Development of Mobile Applications using a Model-Driven Software Development Approach

André Ribeiro
Instituto Superior Técnico
Lisbon, Portugal
andre.ribeiro@tecnico.ulisboa.pt

Alberto Rodrigues da Silva
INESC-ID / Instituto Superior Técnico
Lisbon, Portugal
alberto.silva@tecnico.ulisboa.pt

ABSTRACT
Mobile applications are becoming increasingly more present in our daily lives, allowing people to perform several tasks through the use of mobile devices. Despite fostering the innovation, the rapid growth of the mobile market resulted in some fragmentation of the mobile platforms. The existence of different mobile operating systems with different programming languages and tools can be a problem when we want to develop mobile applications for multiple platforms. Rewriting the application for each platform is usually impracticable either in terms of budget or time.

Therefore, a solution that can generate cross-platform applications without compromising the quality, would decrease the time to market and increase the number of potential users. Fortunately, some work has been conducted over the last years to tackle this problem, namely through the use of web technologies, cross-platform tools and approaches based on MDD.

This paper proposes the use of a MDD approach for the development of mobile applications. This approach, named XIS-Mobile, uses a domain specific language (defined as a UML profile) and its MDD-based framework to address mobile platform fragmentation. They propose the definition of platform-independent models to describe mobile applications and from them automatically generate the application’s source code for multiple platforms. This paper presents XIS-Mobile (language and framework), provides an evaluation of it and discusses its main challenges and benefits in the context of the cross-platform mobile application development.

Categories and Subject Descriptors

General Terms
Design, Languages.

Keywords
Model-Driven Development, Cross-Platform, Mobile Applications, Domain Specific Language.

1. INTRODUCTION
Over the years software systems have become more complex and sophisticated, not just in terms of the problems they try to solve, but also in terms of the technologies, tools and languages they use. Software Engineering has played an important role in the development of these systems allowing developers managing and controlling the complexity through the use of methodologies that clearly define the development process, mechanisms of abstraction, and software quality approaches [1-3].

Simultaneously, a great evolution of the mobile computing industry has been happening, especially over the last decade, with the emergence of new devices increasingly more powerful and operating systems with better functionalities [4]. Therefore, mobile devices have become more present than ever in our daily life tasks. The common tasks of making calls and sending text messages are being backgrounded by others that make use of GPS, accelerometer, video or audio. The existence of these built-in features along with the applications that use them, make mobile devices, such as smartphones or tablets, very desirable and popular nowadays.

All these developments resulted from the intense competition among the major companies that dominate the mobile market, namely Apple, Google, Microsoft and Blackberry. Over the last years these companies developed their own platform with specific tools and application market. This competition has enabled a rapid growing of mobile market and the emergence of increasingly better features, but was also responsible for a certain fragmentation of the operating systems that support each platform. Platform fragmentation becomes a serious issue when someone wants to develop an application for multiple operating systems. Due to the specificity of each platform, an application developed for a given operating system is incompatible with the others. This lack of compatibility forces the developers to rewrite the application for each one of the target platforms increasing the effort and the time to market of that application.

Fortunately, over the last years some work has been conducted to tackle both problems presented previously: the software development complexity and the mobile platform fragmentation. Several approaches like the use of web technologies (e.g. HTML5 and JavaScript libraries), cross-platform tools and frameworks which allow the creation and distribution of mobile application to multiple platforms, or approaches based on Model-Driven Development, MDD, (or Model-Driven Engineering, MDE) [5-7], like the one presented in this paper, are examples of solutions focused on solving these problems. In particular, MDD seeks to move the source code development process to a more abstract level of specification, recurring to models. These models consist
in abstract representations of concepts specific of a certain problem domain. Its main goal is that the model guides all the development activities, resulting in quality improvements, increased productivity [8] and shorter time to market [9]. One of the greatest benefits of adopting MDD is the ability to specify the structure and the behavior of a software system in a more platform agnostic way than the traditional programming approaches [10].

The solution presented in this paper – the XIS-Mobile language and framework –, proposes a UML profile that allows the specification of mobile applications in a platform-independent way and using domain specific concepts. This way, both complexity and platform fragmentation problems can be mitigated resulting in increased productivity. Currently, the XIS-Mobile framework supports the generation of Android, iOS and Windows Phone applications. This framework generates the skeleton of the application code and, if needed, the developer can customize it.

