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

**André Filipe Oliveira Pinto Ribeiro**

Thesis to obtain the Master of Science Degree in

**Information Systems and Computer Engineering**

Supervisor: Prof. Alberto Manuel Rodrigues da Silva

## Examination Committee

Chairperson: Prof. Luís Eduardo Teixeira Rodrigues

Supervisor: Prof. Alberto Manuel Rodrigues da Silva

Member of the Committee: Prof. André Ferreira Ferrão Couto e Vasconcelos

**November 2014**

# Resumo

As aplicações móveis estão a tornar-se cada vez mais presentes nas nossas vidas diárias, permitindo às pessoas realizar várias tarefas através do uso de dispositivos móveis. Apesar de fomentar a inovação, o rápido crescimento do mercado móvel resultou em alguma fragmentação das plataformas móveis. A existência de diferentes sistemas operativos com diferentes linguagens de programação e ferramentas pode ser um problema quando pretendemos desenvolver aplicações para múltiplas plataformas. Reescrever a aplicação para cada plataforma é normalmente impraticável quer em termos de orçamento ou tempo.

Portanto, uma solução que possa gerar aplicações multi-plataforma sem comprometer a qualidade, diminuiria o *time to market* e aumentaria o número de potenciais utilizadores. Felizmente, alguns trabalhos têm sido realizados nos últimos anos para combater este problema, nomeadamente através do uso de tecnologias web, ferramentas multi-plataforma e abordagens baseadas em MDD.

Esta dissertação propõe o uso de uma abordagem MDD para o desenvolvimento de aplicações móveis. Esta abordagem, chamada XIS-Mobile, usa uma linguagem de domínio específico (definida como um perfil UML) e a sua *framework* baseada em MDD para lidar com a fragmentação das plataformas móveis. Propõem a definição de modelos independentes da plataforma para descrever aplicações móveis e a partir deles gerar automaticamente o código-fonte da aplicação para múltiplas plataformas. Esta dissertação apresenta o XIS-Mobile (linguagem e *framework*), fornece uma avaliação deste e discute os seus principais desafios e benefícios no contexto do desenvolvimento de aplicações móveis multi-plataforma.

**Palavras-Chave:** Desenvolvimento Conduzido por Modelos, Multi-Plataforma, Aplicações Móveis, Linguagens de Domínio Específico

# 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 dissertation proposes the use of a MDD approach for the development of mobile application. 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 dissertation 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.

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

# Acknowledgments

There are many people I want to thank for the development of this research work. I believe without their knowledge, expertise or simply motivation, it would not have been possible to complete this important stage of my education.

First, I would specially like to thank Professor Alberto Rodrigues da Silva, who supervised me throughout this work and always believed in me to accomplish the goals we have defined. His guidance, knowledge and support were a great contribution to the development of this work and, therefore, part of its quality is greatly due to him.

I would like to thank my former teammates and colleagues at Indra Sistemas Portugal, whose help, ideas and feedback were very important during the initial development of this work.

I would also like to thank my friends at IST who accompanied me throughout these years and with whom I have learned a lot and shared many moments of fun.

I would also like to thank my colleagues and students of Professor Alberto Rodrigues da Silva and other people who contributed somehow to the development of this work, namely during its evaluation phase.

I believe the outside of this work environment was really important to help me clear my head and gain new energy to pursue this work. Therefore, I owe that to my closest friends and I want to thank them for cheering me up at times when I was more discouraged.

A special thanks to my family, namely to my mother, Paula, my sister and brother, and to my grandparents, Maria da Conceição and Mário, who always believed in me despite the difficulties and always tried to do their best to help me accomplishing this long journey. I cannot even express my gratitude to them in words, but only thank them for their patience during this period, where my availability and good mood were very little. It is to you that I dedicate this work.

Last, but certainly not least, I would like to thank you, Andreia, for your everlasting patience, perseverance and support until the last second of this work. You have been always there for me, specially in the bad times when your encouraging words, unwearying support and availability helped me to find a way. I will never forget your help and for that I consider part of this work is also yours.

# Table of Contents

# List of Figures

# List of Tables

# List of Acronyms

| | |
|---|---|
| **API** | Application Programming Interface |
| **CASE** | Computer-Aided Software Engineering |
| **DSL** | Domain Specific Language |
| **DSM** | Domain Specific Modeling |
| **EA** | Enterprise Architect |
| **EMF** | Eclipse Modeling Framework |
| **GPS** | Global Positioning System |
| **GUI** | Graphical User Interface |
| **MDD** | Model-Driven Development |
| **MDE** | Model-Driven Engineering |
| **MOF** | Meta-Object Facility |
| **OCL** | Object Constraint Language |
| **OMG** | Object Management Group |
| **PIM** | Platform-Independent Model |
| **PSM** | Platform-Specific Model |
| **SDK** | Software Development Kit |
| **TTM** | Time to Market |
| **UI** | User Interface |
| **UML** | Unified Modeling Language |
| **XMI** | XML Metadata Interchange |
| **XML** | eXtensible Markup Language |

# 1. Introduction

Software systems are becoming more complex and sophisticated over the years not just because they try to solve increasingly harder problems, but also due to the size of the projects and the type of technologies involved. How to deal with this complexity is a crucial point that will determine the success and costs of software systems. Fortunately, Software Engineering has been playing an important role in the design, development and maintenance of these systems. Software Engineering allows developers to manage the inherent complexity through the use of methodologies (e.g. Scrum, eXtreme Programming and Rational Unified Process) and approaches (e.g. Waterfall, Prototyping and Iterative) that clearly define the development process, mechanisms of abstraction that encapsulate the complexity (e.g. Object-Oriented programming and UML), and tasks of software verification and quality assurance that control the quality of the software systems (Mitchell, 1990)(Bourguignon)(Fondement & Silaghi, 2004).

Mobile computing is one of the domains in software development that has suffered a greater evolution over the last decade. The emergence of a variety of new mobile devices increasingly more powerful, as well as operating systems with better functionalities (Hartmann, Stead, & DeGani, 2011) (Meeker, Devitt, & Wu, 2010) have made mobile applications 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 the Internet, gestures, GPS, accelerometer, video or audio. These built-in features along with the applications that use them make mobile devices, like smartphones and tablets, very desirable and popular nowadays. All of this resulted primarily from the efforts made by the major mobile industry companies (e.g. Google, Apple and Microsoft) to popularize their respective products. Although the competition between these companies has enabled the rapid growing of the mobile market, it was also responsible for a certain fragmentation of the mobile platforms, since each company provides its own platform with specific language, tools and application market (Allen, Graupera, & Lundrigan, 2010).

The traditional development approach of mobile applications considers source code as the main artifact, while documentation deliverables (e.g. models, requirements or design documents) are considered only as a support artifacts sometimes overlooked. Managing simultaneously the source code and the documentation is usually a time-consuming and error-prone task, since it requires the execution of manual and repetitive tasks by the developers. For this reason and because the main focus is on the source code, is quite common that documentation deliverables are not maintained and kept up to date. An emerging area of Software Engineering, called Model-Driven Engineering (MDE) (Schmidt, 2006), or its development process, Model-Driven Development (MDD) (Book, Beydeda, & Gruhn, 2005)(Sendall & Kozaczynski, 2003), seeks to mitigate this issue by considering models as the main artifact. Other deliverables, such as source code or documentation, are generated automatically from those models through model transformations. Apart from making automatic the repetitive tasks of managing the source code and the documentation, MDD further allows developers to express an application using domain concepts and generate source code for multiple platforms from a single model specification.

## 1.1. Context

This research work has been conducted at the Information Systems Group of INESC-ID (Instituto de Engenharia de Sistemas e Computadores – Investigação e Desenvolvimento) under the supervision of Professor Alberto Rodrigues da Silva, regarding the Master Degree in Information Systems and Computer Engineering at Instituto Superior Técnico. This research work results from the common interest in the area of MDD and its application to mobile application development.

The Information Systems Group was responsible for a variety of projects in the area of Model-Driven Development, namely the ones related to the ProjectIT initiative. ProjectIT (da Silva A. R., 2004)is a research program focused on improving the quality and productivity of Information Technology (IT) projects. ProjectIT defends an approach where the focus of IT projects should be in Project Management (PM), Requirements Engineering (RE) and design activities, while the effort in production activities (e.g. software programming or testing) should be minimized and automated as much as possible by leveraging MDD. To support that, ProjectIT originated a set of languages and tools, namely:

- ProjectIT-RSL and RSLingo(de Almeida Ferreira & da Silva, 2012), Requirements Specification Languages (RSL) to specify and validate requirements of information systems;
- CMS-ML (Saraiva & da Silva, Web-Application Modeling with the CMS-ML Language, 2010), a modeling language for Content Management System-based (CMS) web applications development;
- XIS profile (da Silva A. , Saraiva, Silva, & Martins, 2007), a Domain Specific Language (DSL) for modeling interactive systems at a Platform Independent Model (PIM) level;
- ProjectIT-Studio (da Silva A. , Saraiva, Ferreira, & Silva, 2007), a tool which provides an integrated environment to support tasks ranging from requirements specification, architecture definition, system design and modeling, until code generation.

Therefore, this research work is intrinsically related to the scope of the ProjectIT initiative and can be even encompassed within it. More specifically, the XIS profile served as a starting point for this research due to the similar goal of using a higher level representation, DSL in the form of a UML profile, to design and enhance the development of software applications. However, unlike XIS, this research focuses on mobile applications instead of desktop or web interactive applications. For these reasons, I have named the solution proposed as XIS-Mobile. XIS-Mobile comprises a DSL and a supporting framework for mobile application development. XIS concepts as well as the relation with this work are described in more detail in Section 3.1.

## 1.2. Problem Definition

As introduced previously, the rapid evolution of the mobile market caused the emergence of a mobile platform fragmentation, i.e., heterogeneity of mobile operating systems which have different Software Development Kits (SDKs), languages and supported devices. Platform fragmentation can be an

obstacle particularly when someone wants to develop a mobile application for multiple platforms. Due to the differences between each platform, an application developed for a given operating system is often incompatible with other operating systems. This lack of compatibility forces developers to rewrite the application for each one of the target mobile platforms implying many redundant work, increased development complexity and consequently higher costs of production and larger time to market (TTM).

Thus the main problems addressed in this dissertation are: Software Development Complexity and Platform Fragmentation. Given these problems, the following research questions summarize the situation to address:

**RQ.1:** How to specify a mobile application in a platform-independent way?

**RQ.2:** Which concepts are specific of mobile applications?

**RQ.3:** Which views should be used to specify a mobile application?

**RQ.4:** Which User Interface patterns can be used to specify a mobile application?

**RQ.5:** How to generate code for multiple mobile platforms from a single specification?

The ideal scenario would be to develop a mobile application once and then run it in as many platforms as desired achieving what is known as portability. Providentially some work has been conducted over the last years to tackle both these problems. The use of web technologies (e.g. HTML5, CSS3 and JavaScript libraries), cross-platform tools and frameworks, or approaches based on MDD (Fondement & Silaghi, 2004), like the one proposed in this dissertation, are examples of solutions focused on solving these problems.

# 1.3. Proposed Solution

This research proposes XIS-Mobile, a Domain Specific Language and its supporting framework, as an innovative solution to the problems presented previously, through a MDD approach.

The XIS-Mobile language is a graphical DSL in the form of a UML profile that allows modeling mobile applications in a platform-independent way using concepts specific of the mobile applications domain. The XIS-Mobile language has a multi-view organization and supports two design approaches: the dummy approach and the smart approach. Shortly, the main difference between this two design approaches is that the smart approach uses model-to-model transformations to automatically generate the more time-consuming views (the User-Interface View package), while the dummy approach implies the manual creation of all views.

The use of this language has been materialized through the implementation of a supporting MDD-based framework developed using available, widely used and well-known technologies, such as the Sparx Systems Enterprise Architect (EA)[1] and the Eclipse Modeling Framework (EMF)[2]. As illustrated in Figure 1, the XIS-Mobile framework comprises four major modules: (1) Visual Editor, allows the

[1] http://www.sparxsystems.com.au/products/ea/index.html (Accessed on October 2014)
[2] http://www.eclipse.org/modeling/emf (Accessed on October 2014)

definition of the mobile application using the XIS-Mobile language; (2) Model Validator, checks if the models produced using the Visual Editor are valid; (3) Model Generator, generates more complex models from the previous defined models, through Model-to-Model (M2M) transformations; and (4) Code Generator, generates the native source code of the defined mobile application for multiple platforms, through Model-to-Text (M2T) transformations based on code templates specified using Acceleo[3], a plug-in compatible with the EMF.

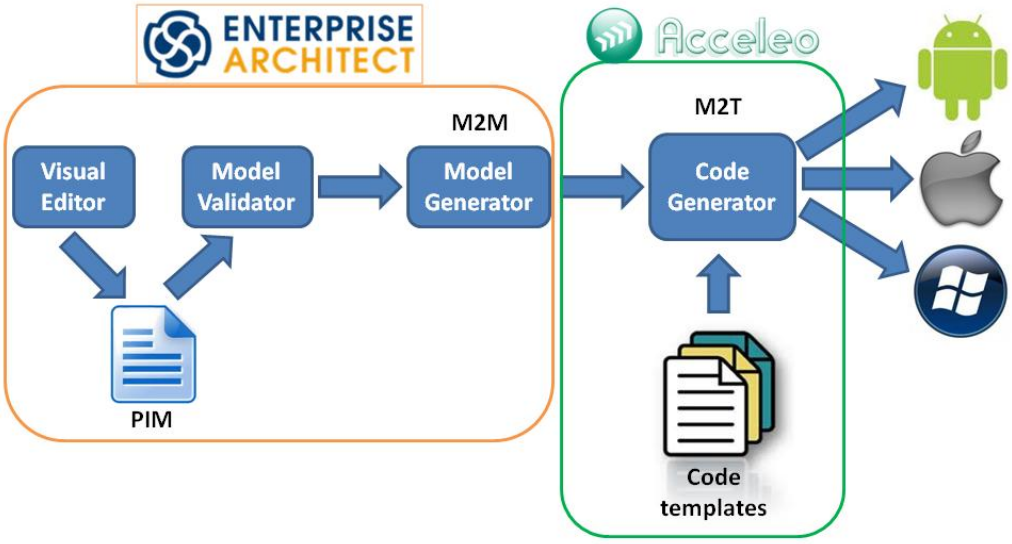



**Figure 1.** Simplified development approach with the XIS-Mobile framework.

It is important to emphasize that XIS-Mobile does not intend to replace the role of a developer, but instead represent a helpful tool that generates the skeleton of mobile applications. Namely, XIS-Mobile helps the developer by generating the repetitive and boilerplate code which represents a great percentage of the application's code. Then, the developer can modify the generated code if it does not fulfill all his needs. For instance, he could need to improve the User Interface or implement some custom actions attached to certain widgets. XIS-Mobile is suitable for a multitude of mobile apps that are typically business (and form) oriented, e.g., that allow consulting and submitting data; but are not intended for very specific domains such as games.

