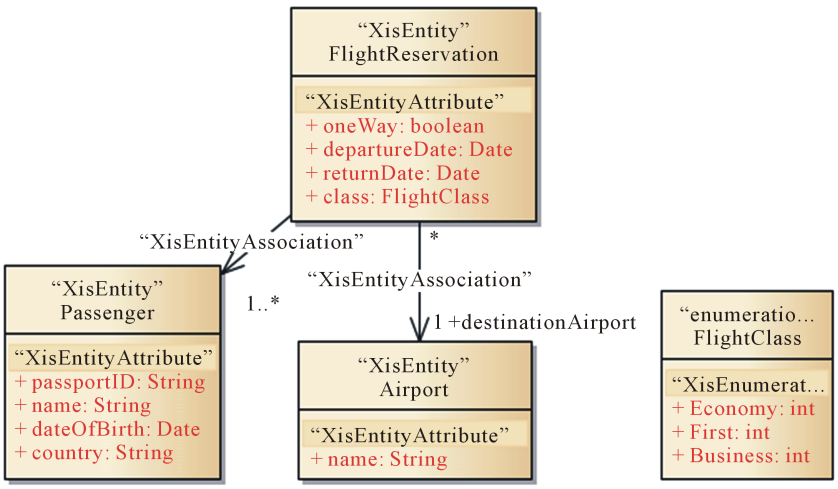

Figure 2. Flight reservation App's Domain View.

or more XisEntities (from the Domain View). The goal is to provide context to use cases and interaction spaces playing an important role during the transformation stages.

A business entity is defined by specifying a domain entity (from the Domain View) as its master entity and, if needed, other domain entities as its detail and reference entities. The definition of the master entity restricts the set of detail and reference entities that can be associated to the business entity in question. Both the detail and the reference entities must be associated to the master entity in the Domain View, through aggregations and associations, respectively. XIS-Mobile provides the following stereotypes to be applied in this view: 1) XisBusinessEntity for classes, 2) XisBE-EntityMasterAssociation, 3) XisBE-EntityDetailAssociation, and 4) XisBE-EntityReferenceAssociation for associations connecting business entities to domain entities, identifying their roles (respectively, master, detail and reference). These three associations also contain a tagged value named "filter" that allows the restriction of the domain entity's attributes that can be used in the context of the XisBusinessEntity.

Regarding the Flight Reservation App, we defined three XisBusinessEntities: FlightReservationBE, PassengerBE and AirportBE. The FlightReservationBE has FlightReservation as master entity and Airport and Passenger as references. The PassengerBE has Passenger as master entity and AirportBE has Airport as main entity.

4.4. Architectural View

The Architectural View represents the interactions between the mobile application and the external entities, and so, it can also be called "Distributed Systems View". The modeling of this view is performed in a Class Diagram. The mobile application, depicted as a class named XisMobileApp, can be connected to several XisServices, represented as interfaces.

A XisService supplies one or more operations, called XisServiceMethods, and is provided or realized by a class called XisProvider. A XisService is divided in two types: 1) XisInternalService, if it is provided by an entity inside the mobile device; and 2) XisRemoteService, if it is provided by an entity outside the mobile device. A XisInternalService can only be provided by a class called XisInternalProvider. Moreover, a XisInternalProvider has a tagged value that defines its type: 1) Location, which represents the location provider of the device and gives information about its geographical location; 2) Contacts, which allows the access to the contacts of the device; 3) Calendar, which allows the interaction with the calendar of the device; 4) Media, which allows the interaction with the camera, media recorder and media player; and 5) Custom, which is used when the developer wants to create his own provider.

In turn, each XisRemoteService can only be provided by a XisServer, which represents physically a web server, or a XisMobileClientApp, which represents a mobile application running on another device.

The XIS-Mobile language provides the following stereotypes to be applied in this view: 1) XisMobileApp, 2) XisInternalProvider, 3) XisClientMobileApp and 4) XisServer for classes; 5) XisInternalService and 6) XisRemoteService for interfaces; 7) XisServiceMethod for operations; 8) XisMobileApp-ServiceAssociation for associations between XisMobileApps and XisServices; 9) XisProvider-ServiceRealization for realization relationships between XisProviders and XisServices.

Considering the Flight Reservation App, it communicates with one XisRemoteService containing one XisServiceMethod for the operation of synchronizing airports provided by a XisServer named FlightServer.

4.5. UseCases View

The UseCases View represents the operations an actor can perform when interacting with the application in the context of a business entity (from the BusinessEntities View) and/or a provider (from the Architectural View), using a Use Case Diagram. This view plays an important role during the model-to-model transformation stages. Namely, according to the use case stereotype, tagged values and the business entity and/or provider associated, a certain type of User-Interfaces View models will be generated based on well defined UI patterns [23]. A use case can have two stereotypes: XisEntityUseCase and XisServiceUseCase.