The outline of this paper is as follows: Section 2 overviews the background from which this work has been based and presents the domain analysis that was conducted to identify specific issues of mobile applications. Section 3 describes the XIS-Mobile language, namely its main views and elements using a simple case study application, in order to show XIS-Mobile in practice and validate its usefulness and adequacy to the mobile domain. Section 4 details the development with XIS-Mobile, namely by clarifying the dependencies between its views and the supported design approaches. Section 5 describes the framework that supports the XIS-Mobile language and that shows the feasibility of its application into mobile platforms. Section 6 presents and discusses the preliminary results obtained from the conduction of an assessment session with user not involved in this work. Section 7 discusses the related work. Finally, Section 8 concludes the paper, summarizing its key points and referring the future work.

2. BACKGROUND
This section presents the background that motivated the development of a UML Profile focused on mobile application development.

2.1 Specific Issues of Mobile Applications
Before defining a Domain Specific Language (DSL) it was necessary to conduct a domain analysis with the goal of identifying the concepts and patterns that characterize mobile applications. This analysis was performed not only in an empirical way, but also with the analysis of existing literature [11-16].

The internet connection is a common feature of the mobile applications and what allows the continuous work while moving between spaces and devices. The internet connection is used not only to retrieve and store data, but also to overcome limitations related to performance (e.g. battery life, computation and bandwidth), environment (e.g., heterogeneity, scalability and availability) and security (e.g. authentication, authorization and privacy) [17].

The wide variety of mobile devices causes the existence of heterogeneity in their screen resolution. Thus, a mobile application should be designed having in mind that it can be used in devices with different screen sizes and resolutions.

Gesture detection also plays a crucial role in mobile applications design. Gestures consist in touch events which represent the main input mechanism used to interact with mobile applications. They are used not only to select something, but also to navigate between screens, what makes mobile applications event-driven applications. For example, navigation can be performed through a tap in a button or even through a swipe gesture.

2.2 XIS Language
The work presented in this paper materializes an idea for the extension of an existing UML profile named XIS [18][19]. XIS focuses on the design of interactive software systems at a PIM level (Platform-Independent Level, according to MDA terminology [20]) following a Model-Driven Development approach. XIS considers three major groups of views: Entities, Use-Cases and User-Interfaces.

The set of Entity views is composed by the Domain and BusinessEntities views. In the Domain view are represented the relevant classes to the problem domain, their attributes and the relationships among them. In turn, the goal of the BusinessEntities view is to define higher-level entities, known as business entities, that aggregate entities of the Domain view or other business entities and that are easier manipulated in the context of a given use case.

The set of Use-Cases View contains the Actors and the UseCases views. The Actors view specifies the entities that can perform actions over the system. The UseCases view relates the actors defined in the previous view with the operations they can perform over the business entities when interacting with the system.

At last, the set of User-Interface views defines the interaction spaces, i.e., the screens of the system, and the navigation flow between them. It comprises the NavigationSpace and InteractionSpace views. The NavigationSpace view defines the navigation flow between interaction spaces with which the user interacts, while the InteractionSpace view details the elements of the graphical interface contained in each screen and also can specify the access control of the actors to these elements.

XIS also defines two modeling approaches: the smart approach and the dummy approach [21]. To take full advantage of the smart approach, the designer only needs to design the Domain, BusinessEntities, Actors and UseCases views. After that, the User-Interfaces views can be automatically generated through model-to-model transformations (from the Domain and UseCases views) and then extended and refined through direct design. On the other hand, in the dummy approach, the designer has to define by scratch the entire Domain, Actors, NavigationSpace and InteractionSpace views.

XIS represents a useful solution to model simple desktop or web interactive applications [19]. However, when the goal is to model mobile applications XIS presents some limitations, namely regarding a proper support for the specification of gestures, internet connection, localization and other context-aware issues commonly used in mobile applications. In addition to that, from what have been researched and our knowledge, the smart approach proposed by XIS was never actually implemented and the support of the XIS language was achieved by a proprietary tool that has not been maintained. Thus, this paper, not only presents the XIS-Mobile language, but also its framework based
on Sparx Systems Enterprise Architect\(^1\) (EA), a widely used and popular modeling tool. This framework will be described in detail in section 5, but it is important to highlight that it already implements the smart approach proposed by XIS.