After presenting the proposed solution, it is important to clearly highlight the goals of this dissertation. So this work aims to:

- **G1:** Collect and analyze the related work done in the scope of cross-platform development of mobile applications, mainly tools and languages;
- **G2:** Design and implement a Domain Specific Language that allows the development of mobile applications for different types of platforms (e.g., Android and Windows Phone);
- **G3:** Implement the tools that support that language, namely:
  - A visual editor that allows the specification of the application using a UML profile;
  - A generator that parses models specified in this language and generates automatically native code for multiple platforms.

[3] https://www.eclipse.org/acceleo (Accessed on October 2014)

**G4:** Evaluate the quality of the proposed system through the implementation of case study mobile applications.

## 1.4. Thesis Statement

This dissertation's thesis states that is possible to mitigate both the software development complexity and the mobile platform fragmentation during the development of mobile applications, by means of a MDD approach.

In particular, I claim that this can be achieved relying on the development of Platform-Independent Models (PIM) using the XIS-Mobile language, and their subsequent transformation using the XIS-Mobile framework. Initially these models are transformed in more complex models (through Model-to-Model transformations) and then in native source code of the corresponding mobile application (through Model-to-Text transformations). This approach generates a great percentage of the final application source code for multiple platforms from a single specification. Therefore, I also claim that the use of XIS-Mobile contributes to an improvement of productivity.

Furthermore, it was possible to develop at least three case study applications using the proposed approach, from model until source code. The conduction of a user session also contributed to assess the acceptance and to aware the participants to the enhancements in productivity this approach can offer.

## 1.5. Methodology

This research work has been conducted in an iterative and gradual way following the Action Research methodology (Baskerville, 1999). As illustrated in Figure 2, this methodology suggests a cyclical process composed of five steps executed in a certain scope, known as client-system infrastructure (or research environment). These steps are:

**(1) Diagnosing –** Represents the identification of the problem domain and the motivation for its relevance;
**(2) Action Planning –** Represents the planning and definition of the proposed solution and the necessary changes to solve/relieve the problem identified in the previous step;
**(3) Action Taking –** Consists in the implementation of the solution planned in the previous step;
**(4) Evaluating –** Represents the evaluation/assessment of the solution developed in the previous step, namely analyzing if the actions performed have succeeded to solve/relieve the problem.
**(5) Specifying Learning –** Represents the lessons learned during the cycle, and in the preparation of the next iteration of the action research cycle if needed.

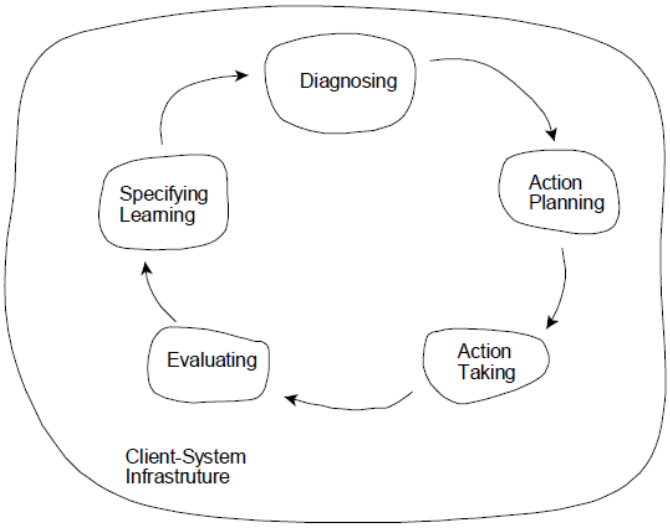



**Figure 2.** Action Research Methodology cycle **(Baskerville, 1999)**.

Particularly, this research work can be subdivided in four iterations of the Action Research methodology.

**Iteration 1 – Month 1 to Month 6:**

At an early stage, the first iteration mostly consisted in the review of the related literature and work, specially in the fields of MDE, mobile computing and cross-platform mobile application development. This review task resulted in the publication of a paper regarding the analysis of six cross-platform tools and languages for mobile application development (Ribeiro & da Silva, Survey on Cross-Platforms and Languages for Mobile Apps, 2013). Then, there were explored the available technologies that could support the definition and use of the XIS-Mobile profile. The choice fell on the environment provided by the Eclipse IDE and two plugins, Papyrus and Acceleo. After this analysis, it was defined a preliminary version of the XIS-Mobile language and implemented a preliminary version of the XIS-mobile framework. At last, a case study application was defined, which was then used to evaluate the language and framework previously designed. Next, the five steps of the Action Research methodology for this first iteration are summarized.

**Diagnosing –** Review of the existing literature and technologies regarding the definition of the XIS-Mobile language and framework.

**Action Planning –** Definition of the organization and concepts of a preliminary version of the XIS-Mobile language. Definition of the architecture and components of a preliminary version of the XIS-Mobile framework.

**Action Taking –** Specification of the XIS-Mobile language as a UML profile using the Papyrus plugin for the Eclipse IDE. Implementation of the XIS-Mobile framework using Papyrus for specifying the XIS-Mobile language and Acceleo for defining the code templates for Android.

**Evaluating –** Definition of a case study application: “To-Do List App” and its manual implementation for the Android platform. Implementation of the case study application using the XIS-Mobile framework.

**Specifying Learning –** Success on modeling the case study application with the XIS-Mobile language, using Papyrus. Then, Acceleo offered an excellent support to perform the Model-to-Text transformations and allowed generating great part of the case study application’s source code from the model specification. However, Papyrus proved poorly suited to model with a language of the complexity of XIS-Mobile, namely revealed usability problems and bugs that hinder the modeling process. Need to add new concepts to the XIS-Mobile language in order to fully capture the functionality of the case study application and simplify the modeling process. Need to improve the M2T templates in order to generate code with a quality equivalent to handwritten code. Need to implement code templates for other platforms besides Android to prove that cross-platform mobile application development can be done. Need to add support for M2M transformations, in order to speed up the modeling process through the generation of the most laborious views. Need to add support for model validation to the XIS-Mobile framework, in order to assure the correctness of the produced models.

**Iteration 2 – Month 7 to Month 12:**

Initially, the second iteration consisted in the correction and reorganization of the XIS-Mobile language. Then, the XIS-Mobile framework was subjected to major changes regarding the issues found in the previous iteration. As Papyrus did not provide the desired user experience, it was decided to migrate from the Eclipse environment to a more sophisticated and refined environment, Sparx Enterprise Architect (EA). Additionally to that, model validation and M2M transformations have been implemented to enhance the quality of the development process using the framework. After that, the code templates for Android have been improved and a preliminary version of the code generator for Windows Phone has been developed. At last, the case study has been modeled using the revised version of the XIS-Mobile language in the EA environment. This iteration originated the publication of a paper describing XIS-Mobile (language and framework) and the results obtained from the implementation of the case study for different platforms (Ribeiro & da Silva, XIS-Mobile: A DSL for Mobile Applications, 2014). Next, the five steps of the Action Research methodology are summarized for this iteration.

**Diagnosing –** Analysis of the issues and needs discovered in the previous iteration. Research about alternative technologies to Papyrus that provide the desired user experience and also support model validation and M2M transformations.

**Action Planning –** Definition of the concepts that should be fixed and added to the XIS-Mobile language, namely the ones that allow the M2M transformations. Review of XIS-Mobile framework’s architecture, specifically through the change to EA’s environment and addition of the model validation and model generation components. Planning of how M2M transformations should be performed. Definition of the rules that should be triggered during model validation.

**Action Taking –** Reorganization and correction of the XIS-Mobile language using EA. Implementation of the model validation and model generation features of the XIS-Mobile framework. Implementation of a preliminary version of the code generator for Windows Phone.

**Evaluating –** Modeling of the case study using the revised version of the XIS-Mobile language and testing the model validation and model generation features. Generation of the case study for Android and Windows Phone through the XIS-Mobile framework.

**Specifying Learning –** Success on modeling the case study application with the XIS-Mobile language, using EA. Success on validating and generating part of the models, and on generating source code for Android and Windows Phone using the revised XIS-Mobile framework. EA provided a very good environment to use XIS-Mobile. Need to define new case studies that exercise other concepts of the XIS-Mobile framework. Need to implement M2T generation for the iOS platform.

**Iteration 3 – Month 13 to 16:**

Initially, the third iteration consisted in the definition and implementation of a second case study application, for Android and iOS, focused on tourism. After that, a preliminary version of the code generator for iOS has been developed in the work by Mats Sandvoll (Sandvoll, 2014). Then, the XIS-Mobile language was subjected to minor changes required to properly model this new case study and to assure the consistency of the concepts nomenclature. At last, the models for this new case study have been created and the corresponding source code for Android and iOS has been generated. Next, the five steps of the Action Research methodology for this iteration are summarized.

**Diagnosing –** Test of some mobile applications for tourism and research about the iOS platform.

**Action Planning –** Definition of a new case study application focused on tourism.

**Action Taking –** Implementation of the new case study application for the Android and iOS platforms. Minor changes of the XIS-Mobile language to properly model this case study and to assure the consistency of the concepts nomenclature. Implementation of a preliminary version of the code generator for iOS.

**Evaluating –** Modeling of the case study using the XIS-Mobile language and generation of its corresponding source code for Android and iOS.

**Specifying Learning –** Success on modeling the new case study application with the XIS-Mobile language. Success on generating source code for Android and iOS using the XIS-Mobile framework. The language provides the necessary concepts to model this new case study. The framework supports well the proposed approach. However, XIS-Mobile needs to be used and evaluated by others, i.e., people not directly involved in its development.

**Iteration 4 – Month 17 to 19:**

The last iteration essentially consisted in the evaluation of XIS-Mobile by third parties. First, it was defined how the evaluation would occur and the aspects to be analyzed. The evaluation has been

performed by conducting a user session and focused on three aspects: Language, Framework and general proposed approach. For that, a third case study has been defined which should be modeled by the users during the session. In addition, a questionnaire has been created with the aim of gathering the opinions of the participants at the end of the evaluation session. After the session, an analysis of the results obtained from the questionnaire, as well as the difficulties expressed by the participants during the session, has been carried out. This analysis originated the publication of a paper (Ribeiro & da Silva, 2014). Next, the five steps of the Action Research methodology for this iteration are summarized.

**Diagnosing –** Identification of the needs of XIS-Mobile in terms of evaluation.

**Action Planning –** Definition of the aspects to be evaluated. Definition of how the evaluation session would occur and the type of participants involved.

**Action Taking –** Specification of the user session guide, including the new case study to be modeled and the session rules. Elaboration of the participation questionnaire.

**Evaluating –** Realization of the evaluation session with users.

**Specifying Learning –** Analysis of the results obtained from the questionnaire, as well as the difficulties expressed by the participants during the session. The questionnaires collected positive results and showed XIS-Mobile's usefulness and feasibility as proof of concept. As future research directions, there are plans to add more patterns of M2M generation, to conduct usability tests in the academic field and/or for teaching purposes, to improve the representation of the models to make them more appealing and explore other areas, like mobile cloud computing (MCC) (Fernando, Loke, & Rahayu, 2013), context awareness (Boudaa, Camp, Hammoudi, & Chikh, 2012) and CMS-based (Saraiva & da Silva, 2009) systems, where the XIS-Mobile approach can also be applied.

# 1.6. Publications

During this research the following papers were published in international conferences and journals, with peer-reviewing:

1. André Ribeiro and Alberto Rodrigues da Silva. 2012. **Survey on Cross-Platforms and Languages for Mobile Apps**. Proceedings of the 2012 Eighth International Conference on the Quality of Information and Communications Technology (QUATIC '12). IEEE Computer Society, Washington, DC, USA, 255-260.
   This paper describes and analyzes six cross-platform tools and languages to develop mobile applications.
2. André Ribeiro and Alberto Rodrigues da Silva. 2014. **XIS-Mobile: a DSL for Mobile Applications**. Proceedings of the 29th Annual ACM Symposium on Applied Computing (SAC '14). ACM, New York, NY, USA, 1316-1323.
   This paper describes the first version of the XIS-Mobile language, namely its goals, multi-view organization, design approaches and main components, as well as the XIS-Mobile

framework, through the use of a case study application: “The To-Do List App”. Part of the content of this paper is contained in Section 6.1 of this dissertation.

3. André Ribeiro and Alberto Rodrigues da Silva. 2014. **Evaluation of XIS-Mobile, a DSL for Mobile Applications**. Journal of Software Engineering and Applications. Scientific Research Publishing.
   This paper describes the second version of XIS-Mobile (language and framework), and presents the results of its evaluation through the realization of a user session. Part of the content of this paper is contained in Section 6.2 of this dissertation.

## 1.7. Outline

The remainder of this dissertation is organized as follows:

**Chapter 2 -** This chapter provides an overview of the main concepts that underlie this research work. Particularly focusing on two main topics: Model-Driven Development (MDD) and Mobile Application Development.

**Chapter 3 -** This chapter presents an overview of the approach proposed in this dissertation, known as XIS-Mobile approach.

**Chapter 4 -** This chapter presents the XIS-Mobile language, a DSL that allows designing mobile applications in a platform-independent way.

**Chapter 5 -** This chapter describes the XIS-Mobile framework, i.e., the integrated environment that supports the Model-Driven Development (MDD) of mobile applications using the XIS-Mobile language.

**Chapter 6 -** This chapter presents and discusses a threefold evaluation performed to XIS-Mobile.

**Chapter 7 -** This chapter presents the main conclusions of this work along with the future work perspectives.

# 2. Background

This chapter provides an overview of the main concepts that underlie this research work. Particularly, focusing on two main topics: Model-Driven Development (MDD) and Mobile Application Development.

First, Section 2.1 describes the basic concepts around the MDD field. Namely, Section 2.1 describes MDD, Model-Driven Engineering (MDE) and related initiatives, explains concepts such as metamodeling, modeling languages, domain specific language and UML profile; and details the types of model transformations used in MDD.

Then, Section 2.2 describes the basic concepts around the Mobile Application Development field. It presents definitions for concepts like mobile computing and mobile device, describes the types of mobile applications, mobile device gestures and the most used mobile platforms

Section 2.3 presents some Mobile UI design patterns considered in this research work. Section 2.4 describes specific issues of mobile applications derived from a domain analysis that was performed in this area. At last, Section 2.5 explains the concept of cross-platform tool for mobile application development and compares some of these tools.

## 2.1. Basics on Model-Driven Development

This section presents some concepts around the Model-Driven Development (MDD) area. Namely, Section 2.1.1 describes what is Model-Driven Engineering (MDE) and Model-Driven Development (MDD), while Section 2.1.2 and Section 2.1.3 present two initiatives of MDD. Then, Section 2.1.4 clarifies the concepts of metamodeling and modeling language. In turn, Section 2.1.5 and Section 2.1.6 define the concepts of domain specific language (DSL) and UML profile, respectively. In Section 2.1.7 are explained the model transformations commonly used in MDD. At last, Section 2.1.8 explains the concept of language workbench.

