# XIS-Mobile: A DSL for Mobile Applications

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

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

## ABSTRACT
Mobile applications are becoming increasingly more present in our daily life, allowing people to perform several tasks through the use of smartphones or tablets. 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 someone wants to release an application in multiple platforms. Rewriting the application for each platform is usually impracticable either in terms of budget or time, requiring a great effort. Therefore, a solution that could generate cross-platform applications without compromising the quality, would decrease the time to market and increase the number of potential users. This paper presents the XIS-Mobile language, defined as a UML profile in the context of the Sparx Systems Enterprise Architect, and discusses its main challenges and benefits in the context of the cross-platform mobile application development.

## Categories and Subject Descriptors
D.2.2 **[Software Engineering]**: Design Tools and Techniques – *Computer-aided software engineering (CASE)*. D.3.3 [**Programming Languages**]: Language Constructs and Features – *Classes and objects, Frameworks, Modules, packages, Patterns.* H.1.0 **[Models and Principles]**: General. I.6.5 **[Simulation and Modeling]**: Model Development – *Modeling methodologies*.

## General Terms
Design, Languages.

## Keywords
Mobile Application, Model-Driven Development, Cross-Platform, 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][2].


*SAC'14*, March 24-28, 2014, Gyeongju, Korea.

http://dx.doi.org/10.1145/2554850.2554926

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 [3]. 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 [4][5], 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 [6] and shorter time to market [7]. 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 [8].

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 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 discusses the related work. Finally, Section 7 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 [9-12].

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) [13].

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 [14][15][16]. XIS focuses on the design of interactive software systems at a PIM level (Platform-Independent Level, according to MDA terminology) 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 [16][17]. To take full advantage of the smart approach, the designer only needs to design the Domain, BussinessEntities, 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 [17]. 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. Fig.1 depicts the multi-view organization proposed by the XIS-Mobile language to model a mobile application.

[1] http://www.sparxsystems.com.au/

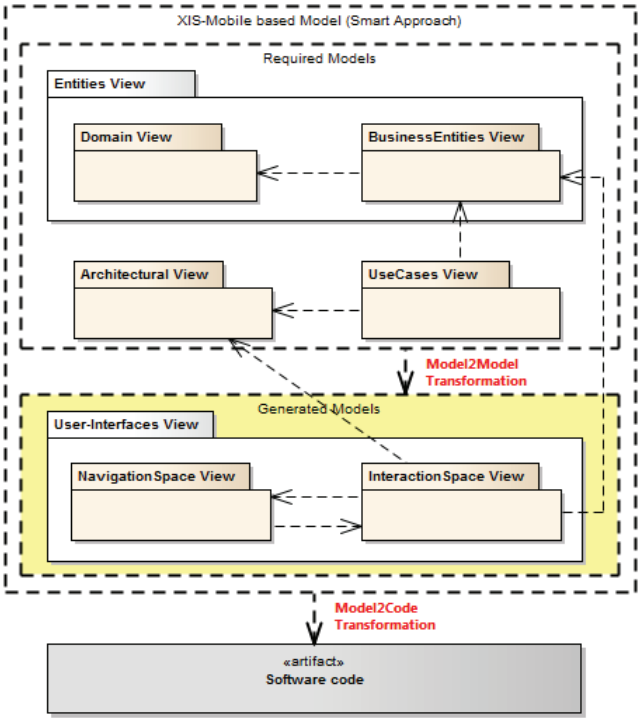


**Fig. 1.** The multi-view organization of XIS-Mobile.

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).

**Case Study – The To-Do App**
The To-Do App consists in an application that manages the tasks a user has to do. Each task has a title, a description and a date of completion. Each task can also contain several notes associated and belong to a certain category.
When a user enters the application, all his tasks are presented. Then, he should be able to manage the tasks: create new tasks, view and edit the details of an existing task or instead delete it from the list. Additionally, the information of every task should be stored persistently.
It should also be possible to synchronize the task information with a remote server and create a calendar alert for each task.

## 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 describes the entities that constitute the problem domain based on a Class Diagram. It is also possible to specify the attributes of the domain entities and the relationships among them using associations, aggregations or inheritances.