### 3. XIS-MOBILE LANGUAGE

As mentioned in Section 2, the XIS-Mobile language reuses some of the best concepts proposed on the XIS language and introduces new ones, resulting from the domain analysis, in order to be more appropriate to mobile applications design. Figure 1 depicts the multi-view organization proposed by the XIS-Mobile language to model a mobile application.

For better understanding and simplicity of the explanation, a small case study which describes a simple, but practical mobile application will be used (see table below).

<table>
<thead>
<tr>
<th>Case Study – The Flight Reservation App</th>
</tr>
</thead>
<tbody>
<tr>
<td>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:</td>
</tr>
<tr>
<td>- Manage his own list of passengers (typically himself, his relatives and friends), which allow him to add, edit or remove items from that list;</td>
</tr>
<tr>
<td>- 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;</td>
</tr>
<tr>
<td>- Update the list of airports from an international external service.</td>
</tr>
</tbody>
</table>

#### 3.1 Entities View

The first package of views of the XIS-Mobile profile is the Entities View, which contains the Domain and BusinessEntities views. This package is used to identify the entities and concepts that are relevant to the problem domain.

##### 3.1.1 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 connected using associations, aggregations or inheritances. This view also allows defining enumerations and using 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 enumeration literals or 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.

them. Due to this fact, the BusinessEntities View plays 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 (see Figure 3), 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 while the AirportBE has Airport as main entity.

![Fig. 3. BusinessEntities View of the Flight Reservation App.](image)

### 3.2 Architectural View

The Architectural View represents the interactions between the mobile application and any external entities, and so, it can also be called “Distributed Systems View”. This view is modeled using a Class Diagram. The mobile application is depicted as stereotyped class named XisMobileApp and 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 as one of the: (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 physically represents 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; and (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.

### 3.3 UseCases View

The UseCases View represents the operations an actor can perform when interacting with the mobile application, in the context of a business entity (from the BusinessEntities View) and/or a provider (from the Architectural View). This view is modeled using a Use Case Diagram and plays an important role during the M2M transformation stage. 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 [16]. The UI Thus, a use case can have two stereotypes: XisEntityUseCase and XisServiceUseCase.

A XisEntityUseCase represents an action 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, because it contains the most commonly used operations as it was observed in [22]. 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 the goal is to add new types in the future.

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 provided 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 is 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 4), 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.

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.

### 3.4 InteractionSpace View

The InteractionSpace View represents the content of a certain interaction space or screen of the mobile application, using a Class Diagram. Due to the amount of abstractions it involves (see Figure 5), this view is perhaps the hardest to design and the most complex view of the XIS-Mobile language. It is source of several details such as the UI layout, the events a certain UI widget can trigger and the gestures that can be performed. All this information will feed the M2T transformation stage in order to generate the respective UI source code of the mobile application.

Each interaction space is represented as a stereotyped class named XisInteractionSpace and it is the main component of this view, because 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 [23]. 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. XisCompositeWidgets are purposely represented with different colors from other XisWidgets, in order to aware the designer to the special context they define and also to ease the visualization of the elements contained by XisCompositeWidgets. For instance, a XisMenu is represented in dark blue and a XisMenuItem, contained by it, in light blue. In turn, a XisSimpleWidget represents the set of simple controls that 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 actions or events, 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: (1) through explicit associations; and (2) 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 commonly used 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 6), 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 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 creation of a reservation by navigating to the FlightReservationEditorIS, the deletion of all reservations of the list and going back to the HomeIS.

4. DEVELOPMENT WITH XIS-MOBILE

As mentioned before, XIS-Mobile reuses some ideas defined in XIS, and therefore the development process is somehow similar, despite the differences between both languages. Thus, the next subsections describe the dependencies between the views proposed by XIS-Mobile and its design approaches, in order to emphasize the differences between XIS-Mobile and XIS.

4.1 Dependencies Between Views

The dependencies between views influence both the design approaches and the generation stages (see Figure 1). The Domain View does not depend of 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 or a Service. The Architectural View does not depend of any other view. The InteractionSpace View and the NavigationSpace View depend on each other. The InteractionSpace View also depends on the BusinessEntities View, because the Interaction Spaces can be associated to a Business Entity; and on the Architectural View, because some widgets can trigger WebServices detailed on that view.

4.2 Design Approaches

The XIS-Mobile profile leverages the use of model transformations to avoid the manual creation of some of the views mentioned before and also to generate the application source code. Therefore, modeling an application using the XIS-Mobile profile, as in the XIS profile, can be performed by following two approaches: the dummy approach and the smart approach.