### 2.1.1. Model-Driven Engineering

Model-Driven Engineering (MDE) (Schmidt, 2006) is a software development methodology that considers domain models as first-class entities. By recurring to those models, MDE seeks to move the usual source code development process to a more abstract level of specification. Domain models consist in abstract representations of concepts specific of a certain domain problem. A model can even be classified as descriptive or prescriptive, according to the use that is made of it. A descriptive model is only used as documentation of the system it details, while a prescriptive model besides describing the system, it is also used to develop the system (Gonzalez-Perez & Henderson-Sellers, 2007). Thus, using this terminology, MDE defends the use of prescriptive models.

The main goal of MDE is that the models guide all the development activities, from system design, code generation and deployment until the system maintenance. This abstraction results in advantages like quality improvements, increased productivity and improved communication with domain experts,

as well as programmers (Book, Beydeda, & Gruhn, 2005). The use of concepts closer to the domain problem also tries to reduce the time to market (TTM) (Sendall & Kozaczynski, 2003). Other of the greatest benefits of MDE 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 (Kuhn, Gotzhein, & Webel, 2006).

Therefore, MDE is a software engineering discipline, because it is concerned with other model-based activities of a complete software engineering process, beyond the development tasks. For instance, model-driven reverse engineering and model-driven evolution are examples of such activities. Thus, it becomes clear that Model-Driven Development (MDD) is a subset of MDE, because it only encompasses the model-based development activities, i.e., the generation of the system through models. Additionally, MDD is materialized through some initiatives. Two of the most popular MDD initiatives are the Model Driven Architecture (MDA), proposed by the Object Management Group (OMG), and the Software Factories, proposed by Microsoft. They are briefly described below in sections 2.1.2 and 2.1.3, respectively.

Concluding, Figure 3 summarizes the range of action of MDE, MDD, MDA and Software Factories.

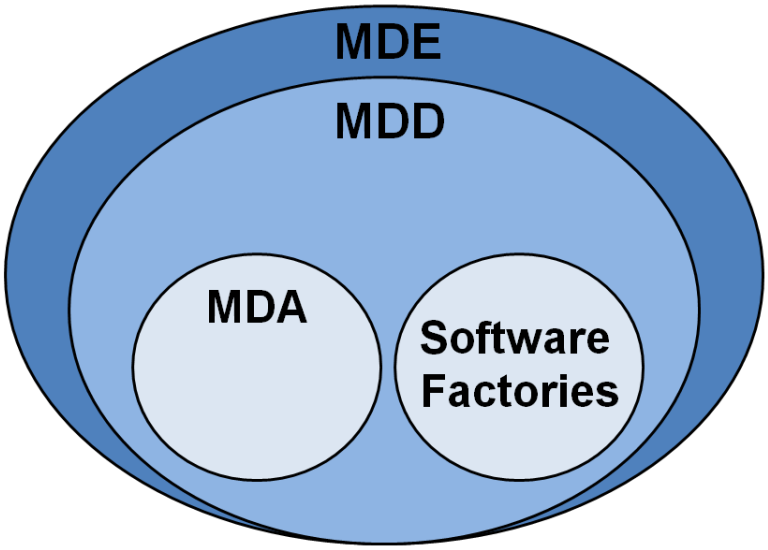



**Figure 3.** Representation of the Model-Driven initiatives (adapted from **(Ameller, 2009)**).

## 2.1.2. Model Driven Architecture

Model Driven Architecture (MDA) is the initiative proposed by the Object Management Group (OMG) to develop software through a MDD approach. MDA provides a set of guidelines and principles to specify a system based on models. These models can reside at different levels of abstraction, each one emphasizing a certain aspect of the system. MDA considers two main types of models: (1) the Platform-Independent Model (PIM) and (2) the Platform-Specific Model (PSM). A PIM is a specification of a system with a high level of abstraction expressed in a platform-independent way, i.e., it is a model independent of the technology used. A PSM is also a specification of a system, but in platform-specific way, i.e., a PSM specifies how a system uses a concrete type of platform. Then, a PIM is translated to

one or more PSMs through the use of Model-to-Model (M2M) transformations which map the PIM with some implementation language or platform (e.g., Java or C#) according to predefined rules. The last step consists in the transformation of each PSM in source code of the respective platform. These transformations are designated by Model-to-Text (M2T) transformations.

The goal of MDA is to provide system specification and interoperability providing a description of a system in a platform independent way (OMG, Object Management Group - MDA Guide, Version 1.0.1, 2003)(Frankel, 2003)(Kleppe, Warmer, & Bast, 2003).

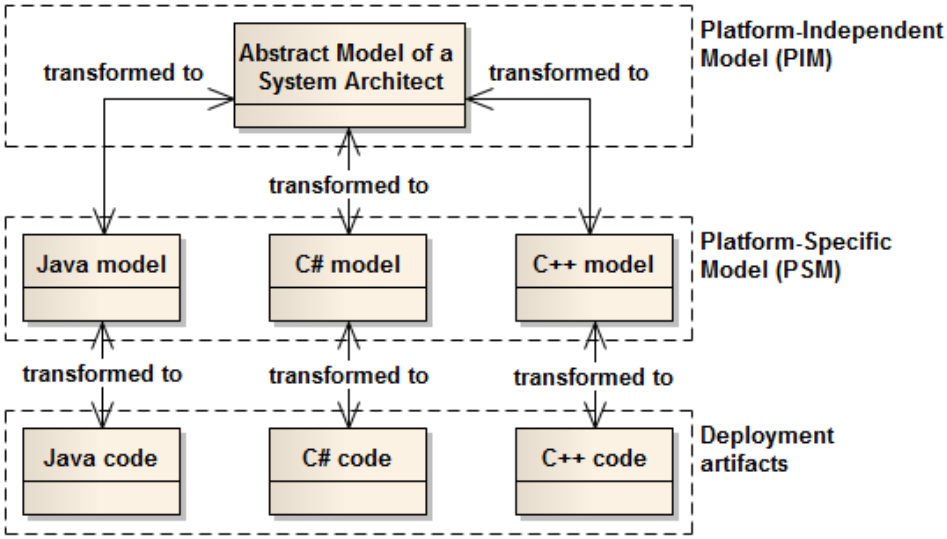



**Figure 4.** Overview of the MDA approach (adapted from **(Saraiva & da Silva, 2008)**).

## 2.1.3. Microsoft Software Factories

Microsoft Software Factory is a product line, or a collection of software, used to create specific kinds of applications. It may be composed of processes, templates, Integrated Development Environment (IDE) configurations and views.

A Software Factory contains three main ideas: a Software Factory Schema, a Software Factory Template and an Extensible Development Environment. The Schema is like a “recipe” that describes the product line architecture and the key relationships between the components and frameworks of which it is comprised. The Template provides elements, such as DSLs, patterns, templates, editors, frameworks or samples, used to build the final product. Finally, the Extensible Development Environment becomes a software factory when it uses the configurations defined in the Template (Greenfield & Short, 2004). Unlike MDA, Software Factories are not so worried with portability and platform independence. Instead, they are more focused in productivity with the goal of reducing the costs and TTM.

## 2.1.4. Modeling language and metamodeling

The models used in MDD can have different representations and encompass different domain concepts, but all of them are specified through a modeling language. A modeling language is defined

by three main components: (1) the abstract syntax, (2) the concrete syntax and (3) the semantics. The abstract syntax specifies the set of concepts provided by the modeling language and the relationships among themselves. Thus, the abstract syntax is a model itself, the metamodel. This means that a metamodel provides a language in which the model is specified. The concrete syntax, also known as notation, defines the representation of the modeling language's concepts. This representation can be either textual and/or visual. Semantics describe the meaning of each concept defined in the abstract syntax that could not be captured by the metamodel. Semantics is often specified through the definition of rules that restrict the possibilities of use between each language element, in order to prevent its invalid use. These rules can be specified using a rule specification language, like OCL (OMG, Object Management Group - Object Constraint Language (OCL) Specification, Version 2.4, 2014), and using a natural language specification that is more easily and quickly understood by model designers.

Therefore, metamodeling is a crucial activity on MDD, because it defines the metamodel used to create modeling languages. Figure 5 summarizes the relationship between these concepts.

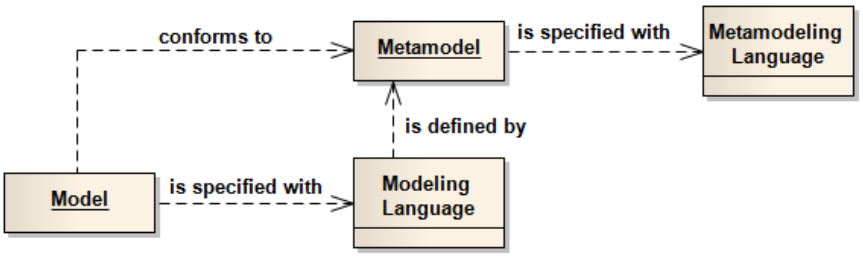



**Figure 5.** The relationship between a model and a metamodel (adapted from **(Saraiva & da Silva, 2008)**).

A modeling language can be classified as a General-Purpose Modeling Language (GPML) or a Domain Specific Modeling Language (DSL/DSML). The use of GPMLs or DSMLs is a recurrent topic of discussion in the area of MDE. A GPML is characterized by having a greater number of generic constructs, which encourages a wider and widespread use in different fields of application. The Unified Modeling Language (UML)[4](OMG, 2011) specified by the OMG, is an example of that kind of modeling language which provides a large set of constructs and notations used primarily for specifying and documenting software systems according to the object-oriented paradigm. On the other hand, DSLs provide a better description in the domains where they are applied. DSLs use constructs that are closer to the most significant concepts of its application domain. Therefore, by using DSLs it is possible to define models that capture more details of the domain problem and do it simultaneously in a more expressive way. However, UML also provides the Profile mechanism that allows extending its concepts (metaclasses) and adapt them to a specific domain problem. The DSL and UML Profile topics are described in more detail in sections 2.1.5 and 2.1.6, respectively.

Several metamodeling approaches are available nowadays, namely because MDD does not obliges the use of any approach in specific (Hennig, Braune, & Koycheva, 2010). Nevertheless, the use of standardized approaches eases and fosters the adoption of the modeling language to be developed.

[4] http://www.uml.org/ (Accessed on October 2014)

The most popular metamodeling approach is the Meta Object Facility (MOF)(OMG, 2014) proposed by the OMG. MOF has been conceived with the goal of providing a metamodel in which OMG's standards could be based. UML, XML Metadata Interchange (XMI) (OMG, 2014) and MOF Model to Text Transformation Language (MOFM2T)(OMG, 2008) are some examples of MOF-based languages. MOF comprises two variants: Essential MOF (EMOF) and Complete MOF (CMOF). EMOF is a subset of MOF that provides the facilities found in object-oriented programming languages and XML. For instance, EMOF is the metamodel of XMI. In turn, CMOF is the result of merging EMOF with its extensions, i.e., CMOF provides the concept of Association. CMOF is the metamodel used to specify UML2.

As shown in Figure 6, MOF is designed as a four-layered metamodel architecture where each layer conforms to the one above it:

The M3 layer corresponds to the meta-metamodel layer, in which MOF is the meta-metamodel, i.e., MOF is defined by itself.

The M2 layer corresponds to the metamodel layer, in which the metamodel is an instance of MOF. For instance, the most commonly used is the UML metamodel.

The M1 layer corresponds to the model layer, in which the metamodel concepts are used to define the user model. Considering that UML was the metamodel defined in the M2 layer, this layer uses UML concepts (e.g. Class, Attribute or Operation) to define the model.

At last, the M0 layer corresponds to the system, in which runtime instances are defined using the model elements specified in the M1 layer.

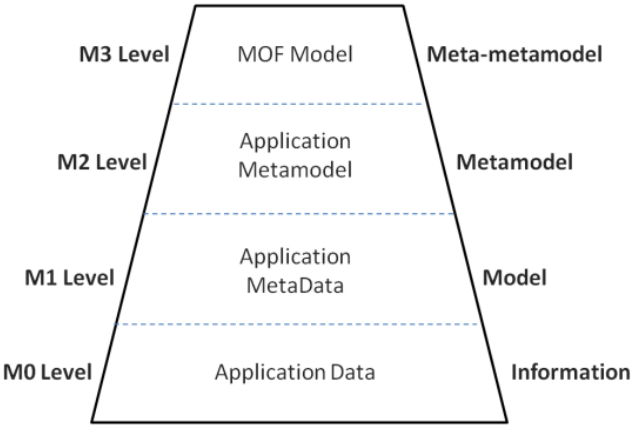



**Figure 6.** OMG's MOF four-layered metamodel architecture **(Cetinkaya & Verbraeck, 2011)**.

As an OMG's standard, MDA highly relies on MOF to define its models (PIMs and PSMs). Namely, UML is commonly used for specifying the PIM. This fact guarantees that the models can be stored in a MOF-compliant repository, parsed and transformed by MOF-compliant tools, and rendered into XMI for transport over a network (OMG, 2003).

Additionally, another variant of MOF, named Ecore[5], has been defined in the context of the Eclipse Modeling Framework (EMF). Ecore is similar to the EMOF and is used to represent models and their runtime support in EMF (Steinberg, Budinsky, Paternostro, & Merks, 2008). For instance, the UML implementation used in Eclipse is defined by Ecore.

### 2.1.5. Domain Specific Language

A Domain Specific Language (DSL) is a language tailored for a specific set of tasks which, through appropriate notations and abstractions (either textual or graphical), depicts concepts of a particular problem domain (Fondement & Silaghi, 2004). Particularly, Domain Specific Modeling Languages (DSMLs), like the one proposed in this dissertation, abstract even more the problem domain, since they usually represent their concepts in a graphical way. DSLs are usually smaller than the usual programming languages and can offer several advantages.

Since a DSL is expressed using domain concepts, it is normally easier to read, understand, validate and communicate with, facilitating cooperation between programmers and domain experts. Moreover, in the literature is mentioned that DSLs can improve productivity, reliability, maintainability and portability (van Deursen, Klint, & Visser, 2000)(Kieburtz, McKinney, & Bell, 1996). However, the use of a DSL can raise some problems, such as the cost of learning, implementing and maintaining a new language, as well as the support tools to develop with it. Some popular examples of DSLs are SQL for databases or HTML for web pages.

### 2.1.6. UML Profile

A UML profile (OMG, 2011) is an extension mechanism provided by UML for customizing UML models for a particular domain purpose. A UML profile does not allow the direct edition of the UML metamodel, but instead allows extending it with new concepts, in order to make it more specific to a given problem domain. For this reason, the UML profile mechanism is called a “lightweight approach”. UML profiles are defined using the following elements:

- **Stereotypes –** A stereotype is a specific metaclass that extends UML metaclasses, i.e., a new element that extends the elements defined in the UML metamodel. For instance, a stereotype extends metaclasses like Class, Attribute or Operation. A stereotype can also define a custom appearance or representation of the element it defines;
- **Tagged Values –** A tagged value is a meta-attribute contained by a stereotype. Essentially, tagged values are equivalent to Class attributes and offer more expressiveness to stereotypes, because they add more information to them;
- **Constraints –** A constraint is a restriction or rule required in a particular domain that restricts the use and the relationships between stereotypes. These constraints cannot weaken the ones specified by the UML metamodel. Constraints can be defined in OCL.