Each domain entity is represented by a XisEntity, which in turn contains one or more attributes, XisAttributes. Both these stereotypes are used, instead of just simple UML Classes or Attributes, to provide useful information to the model-to-text transformations through their tagged values.

The To-Do List App's Domain View is composed of three XisEntities: Task, Note and Category (see Fig. 2). The Task contains three XisAttributes that correspond to the title, the description and the date of a task. The Note contains one attribute named description and the Category contains one attribute corresponding to its name.

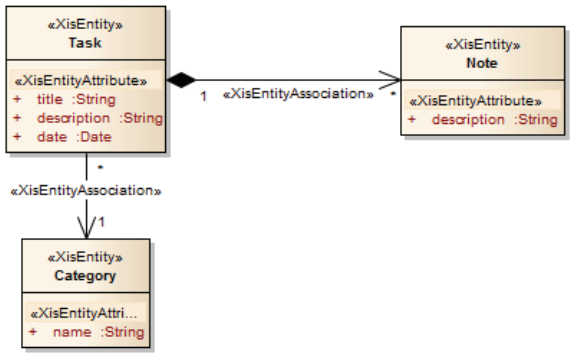


**Fig. 2.** Domain View of the To-Do List App.

### *3.1.2 BusinessEntities View*

The BusinessEntities View represents higher-level entities, called business entities. Each business entity is represented by a XisBusinessEntity that aggregates XisEntities. The goal is to provide context to a certain use case and interaction space, and is useful during the transformation stages.

A XisBusinessEntity defines a master entity (from the Domain View) through a XisMasterAssociation and can also define detail or reference entities through XisDetailAssociations or XisReferenceAssociations, respectively. Both these three associations contain a tagged value named "filter" that allows the restriction of the entitiy's attributes that can be used in the context of the XisBusinessEntity. The definition of a master entity restricts the set of detail and reference entities that can be used by a XisBusinessEntity. Both the detail and the reference entities must be associated to the master entity in the Domain View, through aggregations and associations, respectively.

Considering the To-Do List App, we defined the TaskBE business entity, which has Task as master entity, Note as detail entity and Category as a reference entity.

## 3.2 UseCases View

The UseCases View details the operations an actor can perform when interacting with the application in the context of a business entity and/or by using an external service. These operations are represented as a tagged value inside stereotyped use cases named XisUseCases. The set of operations allowed contains the typical CRUD (Create, Read, Update or Delete) operations and the Search operation. It was decided to include these set of operations, since it contains the most commonly used operations as it was observed in [18].

Additionally, a XisUseCase plays an important role during the model-to-model transformation stages. Namely, according to its type, its set of operations and the business entity or external service is related to. For now, a XisUseCase can have three types: (1) Manage, which will generate interaction spaces that present a list of instances of the master entity (from the associated business entity) and allow their management; (2) Detail, which will

generate an interaction space with the details of a single instance of the master entity (from the business entity); and (3) Service, which will generate interaction spaces that use the operations provided by the associated service. Typically, this last type of use case is used as an extension of the other ones.

In the To-Do List App were defined three use cases (see Fig. 3): (1) "Manage Tasks", of type "Manage" and connected to the TaskBE; (2) "Sync Tasks", of type "Service" used to send the task information to a remote server; and (3) "Create Calendar Task", of type "Service" used to interact with the Calendar Provider in order to create a calendar task.

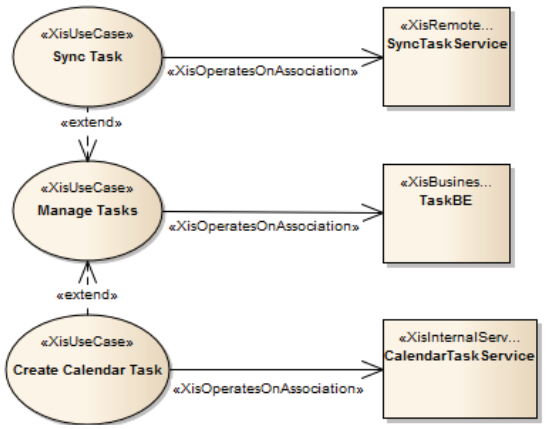


**Fig. 3.** UseCases View of the To-Do List App.

## 3.3 Architectural View

The Architectural View depicts the interactions between the mobile application and other external entities, and so, it can also be known as a "distributed systems view". These interactions are represented as associations, called XisRemoteConnections, which connect the mobile application with XisServices.