A XisEntityUseCase represents an operation over a business entity and, consequently, its domain entities. It contains a set of tagged values representing the CRUD (Create, Read, Update, Delete) and the Search operations for the master, detail and reference entities. It was decided to include these set of operations, since it contains the most commonly used operations as it was observed in [24]. In addition to that, a XisEntityUseCase has a tagged value that defines its type: 1) EntityManagement; or 2) EntityConfiguration. A XisEntityUseCase of type EntityManagement will generate interaction spaces (screens are designated as interaction spaces in XIS-Mobile) for

managing a list of multiple instances of the master entity (from the associated business entity), while a XisEntityUseCase of type EntityConfiguration will generate interaction spaces to manage a single instance of the master entity associated. These two types are, in fact, patterns of generation of interaction spaces and in the future the goal is to add new types.

A XisServiceUseCase represents an action that uses the operations provided by a provider. It can also be connected to a business entity, representing that the operations have an effect over the business entity's domain entities. For now, this type of use case generates interaction spaces that use the operations provide by the provider and its associated services. Typically, this type of use case is used as an extension of the XisEntityUseCase. In addition, it is also possible to define actors in this view with the goal, in the future, of specifying the operations an actor are allowed to perform.

The XIS-Mobile language provides the following stereotypes to be applied in this view: 1) XisActor for actors; 2) XisEntityUseCase and 3) XisServiceUseCase for use cases; 4) XisActor-UCAssociation, for associations between actors and use cases; 5) XisEntityUC-BEAssociation for associations between XisEntityUseCases and XisBusinessEntities; 6) XisServiceUC-BEAssociation for associations between XisServiceUseCases and XisBusinessEntities; and 7) XisServiceUC-ProviderAssociation for associations between XisServiceUseCases and XisProviders.

Concerning the Flight Reservation App (see **Figure 3**), there were defined three use cases: 1) "Manage FlightReservations", a XisEntityUseCase of type "EntityManagement" connected to the FlightReservationBE; 2) "Manage Passengers", a XisEntityUseCase of type "EntityManagement" connected to the PassengerBE; and 3) "Sync Airports", a XisServiceUseCase connected to the AirportBE and to the FlightServer used to synchronize the list of airports with that server.

4.6. User-Interfaces View

The User-Interfaces View contains the NavigationSpace and InteractionSpace views that are used to define the screens of the mobile application, known as interaction spaces, as well as the navigation flow between them.

4.6.1. NavigationSpace View

The NavigationSpace View represents the navigation flow between the various interaction spaces of the mobile

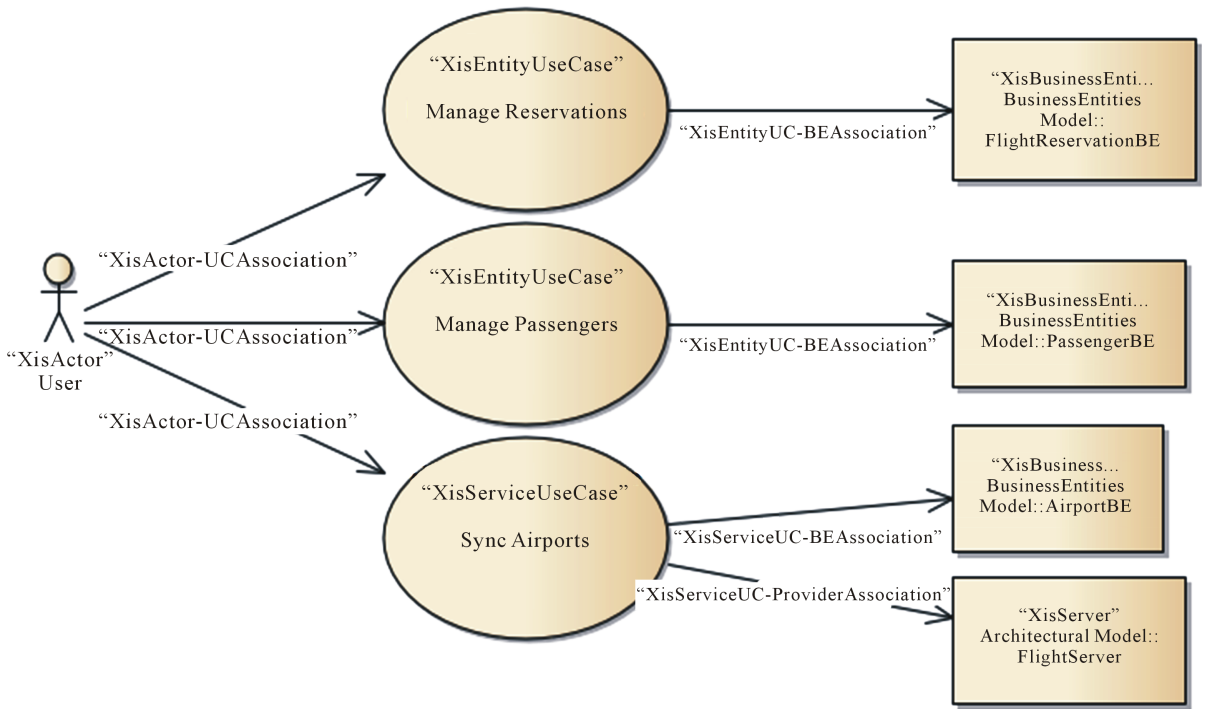

Figure 3. Flight Reservation App's UseCases View.