[5] http://goo.gl/3Lqx3H (Accessed on October 2014)

UML profiles provide a straightforward way to produce DSMLs, since UML is widely known and the UML profile mechanism is supported by the majority of UML tools. Therefore, existent MDD technologies based on UML can also be reused to use UML profiles thereby reducing the cost of implementation. However, since UML profiles are extensions of UML, the model designer must deal with an infrastructure ready for all UML concepts. Namely, he must be aware in order to not mix domain-specific constructs (defined in UML profiles) with predefined UML constructs.

Some examples of UML profiles are MARTE (Modeling and Analysis of Real-Time and Embedded systems), used for specifying real-time and embedded applications, and BPMN (Business Process Modeling Notation), used for specifying business process models. Figure 7 exemplifies the definition and use of a simple UML profile.

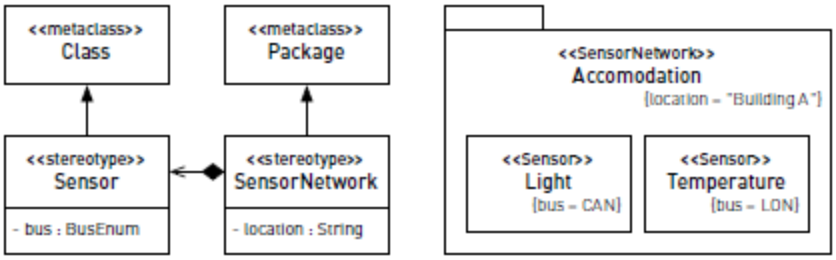



**Figure 7.** Example of a UML profile definition (left) and application (right) **(Hennig, Braune, & Koycheva, 2010)**.

## 2.1.7. Model Transformations

Model transformations are key aspects of MDD, because it is through them that the development process can be (partially) automated. There are two types of model transformations in MDD: (1) Model-to-Model (M2M) transformations and (2) Model-to-Text (M2T) transformations (Czarnecki & Helsen, 2003).

**Model-to-Model Transformations**

A M2M transformation consists in a transformation process that receives one or more models as input and then produces one or more models as output. The most common case is a one-to-one mapping transformation, which receives one input model and produces one output model. An example of such a transformation is the transformation of a class diagram into a relational model for a database. However, there are situations where one-to-many, many-to-one or many-to-many transformations are required (Brambilla, Cabot, & Wimmer, 2012). For instance, in this work we propose a MDD framework that performs many-to-many transformations, i.e., uses multiple diagrams as input to generate multiple diagrams as output. Several languages specialized on M2M transformations have been developed over the last decade, but perhaps the most popular and used ones are QVT and ATL.

QVT (Query-View-Transformation)(OMG, 2011) is a standard proposed by the OMG. As its name suggests, QVT comprises three languages based on MOF for developing model transformations: (1) QVT-Relations; (2) QVT-Operational; and (3) QVT-Core. QVT-Relations defines the correspondences between the source and target metamodels, in a declarative manner. These correspondences can be

both unidirectional and bidirectional. QVT-Relations provides both a graphical and textual concrete syntax. The QVT-Operational language allows the specification of unidirectional model transformations in an imperative manner. QVT-Operation is represented using a textual concrete syntax. At last, the QVT-Core language is a declarative language designed to be the target of the compiler of the QVT Relations language (Brambilla, Cabot, & Wimmer, 2012). Therefore, it is not intended to write transformations directly in QVT-Core. Eclipse provides tool support for QVT, namely for the declarative languages[6] (QVT-Relations and QVT-Core) and QVT-Operation[7].

ATL (ATLAS Transformation Language) (Jouault, Allilaire, Bézivin, & Kurtev, 2008) is a model transformation language developed by and maintained by OBEO and AtlanMod (formerly ATLAS Group). ATL is a rule-based language which uses OCL, but additionally provides features to support model transformations (e.g. the ability to create model elements). ATL transformations are represented textually and are unidirectional. For instance, if a transformation from model A to model B is required, and vice versa, two ATL transformations have to be specified. ATL transformations only allow reading the source model and writing on the target model. This is done to assure that the result of queries in the source model do not differ based on the execution state of the transformation (Brambilla, Cabot, & Wimmer, 2012). Additionally, ATL has been implemented as plug-in for the Eclipse M2M (Model to Model) project[8].

**Model-to-Text Transformations**

In turn, a M2T transformation consists in a transformation process that receives a model as input and then produces text as output. Since MDD intends to generate a running software system from models, the text generated by M2T transformations typically corresponds to source code. Thus, M2T transformation can be also known as code generation. M2T transformations can be divided in two approaches: (1) visitor-based approaches and (2) template-based approaches (Czarnecki & Helsen, 2003).

In visitor-based approaches, a mechanism based on the Visitor design pattern (Gamma, Helm, Johnson, & Vlissides, 1995) is used to traverse the internal representation of the model and write code to a text stream. An example of the application of this approach is the Jamda framework[9] that provides a set of classes to represent UML models, an API for manipulating models and a visitor mechanism.

In turn, the template-based approaches are more popular and commonly used. They consist in the definition of code templates, i.e., text files that contain great part of the code to be generated and some specific annotations (or tags). Those annotations represent the dynamic part of the code template, because in generation time they are replaced by data of the source model. Compared to visitor-based approaches, the structure of a template is resembles more closely the code to be generated. Moreover, since templates are defined in separate files is easier to perform an iterative

[6] http://wiki.eclipse.org/MMT/QVT_Declarative_(QVTd) (Accessed on October 2014)
[7] http://wiki.eclipse.org/M2M/Operational_QVT_Language_(QVTO) (Accessed on October 2014)
[8] http://www.eclipse.org/atl (Accessed on October 2014)
[9] http://jamda.sourceforge.net (Accessed on October 2014)

development. JET (Java Emitter Templates)[10] and Apache Velocity[11] are two examples of simple template engines, while Acceleo and Xpand[12] are two popular examples of template-based code generation frameworks. Table 1 provides a brief comparison between these four technologies.

| Tool | Template Language | Editor Support | Debugger/Profiler Support | Custom Metamodel |
|---|---|---|---|---|
| JET | JSP | Yes (Veloedit Eclipse plugin) | No | No |
| Apache Velocity | VTL | Yes (EMFT JET Editor) | No | No |
| Acceleo | MOFM2T | Yes | Yes | No |
| Xpand | Xpand | Yes | No | Yes |

**Table 1.** Comparison between four template-based code generators.

## 2.1.8. Language Workbench

Over the last decades, some tools were created to support both MDD principles and DSLs. The earlier ones, put forth in the 1980s and 1990s, were the Computer-Aided Software Engineering (CASE) tools. CASE tools focused on providing developers with the tools and methods to express software systems through general-purpose language representations. Despite the attention they attracted, CASE tools failed to be widely adopted and had little impact on commercial software development, chiefly due to: (1) the poorly mapping of general-purpose language representations onto the underlying platforms, which caused the generation of large amounts of code, harder to understand and maintain; and (2) the inability to scale up to handle complex systems, since the tools did not support concurrent engineering, lacked integration of the generated code with other platforms and their graphical representations were too generic and non-customizable to be applied on several domains (Schmidt, 2006). Thereafter, both the evolution of programming languages, as well as the lessons learned from CASE tools, caused the emergence of increasingly better MDD and DSL supporting tools.

More recently, language workbenches are popular solutions to develop DSLs and to overcome the inflexibility of the CASE tools. A language workbench is essentially a tool that supports the definition, reuse and composition of DSLs, as well as the creation of customized Integrated Development Environments (IDEs) (Fowler, 2005). In essence, a language workbench allows not only the creation of DSLs, but also the creation of custom IDEs for their usage. For that reason, language workbenches can also be known as meta-CASE tools (Costagliola, Deufemia, Ferrucci, & Gravino, 2006). Some examples of this kind of tools are the Eclipse Modeling Framework (EMF), Sparx Systems Enterprise

[10] http://www.eclipse.org/modeling/m2t/?project=jet#jet (Accessed on October 2014)
[11] http://velocity.apache.org (Accessed on October 2014)
[12] http://wiki.eclipse.org/Xpand (Accessed on October 2014)

Architect (EA), Sirius[13], JetBrains MPS[14], XText[15] and MetaEdit+[16]. Studies like the ones presented in (Erdweg, et al., 2013)(Saraiva & da Silva, 2008)(Savić, et al., 2014), describe this kind of technologies in more detail and make a comparison between them.

# 2.2. Basics on Mobile Application Development

This section presents some basic concepts around the Mobile Application Development area. Namely, First, Section 2.2.1 provides definitions for mobile computing and mobile devices, and describes some examples of mobile devices. Section 2.2.2 describes the most popularly used mobile platforms, while Section 2.2.3 explains the different types of mobile applications. At last, Section 2.2.4 presents some types of mobile device gestures considered in this work.

## 2.2.1. Mobile Computing and Mobile Devices

Mobile Computing consists in the ability of using a computing device, such as a smartphone, to access data and information from anywhere in the world, i.e., even when the location changes. Nowadays, mobile computing is more present than ever in daily life tasks. This resulted from the great evolution this market has suffered over the last years, with better devices and operating systems. Thus, several types of mobile devices with distinct features (e.g., resolution, memory or GPS) supported by different platforms (or operating systems) and targeted to different functions (e.g., personal or business) are available in the market.

In essence, mobile device is a device that allows people to access data regardless of the location. It is a portable or handheld device of small dimension and weight that combines the characteristics of a computer with a mobile phone. This kind of devices is equipped with a touch screen and/or a keyboard and may have hardware features like camera, GPS, Wi-Fi or Bluetooth card.

Typically, a mobile device is used in situations when the use of a computer would not be practical. Thus, it could be considered as an extension of usual personal computers. Some categories of mobile devices will be presented subsequently.

**PDA –** A PDA (Personal Digital Assistant), also known as personal data assistant, palmtop or pocket computer, is a device that works as a personal information manager. It functions as a phone and a personal organizer, allows wireless connectivity and normally has a touch screen. Nowadays, this kind of devices was supplanted by smartphones. Palm TX or HP iPAQ are two examples of PDAs.

**Smartphone –** A smartphone is a device that combines the characteristics of a mobile phone with a PDA. So, in addition to the support of making calls and sending text messages, smartphones offer capabilities like touch screens, information storage, install applications, Internet access, accelerometer

[13] http://www.eclipse.org/sirius (Accessed on October 2014)
[14] http://www.jetbrains.com/mps (Accessed on October 2014)
[15] http://www.eclipse.org/Xtext (Accessed on October 2014)
[16] http://www.metacase.com/mep (Accessed on October 2014)

or GPS. A crucial difference between smartphones and cell phones is the APIs they offer for running third-party applications. Examples of this kind of devices are the iPhone and Samsumg Galaxy S V.

**Tablet –** A tablet computer, or simply tablet, is a mobile computer larger than PDAs and smartphones that features a flat touch screen and is primarily operated by touching the screen. Usually, tablets use a stylus pen or a digital keyboard instead of a physical keyboard. Tablets are becoming very popular, because they offer many of the features provided by computers and smartphones but are lighter than computers and have a bigger screen than smartphones. iPad and Samsung Galaxy Tab are two popular examples of such devices.

## 2.2.2. Mobile Platforms

A mobile platform (or operating system) consists in the software responsible for controlling and supporting a mobile device. Next, a brief description of the three major mobile platforms (Android, iOS and Windows Phone) is presented. Additionally, Table 2 provides a comparison between these platforms.

| Vendor | Platform | Programming Language | Development Environment | Application Store |
|---|---|---|---|---|
| Google and Open handset Alliance | Android | Java | Eclipse/Android Studio/IntelliJ IDEA | Google Play |
| Apple | iOS | Objective-C | Xcode | App Store |
| Microsoft | Windows Phone | C#/C++ | Visual Studio | Windows Phone Store |

**Table 2.** Comparison between the three major mobile platforms.

**Android**

Android is a free an open source mobile operating system, derived from Linux, developed by Google and the Open Handset Alliance, a consortium of companies devoted to the creation of open standards for mobile devices, where belong companies like Samsung, HTC or LG.

It was initially created by the company Android Inc., which in 2005 was purchased by Google. In September 2008 was released the first Android phone and since then, this operating system suffered several updates with new versions and support for various mobile devices (smartphones and tablets) from different vendors (Allen, Graupera, & Lundrigan, 2010). More recently, Google released the version 4.4 KitKat.

Android applications are written mostly in Java, a language adopted my many developers around the world, a fact that contributes to its large community of developers. Furthermore, Android has an official application market called Google Play (formerly Android Market), where users can download Android apps. Android was considered one of the most popular mobile platforms worldwide during the fourth quarter of 2010, with over 300 million Android devices in use by February 2012.

**iOS**

iOs, formerly known as iPhone OS, is a mobile operating system derived from Mac OS X, developed by Apple. iOS was released in 2007 specifically for the iPhone, but more recently has been introduced in other Apple devices, such as the iPod Touch or iPad. Its release revolutionized the mobile devices market due to the fact that it provided a user experience and applications with unique and high quality (Fling, 2009). The latest version of iOS is version 8 released in September 2014. It is a proprietary operating system, closed source and available only on Apple devices.

iOS applications are written in Objective-C and can be downloaded in the App Store, the application market of Apple, that contains more than 550.000 apps.

**Windows Phone**

Windows Phone (WP) is a proprietary mobile operating system developed by Microsoft. It is the successor of Windows Mobile, but unlike its predecessor, it is more directed to the consumer market. WP was released in 2010 and has as main new features, a graphical interface completely new that uses a design language known as Metro, and allows full integration with either Microsoft Services, such as Windows Live, or third party services like Facebook.

In February 2011, Microsoft and Nokia announced a partnership between the companies that established Windows Phone as the primary mobile platform of Nokia devices. The most famous result from that partnership is the Nokia Lumia devices. The latest version of WP is 8.1. Like its competitors, WP has its own application market, the Windows Phone Marketplace. The WP applications are written in C# using the .NET Framework.

## 2.2.3. Types of Mobile Applications

Essentially mobile applications can be classified in two groups: native and web. A native mobile application is an application designed to run on a specific operating system of a mobile device. For instance, a native application developed for the iPhone will just run on its proprietary platform, iOS. This kind of apps may come installed with the device or downloaded, freely or by small amounts, from the known App Stores.

In turn, a web mobile application consists in a common Internet application that resides on a server and could be access via Internet. Every time the application is executed, it is downloaded and processed locally. Moreover, it is an application designed to fit the screens of most mobile devices and written as web pages using languages supported by browsers, like HTML, CSS and JavaScript. For this reason, this kind of applications can be accessed from any device that has Internet access and a compatible browser.

One of the major disadvantages of web applications compared to the native is related with the poorer user experience they provide. Because access to physical resources of the device (e.g., buttons, GPS, camera or accelerometer) is limited, the performance, responsiveness and even the look and feel offered are lower than those offered by native applications. Another disadvantage of web

applications is the need for constant connection to the Internet that cannot always be available or not even be desirable. Thus, web applications typically do not support offline operation.