A XisService is an interface that supplies operations, called XisServiceMethods. A XisService can be subdivided in two types: (1) XisInternalService, if it is provided by an entity inside the mobile device; and (2) XisRemoteService, if it is provided by a remote entity outside the mobile device. There are four types of entities that realize a XisInternalService: (1) XisLocationProvider, which represents the location provider of the device and gives information about its geographical location; (2) XisContactsProvider, which allows the access to the contacts of the device; (3) XisCalendarProvider, which allows the interaction with the calendar of the device; and (4) XisClientMobileApp, which represents another mobile application running inside the device. In turn, each XisRemoteService is realized by a XisServer that represents physically a web server.

Considering the To-Do List App, it involves two services (see Fig. 3): a XisInternalService provided by a XisCalendarProvider and a XisRemoteService provided by a XisServer.

## 3.4 User-Interfaces View

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

### *3.4.1 NavigationSpace View*

The NavigationSpace View details the navigation flow between the various interaction spaces of the application, known as XisInteractionSpaces. The connection between the interaction spaces is performed through directed associations called XisNavigationAssociations. Each XisNavigationAssociation is also responsible to show the event that triggered the transition between the XisInteractionSpaces.

This view provides a good support for documenting the structure of the system and eases the introduction of future corrections and improvements in the system navigation. For simplicity and readability reasons, the details of each XisInteractionSpaces are not represented in this view, but instead in the InteractionSpace View.

The To-Do List App's NavigationSpace View is composed by four XisInteractionSpaces with ten XisNavigationAssociations between them (see Fig. 6): (1) TaskListIS, which lists all the tasks; (2) TaskEditIS, which displays all the information of each task and allows its edition; (3) TaskCreateIS, which allows the creation of a certain task; and (4) NoteViewIS, which lists all the notes that belong to a certain task.

### *3.4.2 InteractionSpace View*

The InteractionSpace View is perhaps the most complex view of XIS-Mobile due to the amount of abstractions it involves (see Fig. 4). Thus, it is a source of several details about each XisInteractionSpace, 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 transformatios. The modeling of this view is performed through the use of a Composite Class Diagram. The XisInteractionSpace is the main component of this view, since it represents an application screen. Each XisInteractionSpace is depicted as a class and 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 XisDomainAssociation, defining the domain entities that can be bound to the XisInteractionSpace's inner elements.

The XisWidget represents a UI widget or control that builds up the GUI. It is divided in two major categories: XisCompositeWidget and XisSimpleWidget. A XisCompositeWidget is a container class that groups other XisWidgets (simple or composite). As can be noticed, this stereotype follows the well-known Composite design pattern [19]. In turn, the XisSimpleWidget represents the set of simple controls like XisLabel, XisTextBox or XisButton.

Every XisWidget can have many gestures attached, named XisGestures, but at most only one of each type. The set of gestures supported is: Tap, DoubleTap, LongTap, Swipe, Pinch or Stretch. In turn, each gesture can trigger a set of XisActions that ranges from actions like Cancel or WebService to CRUD operations. XisActions can also trigger navigation between Interaction Spaces through the tagged value "navigation" that will have as value the name of the target Interaction Space. XIS-Mobile also gives users the flexibility to define the stub of a Custom operation. 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 (e.g. XisButton have the gesture onTap as default, since it is the most common gesture when interacting with buttons).

Moreover, the value of a XisWidget is defined through the tagged value "value" that can be bound to a domain entity's attribute or can be a constant value. In the first case it assumes the value in the form: <EntityName>.<AttributeName>, and both the entity and the attribute must be available in the Business Entity associated to the InteractionSpace the widget belongs.

Considering the To-Do List App (see Fig. 8), the TaskListIS is the main screen of the application (has the "isMainScreen" tagged value set to true) and is composed of two XisCompositeWidgets: one that represents the task list and other the options menu. The task list contains one XisCompositeWidget of type "Item", named TaskItem, which has "Tap" as the default gesture that triggers the navigation to the TaskEditIS. The menu contains two XisCompositeWidgets of type "Item" that allow the deletion of all tasks of the list or the navigation to the TaskCreateIS.