application, using a Class Diagram. This navigation flow is represented by directed associations with the stereotype XisInteractionSpaceAssociation. The XisInteractionSpaceAssociation is the only stereotype provided by the XIS-Mobile language to be applied in this view. Each XisInteractionSpaceAssociation is also responsible to show the event that triggered the navigation between the interaction spaces, because its name must be the same as the event's name. This view provides a good support for documenting the structure of the system and eases the introduction of future corrections and improvements in the mobile application navigation. For simplicity and readability reasons, the details of each interaction space are represented in the InteractionSpace View.

The Flight Reservation App's NavigationSpace View is composed by eight XisInteractionSpaces: 1) HomeIS, which is the home screen of the application; 2) FlightReservationListIS, which lists all the flight reservations; 3) FlightReservationEditorIS, which displays all the information of a flight reservation and allows its edition; 4) PassengerManagerIS, which lists all the passengers associated to a certain flight reservation; 5) PassengerDetailIS, which displays the information of a passenger associated to a reservation and allows its edition; 6) PassengerListIS, which lists all the passengers; 7) PassengerEditor, which allows the edition or creation of a passenger; and 8) AirportListIS, which lists all airports and allows to synchronize them with a remove server.

4.6.2. InteractionSpace View

The InteractionSpace View represents several details about a certain screen or interaction space of the mobile application, using a Class Diagram. Due to the amount of abstractions it involves, this view is perhaps the hardest to design and most complex view of the XIS-Mobile language. It is source of several details such as the UI layout, the events a certain UI component can trigger and the gestures that can be performed. All this information will feed the model-to-text transformations.

Each interaction space is represented as a stereotyped class named XisInteractionSpace. It is the main component of this view, since it represents an application screen. Each XisInteractionSpace contains one or more XisWidgets, also represented as classes. In addition to that, each interaction space can be connected to a business entity, through a XisIS-BEAssociation, defining the domain entities that can be bound to the interaction space's inner elements.

A XisWidget represents a UI widget or control that builds up the GUI. It is divided in two major stereotype categories: XisCompositeWidget and XisSimpleWidget. A XisCompositeWidget is a container class that groups other XisWidgets (both simple and composite). As can be noticed, this stereotype follows the well-known Composite design pattern [25]. The XisCompositeWidget plays an important role in this view, because it allows the definition of a specific context by associating a domain entity to its "domainEntityName" tagged value. This value should be filled with the name of a domain entity contained in the business entity associated to the interaction space. In addition, whenever this value is filled, all the XisWidgets inside the XisCompositeWidget involved have access to the domain entity. Some examples of specializations of XisCompositeWidgets are the XisList, XisForm, XisMenu or XisDialog. In turn, a XisSimpleWidget represents the set of simple controls and cannot contain other elements. Some examples are the XisLabel, XisTextBox or XisButton. A XisSimpleWidget can be bound to a domain entity attribute value through its tagged value "entityAttributeName".

Every XisWidget can have many gestures attached, called XisGestures, but at most only one of each type. The set of gestures supported is: Tap, DoubleTap, LongTap, Swipe, Pinch and Stretch. Then, each gesture can trigger a set of events or actions, the XisActions, which ranges from actions like Cancel or WebService to CRUD operations. A XisAction can also trigger navigation between interaction spaces through the tagged value "navigation" that should be filled with the name of the target interaction space. The XIS-Mobile language also gives users the flexibility to define the stub of a custom operation by providing a XisAction of type Custom. XisGestures are represented as classes, while XisActions are represented as operations. XisGestures can be attached to a XisWidget in two ways: through explicit associations and through tagged values that represent the default gestures used with that XisWidget. For example, XisButtons have the gesture Tap as default, because it is the most common gesture when interacting with buttons. Therefore, XisButtons have the tagged value "onTap" that should have the name of the XisAction triggered whenever the Tap gesture is detected.

Moreover, a XisWidget can have a constant value and for that purpose it has the tagged value "value". It is also possible to assign a value from an expression using the tagged value "valueFromExpression".