On the other hand, the greatest advantage of web applications is the lower cost of development, deployment and maintenance. This can be explained mainly, because web applications are written with well known and widely used languages throughout the Internet, as opposed to the different languages of the various mobile platforms (such as, Objective-C used specially for iOS). Thus, it's possible to conclude that a web application may be available on several platforms and devices since they have a compatible browser.

The emergence of HTML5 came to try to solve the main problems of web applications with the introduction of new tags and APIs. For instance, HTML5 improves video and media support, animations and graphics, provides geolocation and allows offline operation through the use of cache. This way, HTML5 allows a more proximity between the two types of applications described above, through the creation of hybrid applications that aim to combine the best of both worlds. Hybrid applications are not native nor web, because they are built with web technologies, but then are wrapped in a platform-specific shell that allows them to be installed like native ones. Nevertheless, the specification of HTML5 is still in progress, which implies that this language is not completely defined and still lacks the definition of standards to be universally adopted by browsers. Anyway, the use of hybrid applications is seen as an area with future, existing several tools and frameworks that use these technologies (Charland & LeRoux, 2011).

### 2.2.4. Mobile Device Gestures

Mobile devices make an intense use of onscreen touch gestures, because they have a limited number of physical buttons and a touch screen that occupies most of the device size. Therefore, the preferred way of interacting with a mobile application is through gestures. For this research work, there was considered a set of the most common gestures used either in Android, iOS and Windows Phone. This set of gestures is described in Table 3.

| **Gesture** | **How to do it** | **Functionality** | |
|---|---|---|---|
| **Tap (Touch)** | Press an item and lift. | Triggers the functionally of a certain item (selects or opens). |  |

| | | | |
|---|---|---|---|
| **Double Tap (Double Touch)** | Press an item twice quickly and lift. | Zooms in or out. |  |
| **Long Tap (Long Press or Tap and Hold)** | Press an item, hold and lift. | Opens a context menu or enters data selection mode where the items can be selected. |  |
| **Swipe (Drag, Pan, Flick or Scroll)** | Press the screen, move the finger in the desired direction and lift. | Reveals hidden content by scrolling with the finger, allows navigating through screens or delete an item from a list. |  |
| **Pinch (Pinch Close)** | Press the screen with two fingers, move them inwards and lift. | Zooms gradually out a website, picture or map. |  |
| **Stretch (Pinch Open)** | Press the screen with two fingers, move them outwards and lift. | Zooms gradually in a website, picture or map. |  |

**Table 3.** Types of mobile device gestures (Images retrieved from http://www.windowsphone.com/en-us/how-to/wp7/start/gestures-flick-pan-and-stretch).

## 2.3. Mobile UI Design Patterns

The User Interface (UI) design is a crucial task during the development process of a mobile application. The quality of the UI design can determine the success or not of a mobile application. Particularly, it influences not only the attractiveness of the mobile application, but also, and perhaps more important, its usability. However, designing user interfaces is a hard task, because it varies according to the natural preferences and tendencies of the designer. Therefore, during years, UI design experts gathered and defined UI design patterns in order to ease the development of user interfaces. These design patterns are nothing more than cases of success that solve a certain recurring problem. Additionally, the mobile platform companies have also defined a set of UI guidelines with the goal of helping and standardizing the development of user interfaces for their platforms.

Thus, this section describes some UI patterns for mobile applications that have been taken into account during this research work.

**Springboard –** The Springboard follows a grid layout where each option in the grid triggers the navigation to a screen. The grid layout is typically limited to a maximum of 4x4 options. If the number of options is greater than that, then a horizontal swipe gesture is used to navigate between the other grids of options. Usually, this is performed using a navigation pattern known as Page Carousel, which uses an indicator at the bottom of each screen to display how many pages compose the Springboard (Neil, 2012). An example of the use of this pattern is illustrated in Figure 8.

**Tab Menu –** The Tab Menu consists in a group of navigation options placed side by side and at the top or bottom of the screen. The options or items of a Tab Menu are typically represented by an icon and a text. When a given option is selected the navigation to another screen is performed and the Tab Menu remains there with the selected option highlighted in a different color. This pattern is commonly used when the number of navigation options is small (Neil, 2012)(Mendoza, 2013). An example of the use of this pattern is illustrated in Figure 8.

**List –** The List pattern is similar to the Springboard, but it represents the navigation options in a vertical list. A List is well suited when the set of options is very large. Additionally, it works well for options with icons, long titles or that require subtext descriptions (Neil, 2012)(Mendoza, 2013). The List pattern has several variations, namely lists can be sectioned in categories, can provide widgets for searching, ordering and filtering the list, and apply a certain action to the list items. An example of the use of this pattern is illustrated in Figure 8.

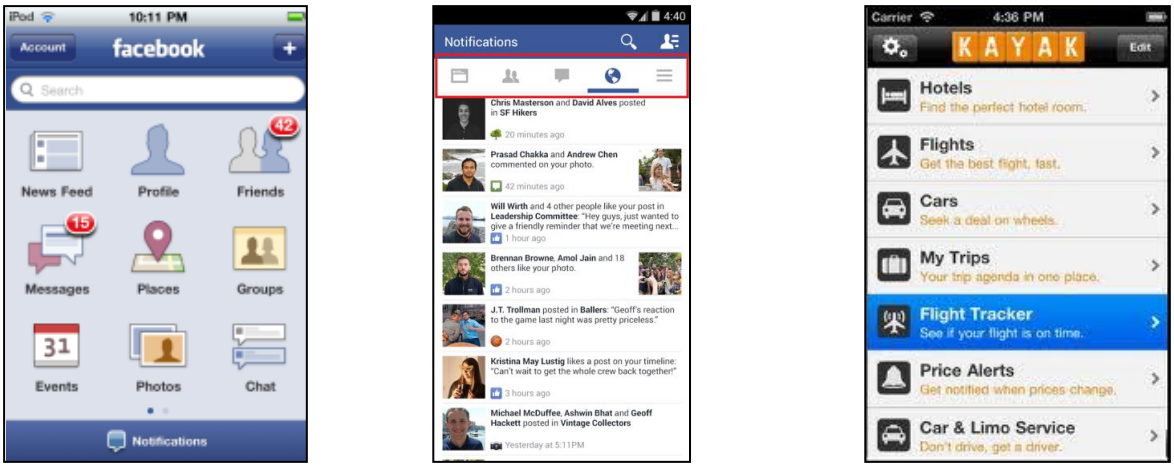



**Figure 8.** Mobile applications that use the Springboard, Tab Menu and List patterns (from left to right).

**Options Menu –** An Options Menu consist in a group of screen-level actions, which can change the state of the current screen or trigger navigation to another one. An Options Menu can be accessed in different ways: through a hardware button, by tapping a menu button widget or even can always be visible on the screen (Neil, 2012)(Hoober & Berkman, 2011). The use of this pattern differs between mobile platform, while Android and Windows Phone use it often, iOS prefers using a top navigation bar for the same purpose. Even the newer versions of Android (from version 3.0 and above) replaced this menu by a top action bar. An example of the use of this pattern is illustrated in Figure 9.

**Context Menu –** Contextual Menus can be used to interact with a particular element on the screen. Similarly to the Options Menu, it provides a set of options. A Context Menu is usually revealed by performing a tap or long-tap gesture on the desired element. This type of menu is commonly used along with lists (Neil, 2012). An example of the use of this pattern is illustrated in Figure 9.

**Form –** A Form is a widely used widget that contains other widgets intended for data entry. It represents a straightforward way of aggregating related widgets (Neil, 2012). An example of the use of this pattern is illustrated in Figure 9.

**Dialog –** A simple dialog with instructions is the most common type of invitation in mobile apps, probably because it is the easiest to program. It is also most likely to be dismissed and ignored (Neil, 2012). An example of the use of this pattern is illustrated in Figure 10.

**Explicit Search –** This pattern consists in the use of a search bar, typically a textbox with a button beside it, where the user specifies what he intends to find. This pattern relies on a specific action to perform the search, like tapping on the search button. Then, the search results are typically displayed in the area below the search bar, but can also be displayed in a dedicated screen or on map for instance. This pattern is usually used with the List pattern. There are other search patterns like Auto-Complete Search or Dynamic Search, but for simplicity purposes we only have considered this one (Neil, 2012). An example of the use of this pattern is illustrated in Figure 10

**Sort and Filter –** The Sort pattern consists in a group of options that allow sorting a list (List pattern). If the number of options is small, the sort options can be placed at the top or bottom of the screen.

Otherwise, the choice of the sort option can be performed using a Menu, Dialog or a Form. The Filter pattern is similar to the Sort pattern with the difference that instead of sorting a list, it restricts the list based on some criterion. Additionally, the Filter allows restricting the resulting list by possibly choosing more than one criterion at a time (Neil, 2012).An example of the use of the Sort and Filter patterns is illustrated in Figure 10.

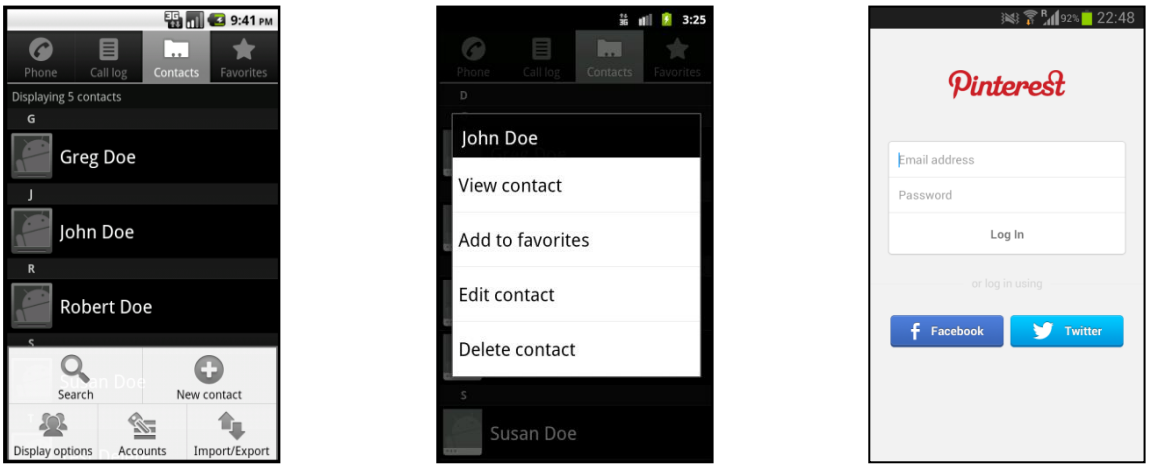



**Figure 9.** Mobile applications that use the Options Menu, Context Menu and Form patterns (from left to right.)

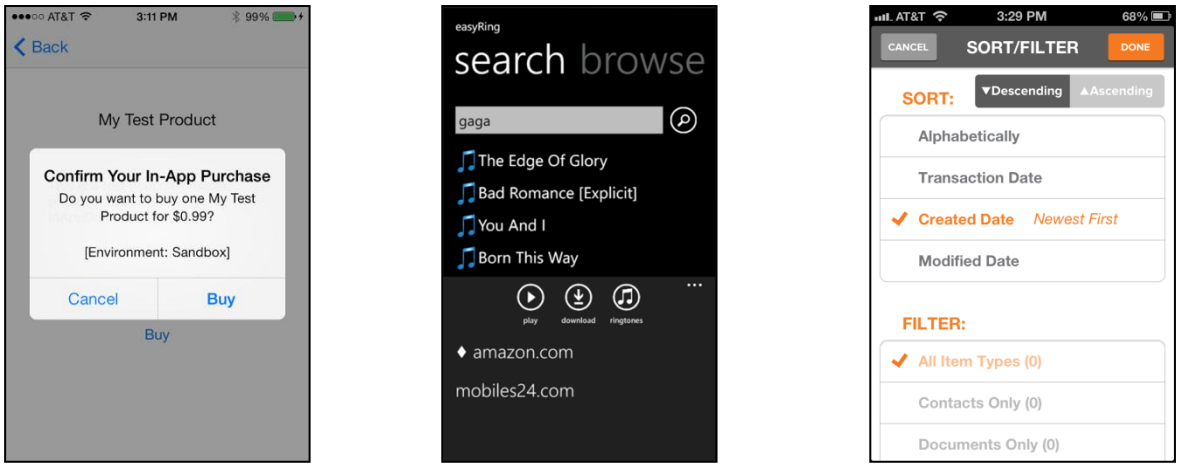



**Figure 10.** Mobile applications that use the Dialog, Explicit Search and Sort and Filter patterns (left to right).

**Call to Action Button –** This pattern is used when there is only one primary action to be performed in a certain screen. That action is performed through a button that should be prominently displayed (Neil, 2012). An example of the use of this pattern is illustrated in Figure 11.

**Map and Location –** This pattern is usually employed when the user wants to search for a given location or point of interest (POI). This pattern shows a map containing markers representing the interest locations. Then, by clicking on each marker is possible to access further details on these physical locations (Mendoza, 2013). An example of the use of this pattern is illustrated in Figure 11.

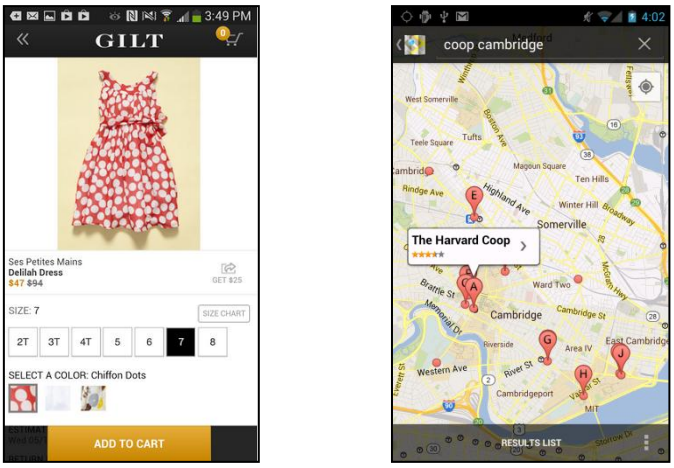



**Figure 11.** Mobile applications that use the Call to Action Button (left) and Map and Location (right) patterns.

# 2.4. Specific Issues of Mobile Applications

Before defining a 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 (Weiser, 1993)(Forman & Zahorjan, 1994)(Satyanarayanan, 1996)(Nilsson, 2009)(Kramer, Clark, & Oussena, 2010)(Neil, 2012).

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) (Fernando, Loke, & Rahayu, 2013).

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, i.e., the components of its layout must have a relative positioning among themselves.

Another pattern observed in several mobile applications is the existence of GUI element containership. Basically, it consists in grouping widgets within each other in order to better organize the information displayed. For instance, a list can be organized in list groups that in turn contain list items. Also, menus contain menu items.

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.

# 2.5. Cross-Platform Tools for Mobile Apps

Over the last years some cross-platform tools and frameworks have appeared. Basically, a cross-platform tool or framework is a software that allows developers to create and distribute applications for multiple platforms from almost the same source code, reducing the incremental cost per platform and maximizing the code reuse (Hartmann, Stead, & DeGani, 2011)(Paananen, 2011).