In the dummy approach, the developer should define all the views, with the exception of the Architectural View that is only required if the application interacts with external entities, like web servers or other mobile applications. This approach only takes advantage of model-to-code transformations.

On the other hand, in the smart approach (see Figure 1), the developer should define the Domain, BusinessEntities and
UseCases views. Again, like in the dummy approach, the Architectural View is only necessary if interaction with external entities is a requirement. Then, by leveraging model-to-model transformations the InteractionSpace and NavigationSpace views are automatically generated and can be later customized if the developer desires. Therefore, this approach can be much less time-consuming than the dummy approach, as the more complex views are automatically generated. Concluding, the smart approach is strongly recommended, since it speeds up the development process.

5. XIS-MOBILE FRAMEWORK

XIS-Mobile becomes a more relevant language together with a MDD-based framework. As illustrated in Figure 7, the suggested development process of a mobile application using the XIS-Mobile framework comprises four steps: (1) the definition of the model using the Model Editor, (2) its validation through the Model Validator, (3) the generation of the User-Interfaces View models with the Model Generator, and, finally, (4) the generation of native source code. To develop the XIS-Mobile Framework it was decided to use the Model Driven Generation (MDG) Technologies provided by Sparx Systems Enterprise Architect (EA), along with the Eclipse Modeling Framework (EMF), leveraging the environment they provide, as well as some compatible plug-ins.

First, the Visual Editor is implemented on top of EA through the use of an MDG Technology plug-in. This technology is fully compliant with the OMG specification for UML2, and therefore offers a very good support for UML profiles. It not only allows the definition of UML profiles, but also the creation of toolboxes, diagrams or patterns customized to those profiles.

Second, the model validation is another issue that must be taken into account, in order to avoid errors by the user, to improve the quality of the model and, consequently, to enhance the quality of the generated models and code. When thinking about the UML models validation, OCL [24] is the standard language commonly used. Unfortunately, due to several limitations with stereotypes validation and ongoing developments of the OCL plug-in for EMF, the use of OCL wouldn’t be fruitful for now. Therefore, it was chosen to code the validation constraints leveraging the Model Validation API provided by EA. Despite not being a standard like OCL, this solution allows the assignment of severity levels (error or warning) to constraints, the definition of custom error messages and by clicking on the produced error or warning messages, the user immediately navigates to the element that caused it.

Third, the model-to-model transformations are implemented also using the environment provided by EA, namely through its Automation Interface, which allows accessing the previously created diagrams and their elements, as well as creating new diagrams and elements.

Four, the XIS-Mobile Generator performs model-to-text transformations with the guidance of code templates. This generator is based on Acceleo, an Eclipse plug-in. Acceleo is a template-based code generator framework that implements the MOF MTL (Model to Text Language) standard [25] and supports 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 provides preliminary versions for iOS and Windows Phone platforms, but if the user desires to support other platforms, he has to define the respective code templates.

![Fig. 7. Development process with the XIS-Mobile Framework.](image-url)

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 Information Technology. The user session was conducted under the following conditions: tests took place in the laboratory (controlled environment); tasks should be performed without previous use and learning (for the first time); the user must have 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 the XIS-Mobile language, its supporting framework and the overall approach.

---

2 The XIS-Mobile language and framework are supported by a set of artifacts that allow any user to design and develop mobile apps in a platform-independent approach. For more details the reader may consult https://github.com/xis-mobile


4 http://goo.gl/XwUvxs (Accessed on October 2014)

5 http://goo.gl/r9jz0Z (Accessed on October 2014)

6 http://www.eclipse.org/acceleo (Accessed on October 2014)
6.1 Questionnaire Analysis
The questionnaire focused on analyzing the quality of three aspects about XIS-Mobile: (1) the Language, (2) the Framework, and (3) the General 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 can 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 mobile application development domain.

<table>
<thead>
<tr>
<th>Question</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>QL.1</td>
<td>3.67</td>
</tr>
<tr>
<td>QL.2</td>
<td>4.22</td>
</tr>
<tr>
<td>QL.3</td>
<td>3.89</td>
</tr>
<tr>
<td>QL.4</td>
<td>3.56</td>
</tr>
<tr>
<td>QL.5</td>
<td>4.11</td>
</tr>
</tbody>
</table>

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

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 have with different types, tagged values and associations to business entities and providers.