Regarding the Flight Reservation App (see **Figure 4**), the FlightReservationListIS is an example of one of its interaction spaces. This interaction space has a XisList, a XisMenu of type "OptionsMenu" and a XisMenu of type "ContextMenu". The XisList is a list of flight reservations and contains one XisListItem with a "Tap" gesture

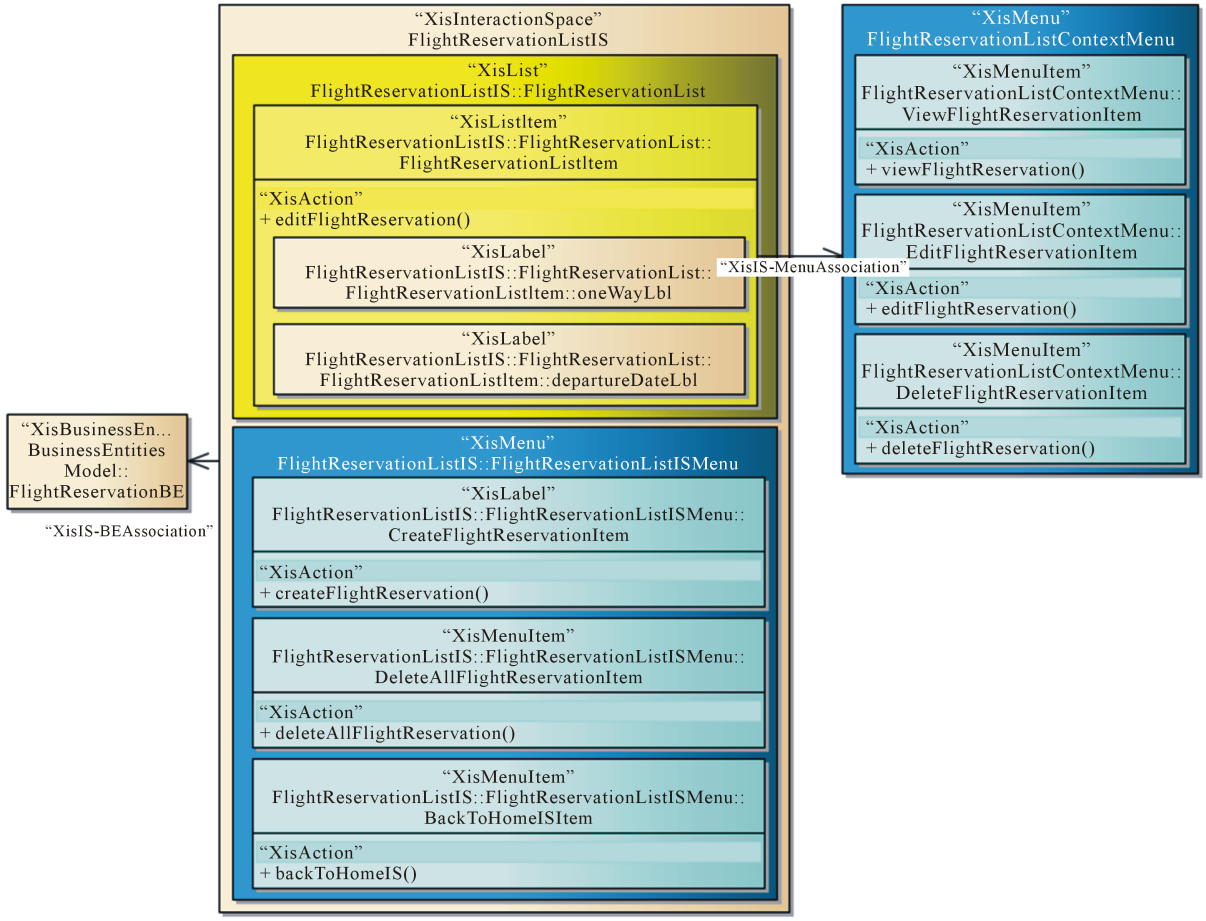

Figure 4. The FlightReservationListIS insteraction space diagram of the flight reservation app.

as default that triggers the navigation to the FlightReservationEditorIS. This list item is associated to the context menu, which is opened whenever the list item is long-tapped. The context menu contains three XisMenuItems that allow navigating to the visualization and edition screen of the associated reservation, or instead delete it from the list. The options menu contains three XisMenuItems that allow creating a reservation by navigating to the FlightReservationEditorIS, the deletion of all reservations, and going back to the HomeIS.

5. XIS-Mobile Framework

The XIS-Mobile language is supported by a MDD-based framework[1] implemented using the Model Driven Generation (MDG) Technologies provided by Sparx Systems Enterprise Architect (EA) along with the Eclipse Modeling Framework (EMF). The use of these technologies leverages the environment they provide, as well as some compatible plug-ins. As shown in **Figure 5**, the suggested development process of a mobile application using the XIS-Mobile framework consists in four steps: 1) the definition of the required views using the Visual Editor; 2) their validation with the Model Validator; 3) the generation of the User-Interfaces View models with the Model Generator; and finally 4) the generation of the application's source code through the Code Generator. After Step 3), if the designer finds that the model is not yet completed, the process returns to Step 1) for a new iteration. It is important to emphasize that only Step 1) is manual, while the other three are automatic.