They can be classified according to its technological approach that can target different users and different use cases. Figure 12 depicts the categories in which they can be classified:

**Web-to-native wrapper –** Set of libraries that allow the delivery of native apps using web languages, such as HTML, CSS or JavaScript. The web code is packaged with these libraries inside a native app shell linking the web code to native features.

**Runtime –** Execution environment and layer that makes the mobile application and the native platform compatible. Runtimes uses several methods of code execution, such as interpretation, translation or virtualization. The typical approach consists in translating the code to bytecode and then, at runtime, that bytecode is executed by a virtual machine supported by the mobile device.

**Source code translator –** This solution translates or cross-compiles source code into another language, such as bytecode, native language (e.g., Java or Objective-C) or assembly. As can be pointed, source code translators are usually used with Runtimes.

**App Factory –** Visual tool that allow users to develop their app without code. Usually, App Factories are based in templates and provide drag and drop features to generate the application code.

**Domain Specific Language (DSL) –** Programming language focused on a particular problem domain that uses abstractions and domain concepts to describe the application. DSLs can be divided in textual and visual. In turn, UML profiles are specific types of visual DSLs.

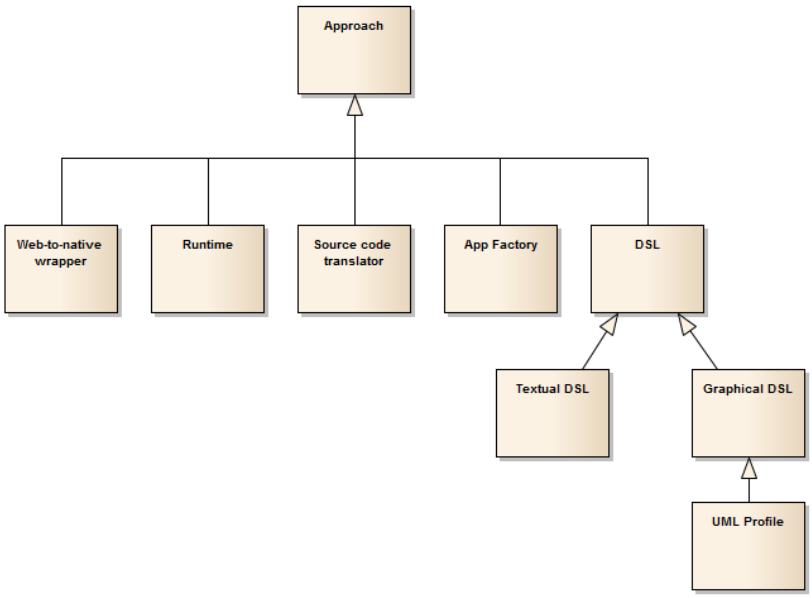



**Figure 12.** Technological approaches of cross-platform tools for mobile apps.

In addition, surveys like the ones presented in (Paananen, 2011)(Ribeiro & da Silva, 2013)(Singh & Palmieri, 2011) evaluate and compare several of these technologies.

# 3. XIS-Mobile Approach

This chapter presents an overview of the approach proposed in this dissertation, known as XIS-Mobile approach. The XIS-Mobile approach relies in two components: the XIS-Mobile language, a DSL for defining mobile applications; and the XIS-Mobile framework, a framework that supports the MDD of mobile applications using the XIS-Mobile language. Thus, Section 3.1 provides the background from which the XIS-Mobile approach is based. Since XIS-Mobile is based on the XIS profile, a presentation of this profile is made. XIS-Mobile is closely linked to XIS, because it reuses, extends and materializes the approach and some of the concepts proposed by XIS. Section 3.2 presents an overview of the organization of the XIS-Mobile language and the dependencies between its views. Section 3.3 details the design approaches supported by the XIS-Mobile approach. At last, Section 3.4 describes the three case study applications used during this research work and that are referenced throughout this dissertation.

## 3.1. Background

This section presents the background from which this research work has been based that motivated the development of a UML Profile focused on mobile application development.

The work presented in this dissertation materializes an idea for the extension of an existing DSL named XIS (eXtreme modeling of Interactive Systems) (da Silva A. , Saraiva, Silva, & Martins, 2007). XIS is a UML profile focused on the design of interactive software systems at a PIM level (Platform-Independent Level, according to MDA terminology) by following a MDD approach, specifically the aforementioned ProjectIT approach. As shown in Figure 13, XIS is organized in three major groups of views: Entities, Use-Cases and User-Interfaces.

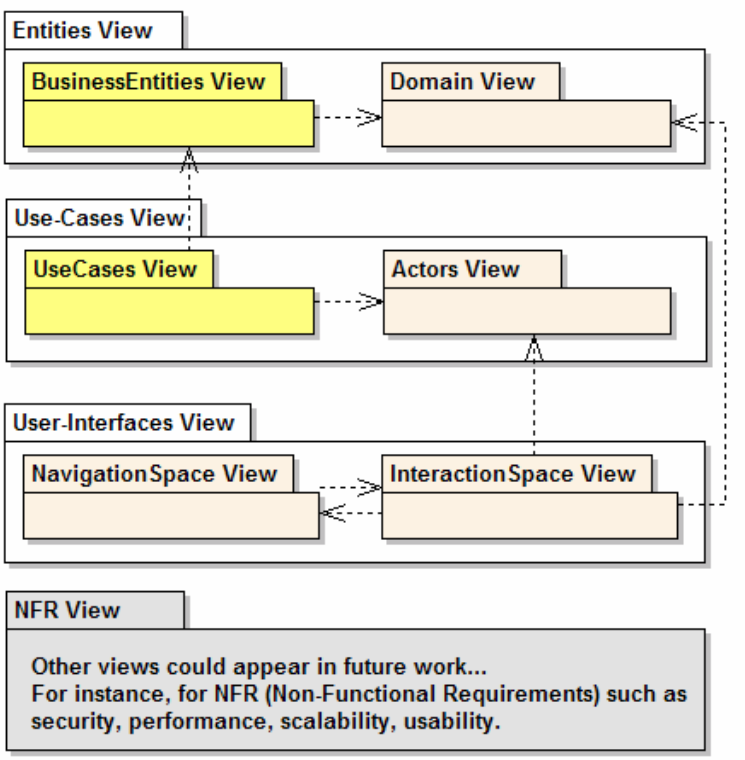



**Figure 13.** Multi-view organization of the XIS profile **(da Silva A. , Saraiva, Silva, & Martins, 2007)**.

The set of Entity views is composed of the Domain and BusinessEntities views. In the Domain View depicts 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. Thus, 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.

Additionally, XIS also defines two modeling approaches: the smart approach and the dummy approach (da Silva A. , Saraiva, Silva, & Martins, 2007)(da Silva A. , Saraiva, Ferreira, & Silva, 2007). 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 M2M transformations (from the Domain and UseCases views) and then extended or refined

through direct design. On the other hand, when usin the dummy approach, the designer has to manually define by scratch the entire Domain, Actors, NavigationSpace and InteractionSpace views.

Summarizing, XIS represents a useful solution to model simple desktop or web interactive applications (da Silva A. , Saraiva, Ferreira, & Silva, 2007). However, when the goal is to model mobile applications XIS reveals some limitations, namely regarding a proper support for the specification of internet connection, gestures, localization and other context-aware issues frequently used in mobile applications. In addition to that, from what have been researched and from 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 dissertation not only presents the XIS-Mobile language focused on modeling mobile applications, but also its supporting framework based on the Sparx Systems Enterprise Architect (EA), a widely used and popular modeling tool. The XIS-Mobile framework is described in detail in Chapter 5, but it is important to highlight that it already implements the smart approach proposed by XIS.

## 3.2. Overview

XIS-Mobile proposes a DSL, in the form of a UML profile, to design mobile applications in a platform-independent way using a MDD approach. XIS-Mobile allows the specification of mobile applications through the production of Platform-Independent Models. Then, XIS-Mobile also proposes a supporting framework that, apart from allowing the edition of these models, permits the application of M2M transformations to generate the more laborious part of the models and M2T transformations to generate code for multiple mobile platforms (Android, Windows Phone and iOS). To allow these features, XIS-Mobile has been designed having in mind the following four fundamental principles of MDD:

**Modularization –** The division of systems in modules is quite important, specially when they are large. XIS-Mobile addresses this principle through the use of packages (views) to organize its structure, by using the concept of business entity which aggregates several domain entities and by representing interaction spaces at different levels of detail depending on the view they are depicted. The business entity and interaction space concepts are explained in detail in Chapter 4.

**Separation of concerns –** XIS-Mobile addresses this principle by using a multi-view organization in which each view handles different concerns and is, as minimal as possible, independent of the others. In the future, other concerns and views can be added to XIS-Mobile (e.g. views to support scenarios related to the mobile cloud computing, context awareness or content management-based systems).

**Use-Case-driven approach –** XIS-Mobile allows the specification of the application functionalities through the identification of use cases. XIS-Mobile's use cases also capture mobile UI patterns that are crucial during the M2M transformation stage.

**Model transformations –** XIS-Mobile advocates the use of M2M transformations in order to increase the productivity of the modeling process by producing the most time-consuming models. Also, XIS-

Mobile proposes the use of M2T transformations to generate source code for multiples mobile platforms from the same models. XIS-Mobile proposes two design approaches, described in Section 3.3, that make a different use of model transformations.

As shown in Figure 14, XIS-Mobile is organized in four major packages of views: Entities, Architectural, UseCases and User-Interfaces views. In turn, the Entities View contains the Domain and BusinessEntities views, while the User-Interfaces View contains the NavigationSpace and InteractionSpace views. These views depend on each other and that fact influence the design approaches and the transformation stages. The Domain View does not depend on any other view, since it is the starting point to define the problem domain. In turn, the BusinessEntities View depends on the Domain View, because business entities aggregate domain entities. The UseCases View depends on the BusinessEntities View and the Architectural View, since each use case must be connected to a business entity (specified in the BusinessEntities View) and/or provider (specified in the Architectural View). The Architectural View does not depend on any other view. The InteractionSpace and the NavigationSpace views depend on each other. The InteractionSpace View also depends on the BusinessEntities View, because an interaction space can be associated to a business entity; and on the Architectural View, because some widgets can trigger WebService actions defined on that view. Each one of these views is detailed in Chapter 4.

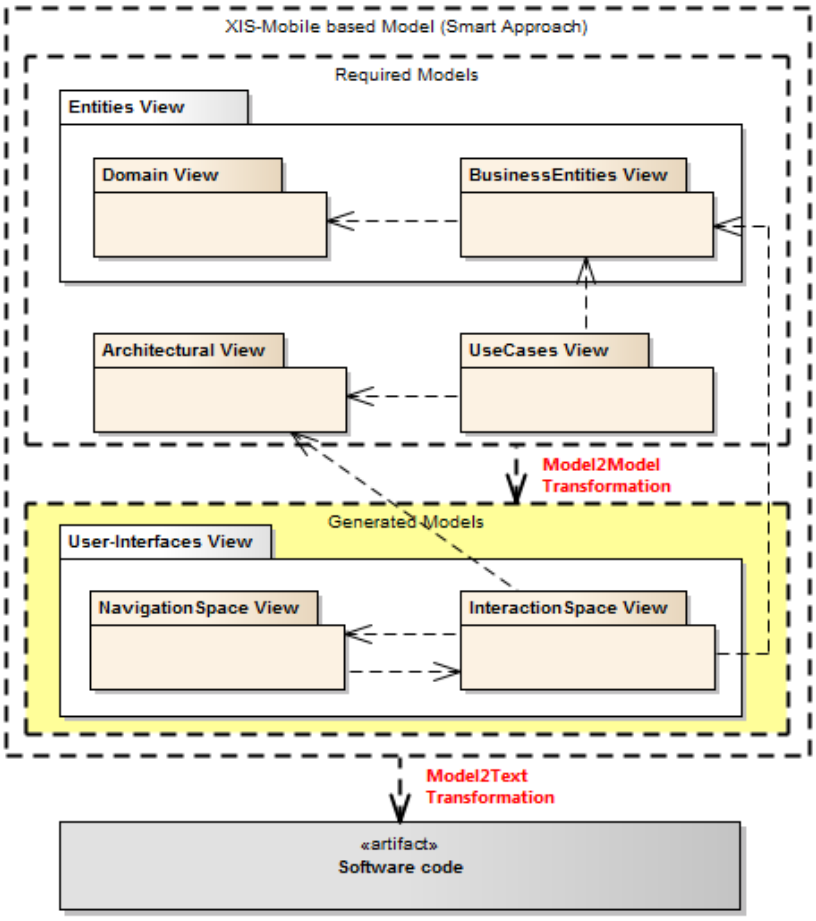



**Figure 14.** Multi-view organization of XIS-Mobile.

## 3.3. Design Approaches

XIS-Mobile supports two design approaches for design mobile application: the dummy and the smart approaches. Essentially, the main difference between them is that the smart approach uses M2M transformations. By leveraging this kind of transformations, XIS-Mobile avoids the manual creation of the User-Interfaces View package, which is the most laborious and time-consuming package.

Using the dummy approach, the developer needs to define all views manually, with the exception of the Architectural View that is only required if the application interacts with external entities, like providers or web servers. On the other hand, using the smart approach (see Figure 14), the developer should define only the Domain, BusinessEntities and UseCases views. Again, like in the dummy approach, the Architectural View is only necessary if the interaction with external entities is a requirement. Then, by leveraging the patterns represented by each use case and the information of the Domain and BusinessEntities views is possible to perform M2M transformations and automatically generate the User-Interfaces View (InteractionSpace and NavigationSpace views). Then the developer can customize the generated models if he desires so. Therefore, this approach can be much less time-consuming than the dummy approach, since the more complex views are automatically generated. This approach should be used whenever possible, because it speeds up the modeling process. After defining all these view models either using the dummy or the smart approach, it is possible to generate native source code from them using M2T transformations.

## 3.4. Case Studies

During this research work, three case studies have been defined in order to test and evaluate the XIS-Mobile approach. Throughout this dissertation, these case studies are used to exemplify various aspects of XIS-Mobile, contributing for a better understanding and simplicity of explanation. Next, a description of each case study is provided.

**Case Study A – The To-Do List 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 is able to manage the tasks: create new tasks, view and edit the details of an existing task or instead remove 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.

**Case Study B – The Tourism App**

The Tourism App consists in a tourism advisor that presents the Points of Interests (POIs), organized in tours and categories, of a certain context area. A context area contains one or more POIs and tours. Then, each POI has a category, can be associated to many tours and belong to the favorites list. A tour also has a category associated.
When a user enters the application, a list with the tour categories and another one with the POI categories are presented. There is also a toolbar that allows the user to navigate to a screen with a map with all POIs, to navigate to a screen where he can search through the full list of POIs, to synchronize his application data with a remote server, and also to see his list of favorites POIs. When the user selects a tour category, he will navigate to a screen with the list of the tours belonging to that category, and posteriorly read the information of the tour, namely the POIs that compose it, the distance and the duration. In turn, when a user selects a POI category, a list with POIs belonging to that category will be shown. Then, by choosing a POI, he will navigate to the POI details page where he can read information about it, see the POI location in a map, add it to the favorites list and navigate to a web page containing more information about the POI.