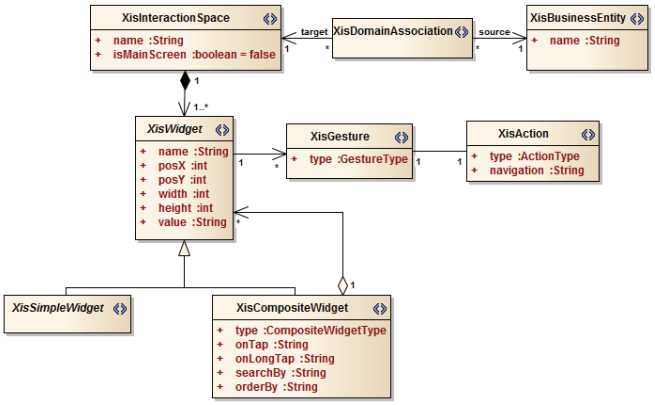


**Fig. 4.** Excerpt of the XIS-Mobile InteractionSpace metamodel.

# 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 Fig. 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 Fig. 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[2]. As illustrated in Fig. 5, 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)[3], 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 [20] is the standard language commonly used. Unfortunately, due to several limitations with stereotypes

[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

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

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[4]. 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[5], 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[6], an Eclipse plug-in. Acceleo is a template-based code generator framework that implements the MOF MTL (Model to Text Language) standard [21] 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 Windows Phone platforms, but if the user desires to support other platforms, he has to define the respective code templates.

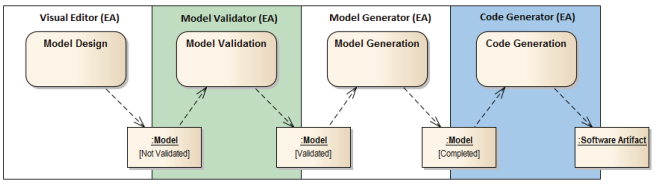


**Fig. 5.** Development process with the XIS-Mobile Framework.

## 6. 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 [22-25] 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 [26] 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 take advantage of Cloud Computing.

Unlike XIS-Mobile, MobDSL [27] 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 [28].

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 [29] is more focused on mobile context-aware applications, while MAM-UML [30] targets mobile-agent applications addressing concepts like code mobility.

The Cameleon Reference Framework [31] 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, being UsiXML [32], IDEALXML [33],

[4] http://goo.gl/XwUvsx
[5] http://goo.gl/r9jz0Z
[6] http://www.eclipse.org/acceleo/
[7] http://phonegap.com
[8] http://www.appcelerator.com

UsiXML4ALL [34] or MARIA [35] some examples. The UsiXML 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 UsiXML by providing a graphical way for specifying it and managing UI patterns. 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.

Some other initiatives, namely the Google App Inventor [36], 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.