First, the Visual Editor is implemented on top of EA through the use of an MDG Technology plug-in. This plug-in technology allowed the definition of UML profiles fully compliant with the OMG specification for UML2, the creation of toolboxes, diagram types, diagram templates and patterns customized to the XIS-Mobile profile.

[1]More details about XIS-Mobile are available in https://github.com/xis-mobile.

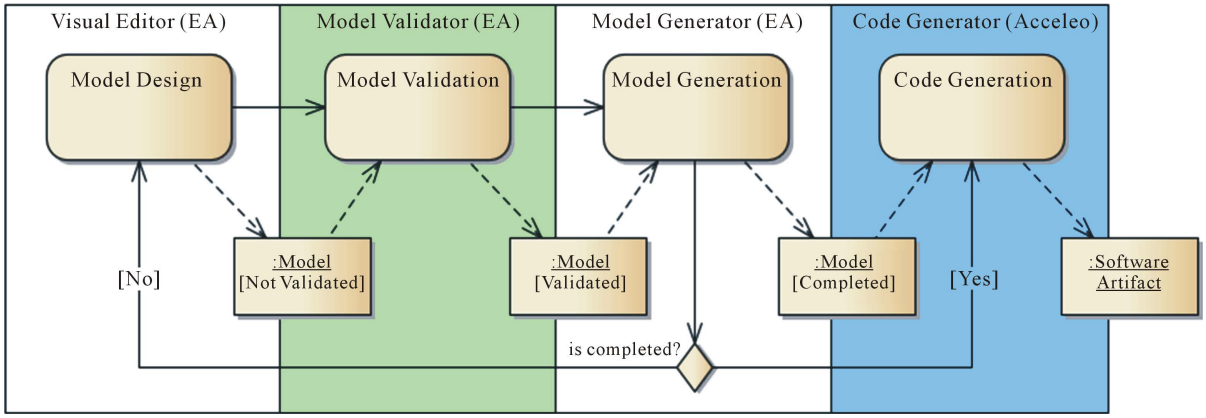

Figure 5. Suggested development process with the XIS-Mobile framework.

Second, the Model Validator is implemented as a plug-in using EA's Model Validation API [26]. It plays a crucial role avoiding errors by the designer, improving the quality of the models and, consequently, enhancing the quality of the generated models and code. Despite not being a standard like OCL [27], this solution allows the definition of custom error messages for each constraint, the assignment of severity levels (error or warning) to them, and the immediate navigation to the element that caused the message.

Third, the Model Generator is implemented as a plug-in leveraging EA's Automation Interface [28]. It permits accessing the repository containing the created diagrams and their elements, as well as creating new diagrams and elements. Thus, the Model Generator performs the model-to-model transformations using the information of the UseCases, Domain, BusinessEntities and Architectural views. Usually a single instance of each one of these diagrams yields to multiple InteractionSpace View diagrams and a single NavigationSpace View diagram.

Fourth, the Code Generator is based on Acceleo [29], a template-based code generator framework available as an Eclipse plug-in. Acceleo implements the MOF MTL (Model to Text Language) standard [30] and allows the definition of code templates for any kind of model compatible with EMF. The code templates are composed by regular text (static part of the template) and several annotations (dynamic part of the template) that are replaced by values of the model during generation time. For now, the XIS-Mobile framework supports the generation of applications for Android, and also early versions for iOS and Windows Phone have been developed. An important point is that whenever the user desires to add support for other platforms, he needs to define the corresponding code templates.

6. Evaluation

To better evaluate XIS-Mobile, receive feedback from people not directly involved in this work and detect potential bugs and user limitations, we decided to conduct a pilot user session. This session involved a group of 9 participants in total with ages ranging from 21 to 48 and with at least a Bachelor of Science degree. All participants had previous knowledge and experience with UML, 5 of them also had experience with mobile application development and 7 had professional experience in the Information Technology area. The user session was conducted under the following conditions: tests took place in the laboratory (controlled environment); realization of the task without previous use and learning (for the first time); the user must had a computer running Windows, Java and previously installed Sparx Enterprise Architect (version 7.5, 10 or above); Direct Observation, *i.e.*, while users performed the assigned task, their behavior and performance could be logged; users were free to think out loud and share ideas if they wanted.