**Case Study C – The Flight Reservation App**

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

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

Case studies A and B have been used to evaluate the XIS-Mobile approach through their implementation either manually or by using the XIS-Mobile framework. More details about this evaluation are described in Section 6.1. Case study C has been used as assignment of an evaluation user session with people not involved in this research work. Section 6.2 provides more information about this session.

# 4. XIS-Mobile Language

This chapter presents the XIS-Mobile language, a DSL that allows designing mobile applications in a platform-independent way. As seen before, the XIS-Mobile language is divided in four main views: (1) Entities View; (2) Architectural View; (3) UseCases View; and (4) User-Interfaces View. Then, the Entities View package comprises the Domain and BusinessEntities views. In turn, the User-Interfaces View comprises the NavigationSpace and InteractionSpace views. Therefore, each one of the views of the XIS-Mobile language is described in detail and its use is exemplified for the case study A – The To-Do List App described in Section 3.4. It was decided to use this case study due to its simplicity and practical use that contribute for a better understanding and ease of the explanation. Additionally, since the XIS-Mobile language is represented as UML profile, every concept of its views has a stereotype associated and possibly some tagged values, which provide useful information to the M2T transformation stage.

Thus, Section 4.1 describes the Entities View package and its constituents (Domain and BusinessEntities views). Section 4.2 describes the Architectural View. Section 4.3 describes the UseCases View. At last, Section 4.4 describes the User-Interfaces View, namely the NavigationSpace and the InteractionSpace views it comprises. Additionally, the full and extensive specification of the XIS-Mobile language is provided in Appendix A.

## 4.1. Entities View

The Entities View is the first view to be defined as soon as the requirements of the mobile application to be developed are clearly specified. Thus, the Entities View identifies and defines the concepts relevant to the problem domain. This view is divided in the Domain and BusinessEntities views that are described below.

### 4.1.1. 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. The metamodel of this view is shown in Figure 15.

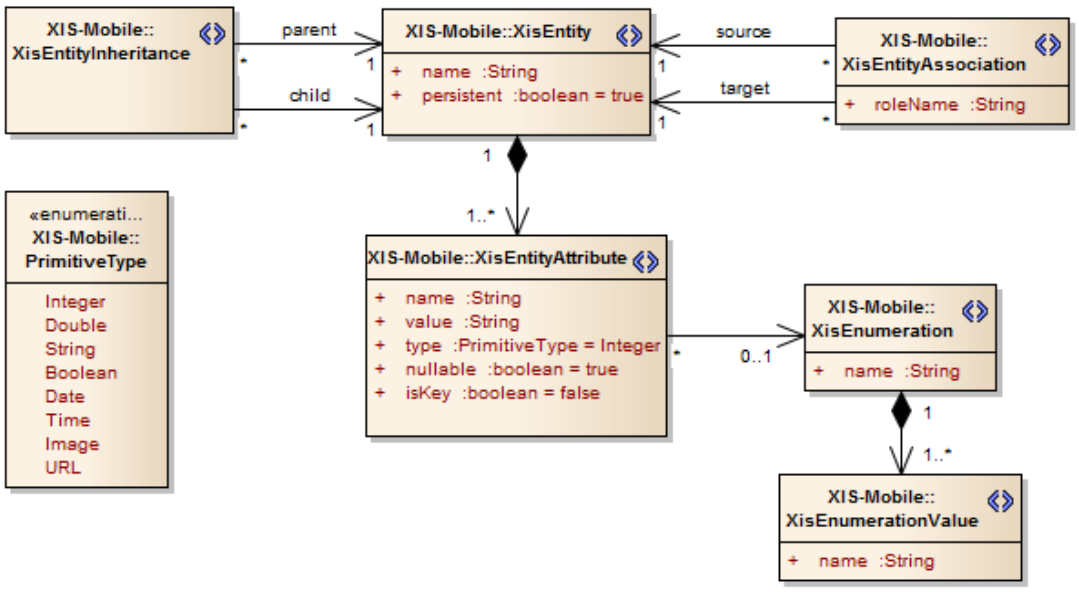



**Figure 15.** Metamodel of the Domain View.

Figure 16 illustrates the To-Do List App's Domain View. Thus, this case study is composed of three XisEntities: Task, Note and Category. The Task contains three XisEntityAttributes 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.

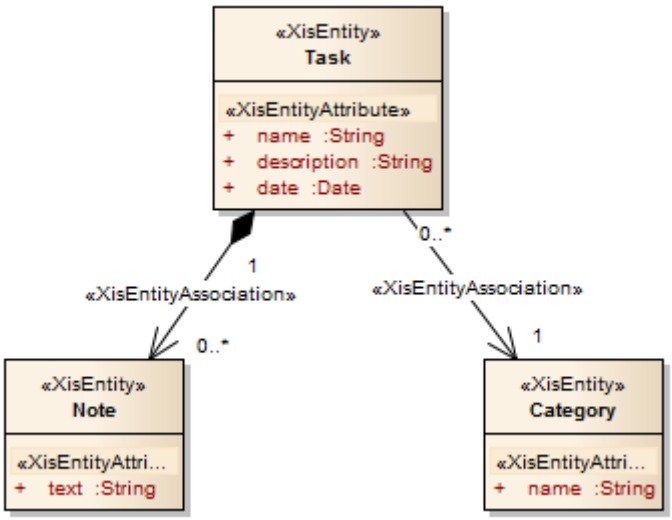



**Figure 16.** Domain View of the To-Do List App.

## 4.1.2. BusinessEntities View

The BusinessEntities View represents high-level entities, known as business entities, using a Class Diagram. A business entity is an entity of a coarser granularity, represented as a class with the stereotype XisBusinessEntity, which aggregates one or more XisEntities (from the Domain View). The goal of a business entity is to provide context to use cases and interaction spaces by defining the set of domain entities that can be manipulated by 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. The metamodel of this view is shown in Figure 17.

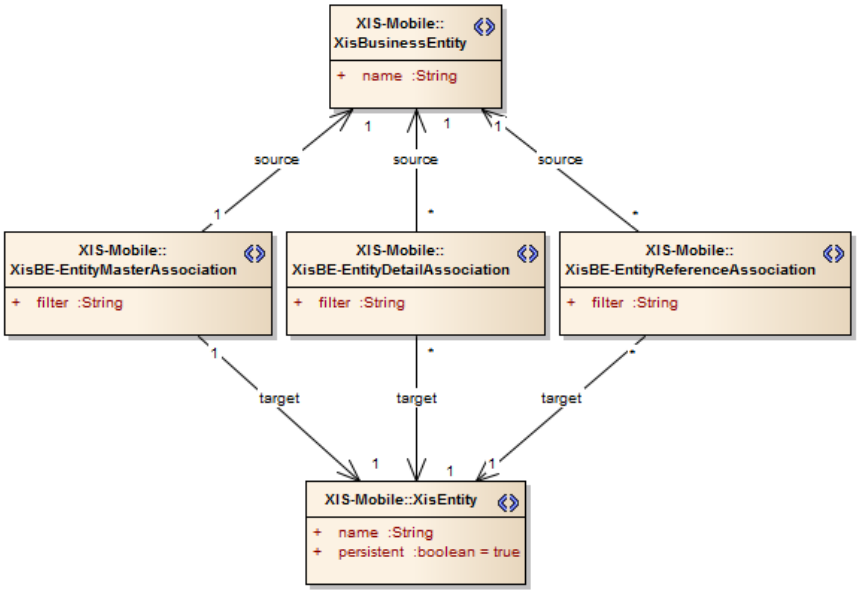



**Figure 17.** Metamodel of the BusinessEntities View.

Regarding the To-Do List App, we defined the TaskBE business entity, which has Task as master entity, Note as detail entity and Category as reference entity. The representation of this view is shown in Figure 18.

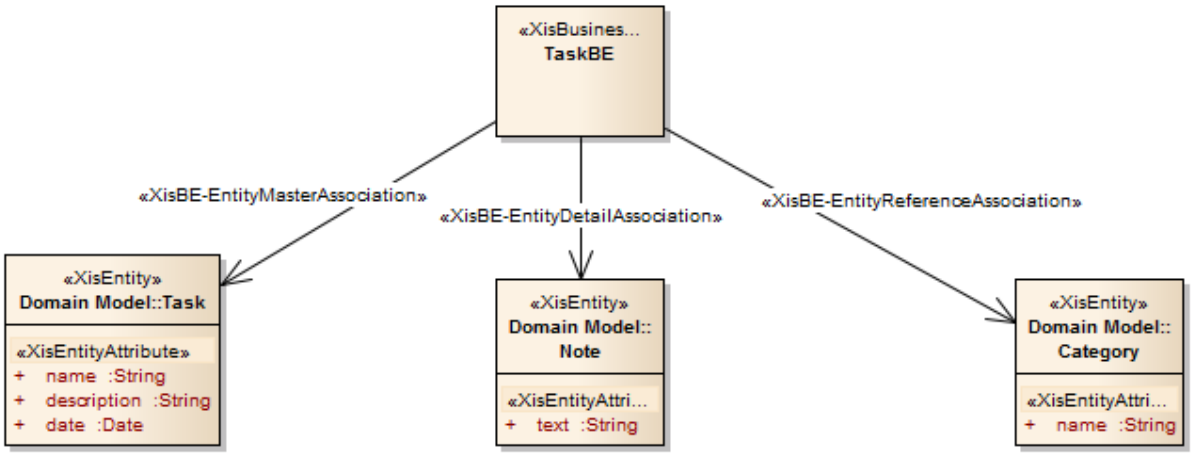



**Figure 18.** BusinessEntities View of the To-Do List App.

# 4.2. 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. The metamodel of this view is shown in Figure 19.

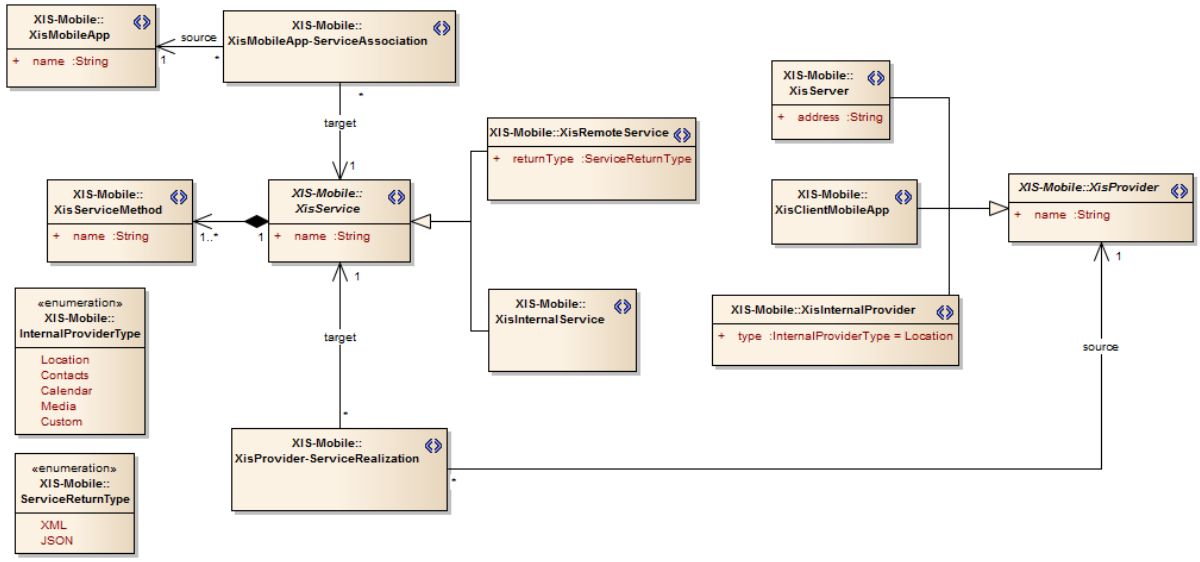



**Figure 19.** Metamodel of the Architectural View.

Considering the To-Do List App, it involves two services (see Figure20): a XisInternalService, named CalendarService, provided by a XisCalendarProvider and a XisRemoteService, named To_DoService, provided by a XisServer. Each one of these services provide one XisServiceMethod.

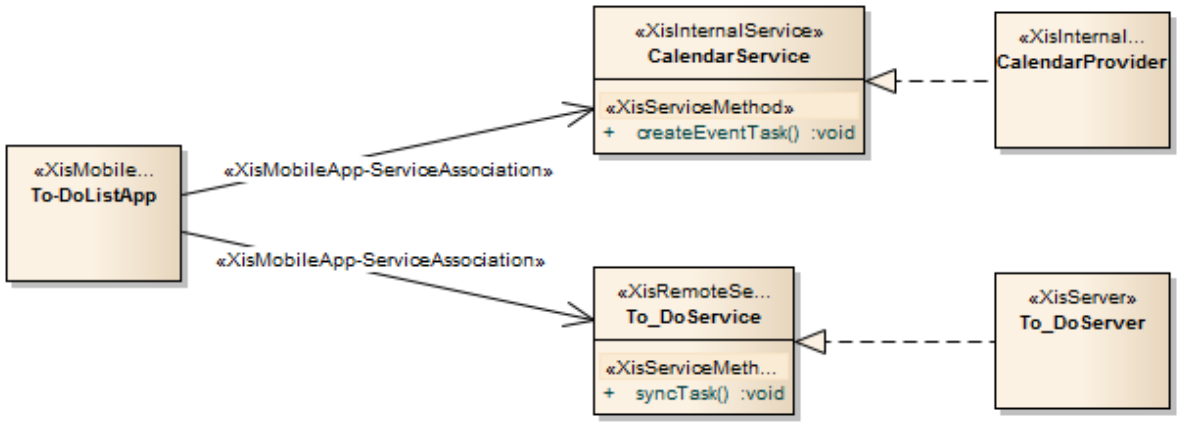



**Figure 20.** Architectural View of the To-Do List App.

# 4.3. 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 (Neil, 2012). The UI patterns used, as well as the full M2M transformation stage, are described in more detail in Section 5.3. 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 (Langlands, 2010). 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. The metamodel of this view is shown in Figure 21.