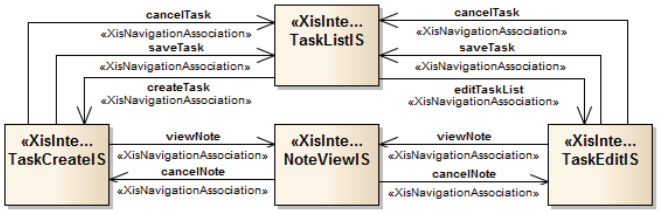


**Fig. 6.** NavigationSpace View of the To-Do List App.

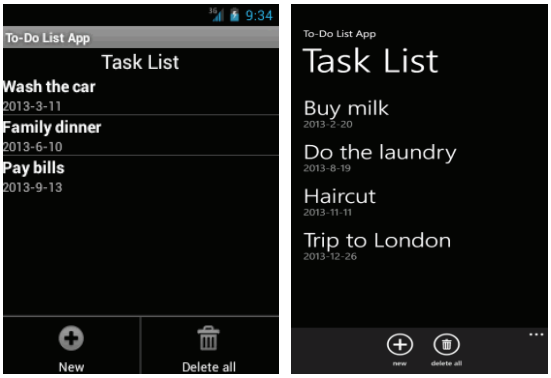


**Fig. 7.** The generated To-Do List App for Android 4.2 Jelly Bean (left) and Windows Phone 7.1 (right).

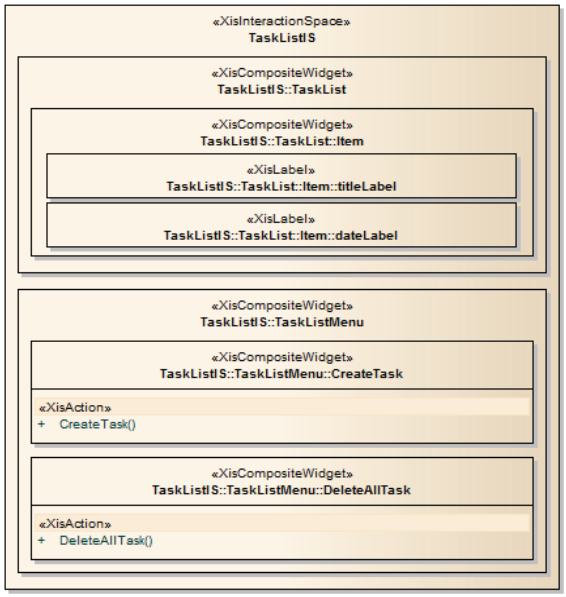


**Fig. 8.** The TaskListIS XisInteractionSpace of the To-Do List App.

## 7. CONCLUSION

This paper presented the XIS-Mobile UML profile as an innovative solution to develop mobile applications and to tackle the problems of software development complexity and mobile platform fragmentation. The use of a multi-view approach adheres to the "separation of concerns" principle and fosters the use of domain concepts, such as widgets or gestures, which represent an advantage allowing the user to concentrate only on main aspects of the application. Therefore, XIS-Mobile comprises six views: Domain, BusinessEntities, UseCases, Architectural, InteractionSpace and NavigationSpace views. Additionally, the use of model transformations to increase the productivity and reduce errors, meets some of the principles proposed by MDD. Along with the XIS-Mobile language it is also proposed an Enterprise Architect MDG and EMF-based framework that intends to generate source code from UML models through model-to-model and model-to-text transformations. The use of this framework can increase the productivity, namely by using a single specification of the system with a Platform-Independent Model and by avoiding the implementation of boilerplate code. For future work, there are plans to add, in the near future, support for other mobile platforms, such as iOS. Also, the integration with other visual editors represents a future step, in order to offer more portability and flexibility to the users. The design and development of more complex and real-world applications (namely for health and tourism domains) 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 (language and framework) to better support scenarios related with the mobile cloud computing (MCC) [13], context awareness [29] and CMS-based [37] systems.

## 8. ACKNOWLEDGMENTS

This work was partially supported by national funds through FCT – Fundação para a Ciência e a Tecnologia, under the PEst-OE/EEI/LA0021/2013 project.

## 9. REFERENCES


[1] Bowman Mitchell, R.J.: *Managing Complexity In Software Engineering*, IEEE Computing Series No. 17 (1990)

[2] Fondement, F., Silaghi, R.: *Defining Model Driven Engineering Processes*, Proc. of the 3rd Workshop in Software Model Engineering (WiSME'04) (2004)

[3] Devitt, S, Meeker, M., Wu, L.: *Internet Trends*, Morgan Stanley Research (2010)

[4] Atkinson, C., Kühne, T.: *Model-Driven Development: A Metamodeling Foundation*, IEEE Software (2003)

[5] Stahl, T., Volter, M.: *Model-Driven Software Development*, Wiley (2005)

[6] Book, M., Beydeda, S., Gruhn, V.: *Model-driven Software Development*, Springer, (2005)

[7] Sendall, S., Kozaczynski, W.: *Model transformation: the heart and soul of model-driven software development*, Software IEEE (2003)

[8] Kuhn, T., Gotzhein, R., Webel, C.: *Model-Driven development with SDL - process, tools, and experiences*, Proc. of the MoDELS'2006 Conference, Springer (2006)

[9] Weiser, M., *Some computer science issues in ubiquitous computing*, Comm. ACM (1993)

[10] Forman, G.H., Zahorjan, J.: *The challenges of mobile computing*, Computer (1994)

[11] Nilsson, E.G.: *Design patterns for user interface for mobile applications*, Advances in Engineering Software (2009)