Based on these conditions participants received a 40 minutes training explaining the fundamental concepts of the XIS-Mobile language and its framework, as well as a demonstration of the development of a simple case study using XIS-Mobile from model design until code generation. Thereafter, they had a script describing the Flight Reservation App (see Section 3) and were asked to create the corresponding models in XIS-Mobile, validate them and launch the code generation for Android in a maximum period of 60 minutes. Finally, participants were asked to fill in a questionnaire to rate: 1) the XIS-Mobile language; 2) its supporting framework; and 3) the

overall approach. The answers were classified in a scale of: 0 (N/A-Do not know), 1 (Very Low), 2 (Low), 3 (Medium), 4 (High) and 5 (Very High).

The *XIS-Mobile Language* aspect included five questions:

QL.1. How suitable is the size (number of concepts) of the language?

QL.2. How easy to use is the notation used (defined as a UML Profile)?

QL.3. How easy to learn is the language without the UI concepts?

QL.4. How easy to learn is the language with the UI concepts?

QL.5. How suitable is the language for the Mobile Apps development domain?

Table 1 summarizes the average score for the answers regarding the language aspect, broken down by question. In general, we can observe that all questions had a positive score (greater than 2.5) implying some sort of success. Concerning Question 1, we conclude that almost all participants considered the language size suitable (with a score of 4 or 5), while 1 participant considered it inadequate, possibly too large. Questions 2, 3 and 5 were the ones that obtained more favorable answers indicating that XIS-Mobile language notation is easy to use, easy to learn without the UI concepts and suitable for the mobile application development domain, respectively. However, from the answers to Question 4 we can conclude that participants found the User-Interface View more complex and harder to learn than the other views. This can be explained by the degree of detail, specially of the InteractionSpace View that contains a huge set of concepts either for UI widgets, gestures or actions. This is a challenging point that we will take into account in future developments with the goal of improving the representation of each interaction space in a "What You See Is What You Get" (WYSIWYG) fashion. Summarizing, we assume that XIS-Mobile language contains several concepts and initially can be a bit difficult to learn and understand all of them. Despite that we believe from these results that its size, notation and concepts are adequate for the development of mobile applications.

The *XIS-Mobile Framework* aspect included five questions:

QF.1. How do you rate the usability of EA with the XIS-Mobile plugin?

QF.2. How do you rate the usability of the Model Editor (Stereotypes, Toolboxes, Project template)?

QF.3. How do you rate the usability of the Model Validator?

QF.4. How do you rate the simplicity of the Model-to-Model transformation (Model generation) process?

QF.5. How do you rate the simplicity of the Model-to-Text transformation (Code generation) process?

Table 2 summarizes the average score for the answers to the Framework aspect, broken down by question. Similarly to the previous aspect, we can observe that all answers had a positive score. Question 1 and 2 were the ones that obtained more positive answers and so we can conclude that participants felt highly comfortable with the overall usability of EA with the XIS-Mobile plugin and modeling the Flight Reservation App with the Model Editor. Concerning Questions 3, 4 and 5, it seems that participants found the process of triggering code generation simpler than using the model validation and the model generation process. This can be explained by the fact that the model generation process requires more time to be properly learned, because it highly depends on the use cases that can involve different types, tagged values and associations to business entities and providers.

Finally, the *XIS-Mobile General Approach* aspect included two questions:

QA.1. How do you rate the productivity with XIS-Mobile comparing to traditional processes?

QA.2. Would you use such a tool on your own Mobile App projects?

Table 1. Questionnaire's average score (in a scale of 0 - 5) by question for the XIS-Mobile language aspect.

	XIS-Mobile Language				
	QL.1	QL.2	QL.3	QL.4	QL.5
Average	3.67	4.22	3.89	3.56	4.11

Table 2. Questionnaire's average score (in a scale of 0 - 5) by question for the XIS-Mobile framework aspect.

	XIS-Mobile Framework				
	QF.1	QF.2	QF.3	QF.4	QF.5
Average	4.44	4	3.33	3.33	3.56

Table 3 presents the average score for the answers concerning the XIS-Mobile General Approach aspect, broken down by question. We can observe that the score obtained for both questions is highly positive. However, the participants expressed more interest in using XIS-Mobile in mobile app projects than considering the productivity with XIS-Mobile high comparing to the traditional approach.

Summarizing, as can be seen in **Table 4**, the results were generally encouraging with positive scores in all three analyzed aspects. Nevertheless, it was observed that the XIS-Mobile Framework and namely its model generation process had the lowest score and possibly need to be refined in order to improve their simplicity. Moreover it was observed that the learnability of the XIS-Mobile language should be improved namely in terms of the InteractionSpace View concepts representation. It can be stated that the number of participants of the session is relatively small. We believe that number is sufficient to take meaningful conclusions because usability experts have noted that a group of 5 testers is enough to uncover over 80% of the usability problems [31]. Also, since this evaluation focuses on the usability of the language, framework and approach, we believe 9 participants is a reasonable number for an exploratory assessment, at least in order to identify challenges on the usability of these aspects.