<table>
<thead>
<tr>
<th>Question</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>QF.1</td>
<td>4.44</td>
</tr>
<tr>
<td>QF.2</td>
<td>4</td>
</tr>
<tr>
<td>QF.3</td>
<td>3.33</td>
</tr>
<tr>
<td>QF.4</td>
<td>3.33</td>
</tr>
<tr>
<td>QF.5</td>
<td>3.56</td>
</tr>
</tbody>
</table>

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

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

**QA.1.** How do you rate the productivity with XIS-Mobile comparing to the traditional software development process?

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

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.

<table>
<thead>
<tr>
<th>Question</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>QA.1</td>
<td>4</td>
</tr>
<tr>
<td>QA.2</td>
<td>4.33</td>
</tr>
</tbody>
</table>

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

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 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 [26]. Also, since our questionnaire 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.

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

<table>
<thead>
<tr>
<th>Language</th>
<th>Framework</th>
<th>General Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>XIS-Mobile</td>
<td>3.89</td>
<td>3.73</td>
</tr>
</tbody>
</table>

7. RELATED WORK

Several Cross-Platform Tools for mobile application development, like PhoneGap\(^7\) or Appcelerator Titanium\(^8\), have appeared in the last years and are widely used to overcome the mobile platform fragmentation. Surveys like the ones presented in [27-30] evaluate and compare several of these tools. Mostly, this kind of tools does not reduce the complexity of implementation, because they do not provide a high-level view of the systems. Instead, they only reduce the number of implementations required, i.e., the developer implements the application only once, but that implementation is not necessarily simpler than using a native mobile platform SDK and in some cases does not produce truly native applications. For instance, an application based on PhoneGap is developed using HTML5, JavaScript and CSS3, and relies on the device’s browser capabilities to emulate native features.

MobiCloud [31] 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 backend, 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 take advantage of Cloud Computing.

Unlike XIS-Mobile, MobDSL [15] 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 NavigationSpace and InteractionSpace 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 [32].

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 [33] is more focused on mobile context-aware applications, while MAM-UML [34] targets mobile-agent applications addressing concepts like code mobility.

The Cameleon Reference Framework [35] proposes, like XIS-Mobile, a multi-step development process with different abstraction layers for developing UIs independently of the platform. Therefore, three of its four steps can be compared to XIS-Mobile’s development process: (1) the Tasks & Concepts step has as equivalent the UseCases and Navigation views (for Tasks) and the Domain view (for Concepts); (2) the Abstract UI step corresponds to the InteractionSpace view and (3) the Final UI corresponds to the output of the model-to-code transformations in XIS-Mobile. A lot of work has been done based on the Cameleon Reference Framework, like UisXML [36], IDEALXML [37], UisXML4ALL [38] or MARIA [39] some examples. The UisXML is a XML-compliant language which aims to describe the UI for multiple contexts of use. Like XIS-Mobile, it is decomposed in several models and allows the definition of model-to-model transformations between the different models. In opposition to XIS-Mobile, it is a textual language and focuses either on desktop or mobile applications. IDEALXML complements UisXML by providing a graphical way for specifying it and managing UI patterns. UisXML4ALL 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.

Some other initiatives, namely the Google App Inventor [40], 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. Despite that, Google App Inventor only supports the generation for Android and also has been discontinued by Google and its support was transferred to the MIT Center for Mobile Learning.

Finally, 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 a mobile application. Namely, XIS-Mobile will generate the boilerplate code that represents the great majority of the application’s code. From that point on, the developer has access to the generated code and can customize it if it does not fully fulfill his needs. For instance, he could need to improve the GUI or implement the custom actions attached to a certain widget.

8. CONCLUSION
In this paper, we have presented XIS-Mobile, a domain specific language (define 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. Regarding its multi-view organization, XIS-Mobile comprises six views: Domain, BusinessEntities, Architectural, UseCases, InteractionSpace and NavigationSpace views. Regarding the design approaches, the dummy approach uses model-to-text transformations, while the smart approach uses both model-to-model and model-to-text transformations which contribute to increase the productivity and the quality of the models.

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: (1) defining of the required views using the Visual Editor, (2) validating them using the Model Validator, (3) generating the User-Interfaces View models with the Model Generator, and finally (4) 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 also intend to further evaluate XIS-Mobile, 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, in order 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) [17], context awareness [33] and CMS-based [41] systems.

9. REFERENCES
IEEE International Conf. on Networked Embedded Systems for Enterprise Applications (NESEA) (2010)