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

[13] Fernando, N., Loke, S.W., Rahayu, W.: *Mobile cloud computing: A survey*, Future Generation Computer Systems (2013)

[14] Silva, A.R.: *The XIS Approach and Principles*, Proc. of the 29th Conf. on EUROMICRO (EUROMICRO '03), IEEE Computer Society (2003)

[15] Silva, A.R. et al.: *The XIS Generative Programming Techniques*, in Proceedings of the 27th COMPSAC Conference, IEEE Computer Society (2003)

[16] Silva, A.R. et al.: *XIS-UML Profile for eXtreme Modeling Interactive Systems*, Proc. of the 4th International Workshop on Model-Based Methodologies for Pervasive and Embedded Software (MOMPES'07), IEEE Computer Society (2007)

[17] Silva, A.R. et al.: *Integration of RE and MDE paradigms: The ProjectIT Approach and Tools*, IET Software Journal (2007)

[18] Langlands, M.: *Inside The Oval: Use-Case Content Patterns*, Technical report, Planet Project (2010). Accessed on 2013.
http://planetproject.wikidot.com/use-case-content-patterns

[19] Gamma, E. et al.: *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley (1995)

[20] Object Management Group: Object Constraint Language (OCL) Specification. Accessed on December 2013. http://www.omg.org/spec/OCL/

[21] Object Management Group: MOF Model To Text Transformation Language (MOFM2T). Accessed on December 2013. http://www.omg.org/spec/MOFM2T/1.0/

[22] Vision Mobile: *Cross-Platform Developer Tools 2012: Bridging the worlds of mobile apps and the web* (2012)

[23] Singh, I., Palmieri, M.: *Comparison of cross-platform mobile development tools*, IDT: Malardalen University (2011)

[24] Paananen, T.: *Smartphone Cross-Platform Frameworks*, Bachelor's Thesis, Jamk University of Applied Sciences (2011)

[25] Ribeiro, A., Silva, A.R.: *Survey on Cross-Platforms and Languages for Mobile Apps*, Proc. of QUATIC'2012 Conf., IEEE Computer Society (2012)

[26] Ranabahu, A.H. et al.: A Domain Specific Language for Enterprise Grade Cloud-Mobile Hybrid Applications, Proc. of the compilation of the co-located workshops on SPLASH '11, ACM (2011)

[27] Kramer, D., Clark, T., Oussena, S.: *MobDSL: A Domain Specific Language for multiple mobile platform deployment*, IEEE International Conf. on Networked Embedded Systems for Enterprise Applications (NESEA) (2010)

[28] Balagtas-Fernandez, F., Tafelmayer, M., Hussmann, H.: *Mobia Modeler: easing the creation process of mobile applications for non-technical users*, Proc. of the 15th International Conf. on Intelligent UI (IUI '10), ACM (2010)

[29] Boudaa, B. et al., *Model-Driven Development of context-aware services: Issues, techniques and review*, International Conf. on IT and e-Services (ICITeS) (2012)

[30] Belloni, E., Marcos, C.: *MAM-UML: an UML profile for the modeling of mobile-agent applications*, 24th International Conf. of the Chilean, Computer Science Society (2004)

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

[32] Vanderdonckt, J., *A MDA-compliant environment for developing user interfaces of information systems*, Proc. of 17th CAiSE'05 Conf., Springer (2005)

[33] Vanderdonckt, J., Simarro, F.M.: *Generative Pattern-Based Design of User Interfaces*, Proc. of the 1st International Workshop on Pattern-Driven Engineering of Interactive Computing Systems (PEICS '10), ACM (2010)

[34] Trindade, F.M., Pimenta, M.S., *Prototyping Multi-platform Software Using the UsiXML4ALL Tool*, Tech. Report (2007)

[35] Patternò, F., Santoro, C., Spano, L.D.: *MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments*, ACM Trans. Comput.-Hum. Interact. (2009)

[36] Wolber, D.: *App inventor and real-world motivation*, Proc. of the 42ndACM Technical Symposium on Computer science Education (SIGCSE '11), ACM (2011)

[37] Saraiva, J.S., Silva, A.R.*: CMS-based Web-Application Development Using Model-Driven Languages*, Proc. of the ICSEA Conference, IEEE Computer Society (2009)