7. Conclusions

In this paper we presented XIS-Mobile, a domain specific language (defined as a UML profile), focused on the development of mobile applications and on mitigating the problems related to the complexity of software development and mobile platform fragmentation. XIS-Mobile has a multi-view organization and supports two design approaches: the dummy approach and the smart approach. XIS-Mobile has a supporting framework based on Sparx Systems Enterprise Architect MDG Technology and EMF, which intends to generate source code for multiple platforms from a single model specification, through model-to-model and model-to-text transformations. Composed of four major components, this framework suggests developing a mobile application in four steps whenever possible: defining the required views using the Visual Editor, validating them using the Model Validator, generating the User-Interfaces View models with the Model Generator, and finally generating the application's source code through the Code Generator. This way the developer takes advantage of the MDD benefits, namely increasing his productivity by using a single specification of the system with a PIM, by avoiding the implementation of boilerplate code and reducing errors.

XIS-Mobile has been developed over the last eighteen months following the Action Research Methodology and has been evaluated in order to assess its usefulness and adequacy to the purpose of modeling mobile applications. This evaluation process has been done in an iterative and gradual way using case studies applications and a user session. The user session focused on three aspects of XIS-Mobile: the Language, the Framework and the General Approach. During the user session the participants had to develop a case study application using XIS-Mobile and in the end fill in a questionnaire. The questionnaires collected positive results and showed preliminary evidences that demonstrate XIS-Mobile's usefulness and feasibility.

For future work, there are plans to add new use case types and patterns, in order to extend the range of the smart approach and its associated model-to-model transformations. We intend to further evaluate XIS-Mobile,

Table 3. Questionnaire's average score (in a scale of 0 - 5) by question for the XIS-Mobile General Approach aspect.

	XIS-Mobile General Approach	
	QA.1	QA.2
Average	4	4.33

Table 4. Questionnaire's average score (in a scale of 0 - 5) for each XIS-Mobile framework aspect.

	XIS-Mobile		
	Language	Framework	General Approach
Average	3.89	3.73	4.17

possibly conducting usability tests in the academic field and for teaching purposes. There is also concern in improving the representation of XIS-Mobile stereotypes with the goal of make them more appealing and closer to the final result. The integration with other visual editors represents another interesting future step, aiming to offer more portability and flexibility to the users. The design and development of more complex and real-world applications consist in interesting future research directions, to better exercise all language concepts. Finally, some more research has to be done related on how to extend XIS-Mobile to better support scenarios related with the mobile cloud computing (MCC) [32], context awareness [9] and CMS-based [33] systems.

Acknowledgements

This work was partially supported by Portuguese funds through FCT-Fundação para a Ciência e a Tecnologia, under project PEst-OE/EEI/LA0021/2013 and DataStorm Research Line of Excellency funding (EXCL/EEI-ESS/0257/2012).

References

[1] Meeker, M., Devitt, S. and Wu, L. (2010) Internet Trends. Morgan Stanley Research. http://www.slideshare.net/CMSummit/ms-internet-trends060710final

[2] Schuermans, S., Constantinou, A. and Vakulenko, M. (2014) Mobile Megatrends 2014, Vision Mobile. http://www.visionmobile.com/product/mobile-megatrends-2014

[3] Atkinson, C. and Kühne, T. (2003) Model-Driven Development: A Metamodeling Foundation. IEEE Software. IEEE Computer Society Press, Los Alamitos.

[4] Stahl, T. and Volter, M. (2006) Model-Driven Software Development: Technology, Engineering, Management. John Wiley & Sons Ltd., Chichester.

[5] Ribeiro, A. and Silva, A.R. (2014) XIS-Mobile: A DSL for Mobile Applications. *Proceedings of* 29*th Symposium on Applied Computing* (*SAC*'14), Gyeongju, 24-28 March 2014, 1316-1323.

[6] Silva, A.R., Saraiva, J., Silva, R. and Martins, C. (2007) XIS-UML Profile for Extreme Modeling Interactive Systems. *Proceedings of the* 4*th International Workshop on Model-Based Methodologies for Pervasive and Embedded Software* (*MOMPES*'07), Braga, 31 March 2007, 55-66.

[7] Silva, A.R., Saraiva, J., Ferreira, D., Silva, R. and Videira, C. (2007) Integration of RE and MDE Paradigms: The Project It Approach and Tools. *IET Software Journal*, **1**, 294-314.

[8] Baskerville, R.L. (1999) Investigating Information Systems with Action Research. Communications of the Association for Information Systems. Association for Information Systems, **2**, Article 4.

[9] Boudaa, B., Camp, O., Hammoudi, S. and Chikh, M.A. (2012) Model-Driven Development of Context-Aware Services: Issues, Techniques and Review. *International Conference on IT and e-Services* (*ICITeS*), Sousse, 24-26 March 2012, 1-8.

[10] Belloni, E. and Marcos, C. (2004) MAM-UML: An UML Profile for the Modeling of Mobile-Agent Applications. 24*th International Conference of the Chilean*, Computer Science Society, Arica, 11-12 November 2004, 3-13.

[11] Phone Gap Website. http://phonegap.com

[12] Appcelerator Titanium Website. http://www.appcelerator.com/titanium/

[13] Singh, I. and Palmieri, M. (2011) Comparison of Cross-Platform Mobile Development Tools. IDT. Malardalen University, Västerås.

[14] Paananen, T. (2011) Smartphone Cross-Platform Frameworks. B.Sc. Thesis, JAMK University of Applied Sciences, Jyväskylä.

[15] Ribeiro, A. and Silva, A.R. (2012) Survey on Cross-Platforms and Languages for Mobile Apps. *Proceedings of the* 8*th International Conference on the Quality and Communications Technology* (*QUATIC*), Lisbon, 3-6 September 2012, 255-260.

[16] Ranabahu, A.H., Maximilien, E.M., Sheth, A.P. and Thirunarayan, K. (2011) A Domain Specific Language for Enterprise Grade Cloud-Mobile Hybrid Applications. *Proceedings of the Compilation of the Co-Located Workshops on SPLASH*'11, Portland, 22-27 October 2011, 77-84.

[17] Kramer, D., Clark, T. and Oussena, S. (2010) MobDSL: A Domain Specific Language for Multiple Mobile Platform Deployment. *Proceedings of the* 2010 *IEEE International Conference on Networked Embedded Systems for Enterprise Applications* (*NESEA*), Suzhou, 25-26 November 2010, 1-7. http://dx.doi.org/10.1109/NESEA.2010.5678062

[18] Balagtas-Fernandez, F., Tafelmayer, M. and Hussmann, H. (2010) Mobia Modeler: Easing the Creation Process of

Mobile Applications for Non-Technical Users. *Proceedings of the* 15*th International Conference on Intelligent User Interfaces* (*IUI*'10), Hong Kong, 7-10 February 2010, 269-272.

[19] Calvary, G., *et al.* (2002) The CAMELEON Reference Framework. Deliverable 1.1, CAMELEON Project.

[20] Trindade, F.M. and Pimenta, M.S. (2007) Prototyping Multi-Platform Software Using the UsiXML4ALL. Technical Report.

[21] Patternò, F., Santoro, C. and Spano, L.D. (2009) MARIA: A Universal, Declarative, Multiple Abstraction-Level Language for Service-Oriented Applications in Ubiquitous Environments. *ACM Transactions on Computer-Human Interaction*, **16**, 1-30.

[22] Wolber, D. (2011) App Inventor and Real-World Motivation. *Proceedings of the* 42*nd ACM Technical Symposium on Computer Science Education* (*SIGCSE*'11), Dallas, 9-12 March 2011, 601-606.

[23] Neil, T. (2012) Mobile Design Pattern Gallery: UI Patterns for Mobile Applications. O'Reilly Media, Inc., Sebastopol.

[24] Langlands, M. (2010) Inside the Oval: Use-Case Content Patterns. Technical Report, Planet Project. http://planetproject.wikidot.com/use-case-content-patterns

[25] Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1995) Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Boston.

[26] EA's Model Validation API Website. http://goo.gl/XwUvsx

[27] Object Management Group: Object Constraint Language (OCL) Specification. http://www.omg.org/spec/OCL

[28] EA's Automation Interface Website. http://goo.gl/r9jz0Z

[29] Acceleo Website. http://www.eclipse.org/acceleo

[30] Object Management Group: MOF Model to Text Transformation Language (MOFM2T). http://www.omg.org/spec/MOFM2T/1.0/

[31] Nielsen, J. and Landauer, T.K. (1993) A Mathematical Model of the Finding of Usability Problems, INTERCHI'93.

[32] Fernando, N., *et al.* (2013) Mobile Cloud Computing: A Survey. *Future Generation Computer Systems*, **29**, 84-106. http://dx.doi.org/10.1016/j.future.2012.05.023

[33] Saraiva, J.S. and Silva, A.R. (2009) CMS-Based Web-Application Development Using Model-Driven Languages. *Proceedings of the* 4*th International Conference on Software Engineering Advances*, Porto, 20-25 September 2009, 21-26.

Scientific Research Publishing (SCIRP) is one of the largest Open Access journal publishers. It is currently publishing more than 200 open access, online, peer-reviewed journals covering a wide range of academic disciplines. SCIRP serves the worldwide academic communities and contributes to the progress and application of science with its publication.

Other selected journals from SCIRP are listed as below. Submit your manuscript to us via either submit@scirp.org or Online Submission Portal.