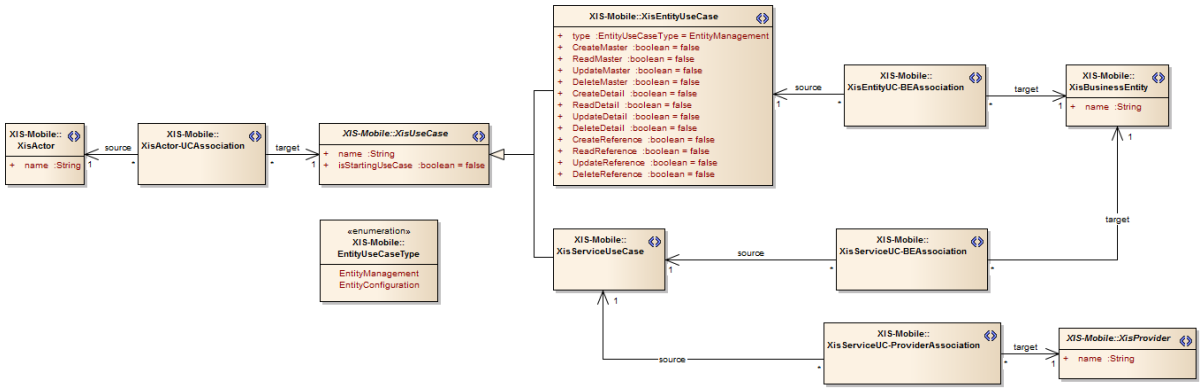



**Figure 21.** Metamodel of the UseCases View.

Concerning the To-Do List App, there were defined three use cases (see Figure 22): (1) "Manage Tasks", a XisEntityUseCase of type "EntityManagement" and connected to the TaskBE; (2) "Sync Tasks", a XisServiceUseCase connected to the TaskBE and to the To_DoServer, and used to send the

task information to that server; and (3) “Create Calendar Task”, of a XisServiceUseCase type “Service” used to interact with the Calendar Provider in order to create a calendar task.

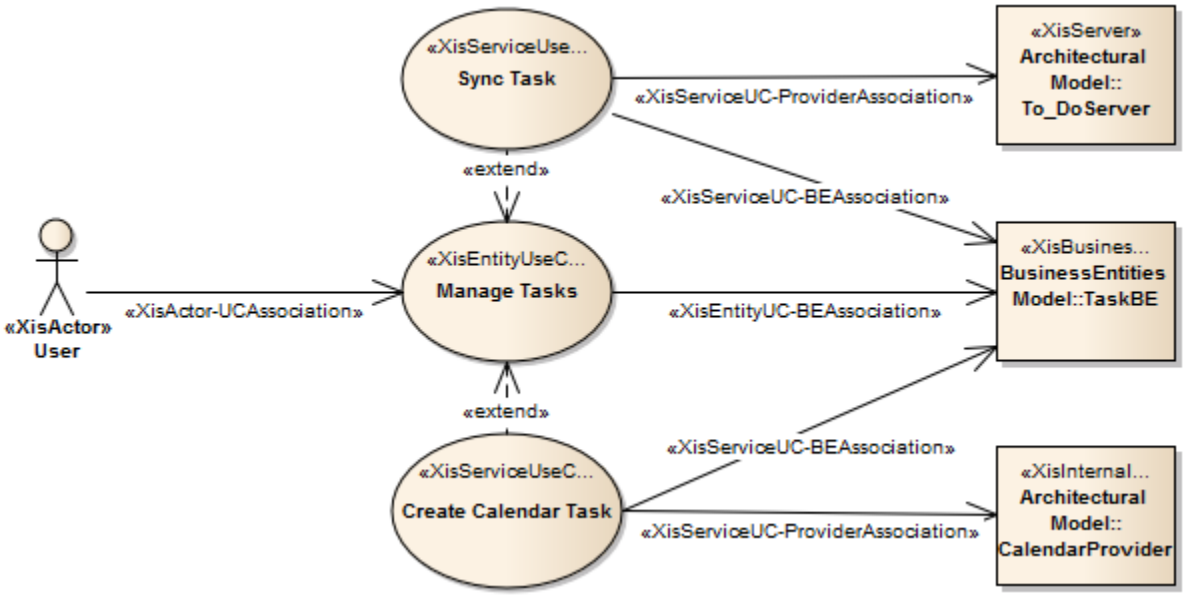



**Figure 22.** UseCases View of the To-Do List App.

# 4.4. User-Interfaces View

The User-Interfaces View is used to specify the content and organization of the screens of the mobile application, known as interaction spaces, as well as the navigation flow between them. The User-Interfaces View is divided in the NavigationSpace and InteractionSpace views. The NavigationSpace View is responsible to model the navigation flow between the interaction spaces, while the InteractionSpace View is responsible to detail each interaction space. These views are described below.

## 4.4.1. NavigationSpace View

The NavigationSpace View represents the navigation flow between the various interaction spaces of the mobile application, using a Class Diagram. This navigation flow is represented by directed associations with the stereotype XisInteractionSpaceAssociation. The XisInteractionSpaceAssociation is the only stereotype provided by the XIS-Mobile language to be applied in this view. Each XisInteractionSpaceAssociation is also responsible to show the action (or event) that triggered the navigation between the interaction spaces, because its name must be the same as the action’s name. This view provides a good support for documenting the structure of the system and eases the introduction of future corrections and improvements in the mobile application navigation. For simplicity and readability reasons, the details of each interaction space are represented in the InteractionSpace View. The metamodel of this view is shown in Figure 23.

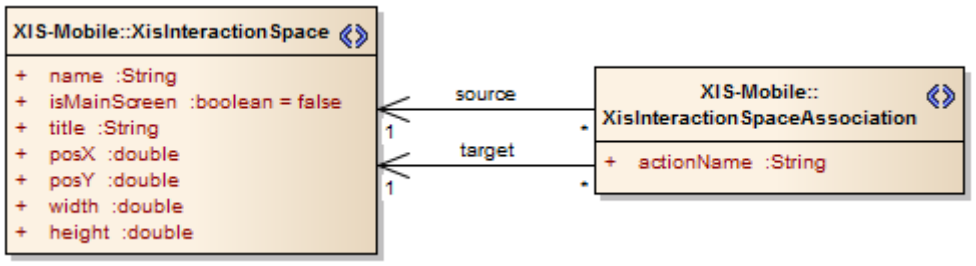



**Figure 23.** Metamodel of the NavigationSpace View.

The To-Do List App's NavigationSpace View is composed of four XisInteractionSpaces with eight XisInteractionSpaceAssociations between them (see Figure 24): (1) TaskListIS, which lists all the tasks; (2) TaskEditorIS, which allows creating, editing and visualize the information of a task; (3) NoteManagerIS, which lists all the notes associated to a certain task; and (4) NoteEditorIS, which allows visualizing the details of a certain note.

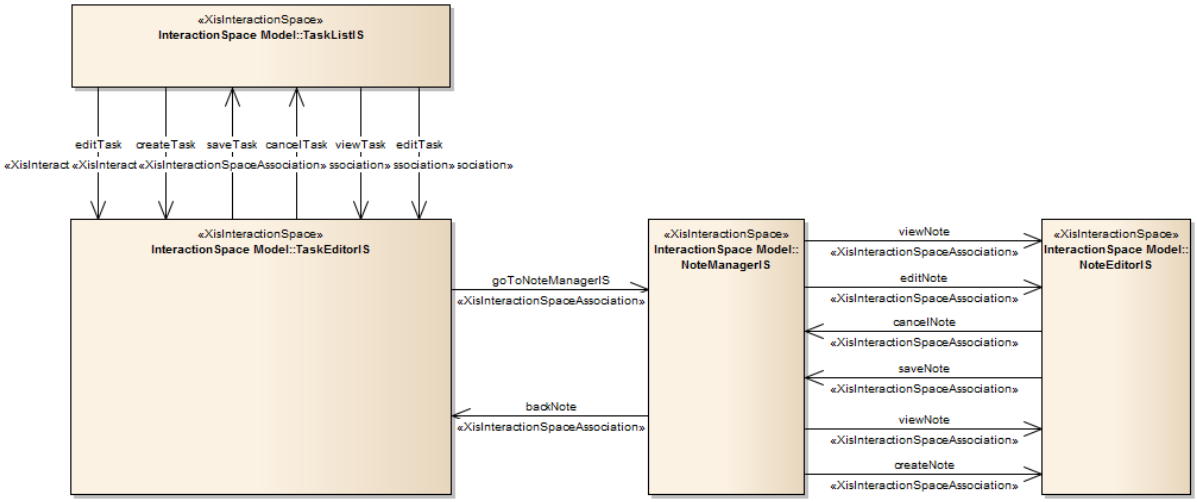



**Figure 24.** NavigationSpace View of the To-Do List App.

## 4.4.2. 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, 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 (Gamma, Helm, Johnson, & Vlissides, 1995). 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”. An excerpt of the metamodel of the InteractionSpace View is shown in Figure 25.

Considering once again the To-Do List App (see Figure 26), the TaskListIS is the main screen of the application (has the “isMainScreen” tagged value set to true). This interaction space has a XisList, a XisMenu of type “OptionsMenu” and a XisMenu of type “ContextMenu”. The XisList is a list of tasks and contains one XisListItem with a “Tap” gesture as default that triggers the navigation to the TaskEditorIS. 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 task, or instead delete it from the list. The options menu contains two XisMenuItems that allow creating a task by navigating to the TaskEditorIS and deleting all tasks.

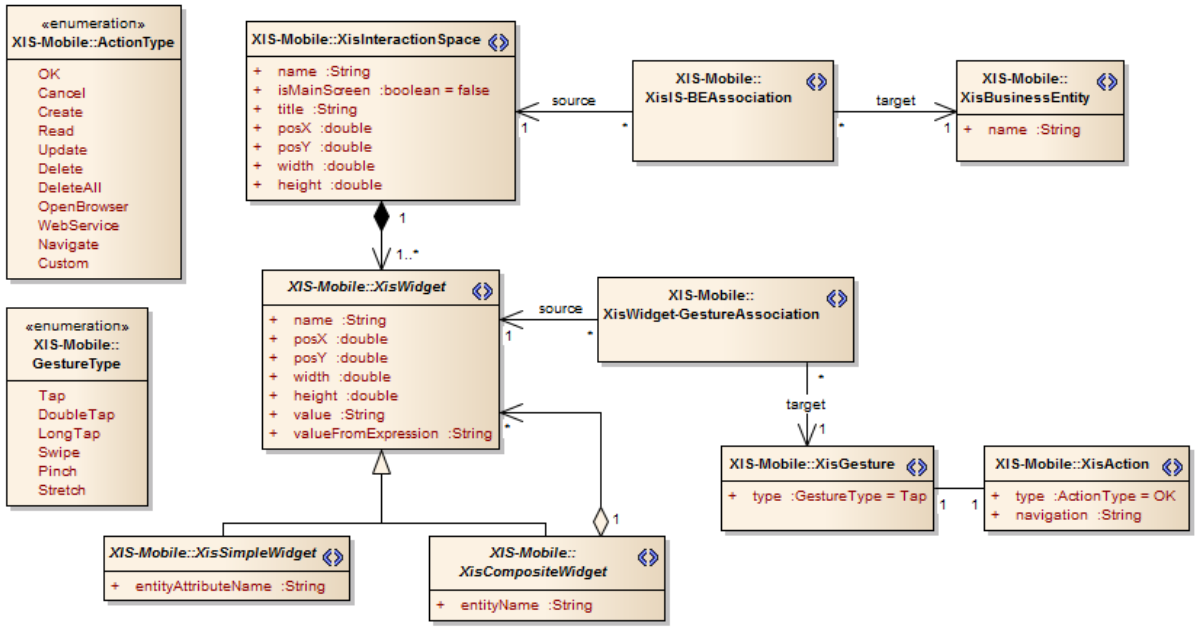



**Figure 25.** Excerpt of the InteractionSpace metamodel.

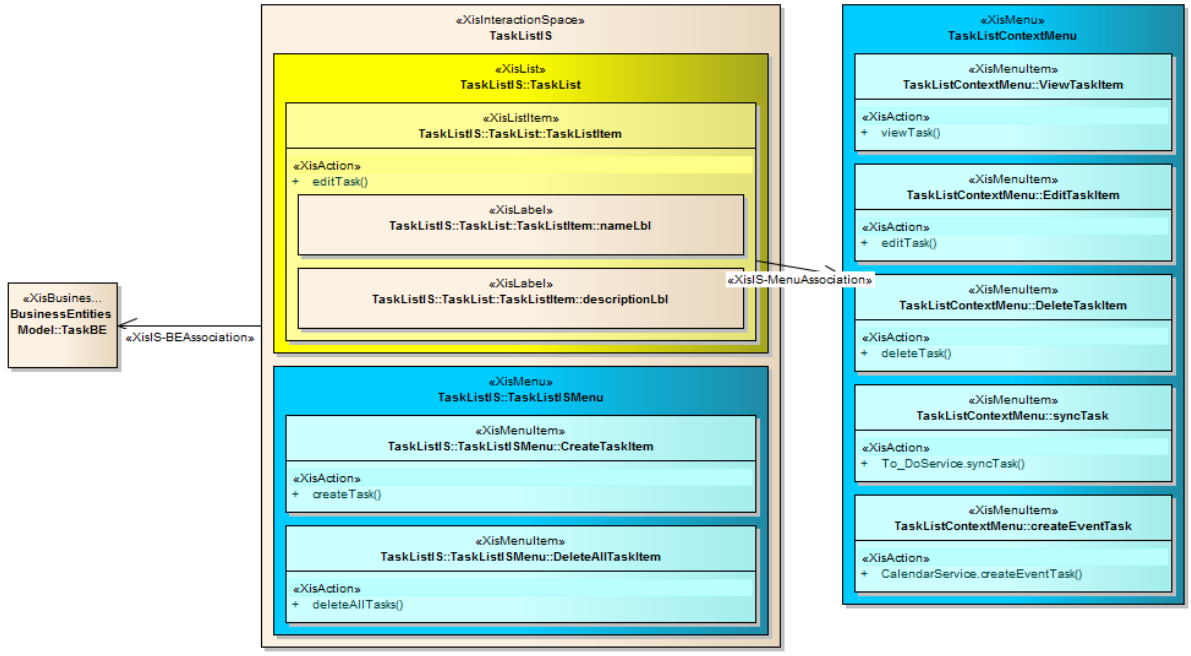



**Figure 26.** The TaskListIS interaction space of the To-Do List App.

# 5. XIS-Mobile Framework

This chapter describes the XIS-Mobile framework, i.e., the integrated environment that supports the Model-Driven Development (MDD) of mobile applications using the XIS-Mobile language. First, Section 5.1 provides an overview of the XIS-Mobile framework. Namely, describing its architecture, used technologies and the proposed development process. Section 5.2 explains how EA has been customized and tailored in order to allow design models using the XIS-Mobile language. Then, Section 5.3 delves into the details of how Model-to-Model (M2M) transformations are performed in the XIS-Mobile framework. At last, Section 5.4 focuses on explaining how the Model-to-Text (M2T) transformation stage has been implemented in the XIS-Mobile framework.

## 5.1. Overview

The XIS-Mobile language is supported by a MDD-based framework[17] that makes this proposed DSL more relevant and allows its use. As shown in Figure 27, the suggested development process of a mobile application using the XIS-Mobile framework consists in four steps: (1) the definition of the required views using the Visual Editor; (2) their validation with the Model Validator; (3) the generation of the User-Interfaces View models with the Model Generator; and finally (4) the generation of the application's source code through the Code Generator. After step (3), if the designer finds that the model is not yet completed, the process returns to step (1) for a new iteration. It is important to emphasize that only step (1) is manual, while the other three are automatic.

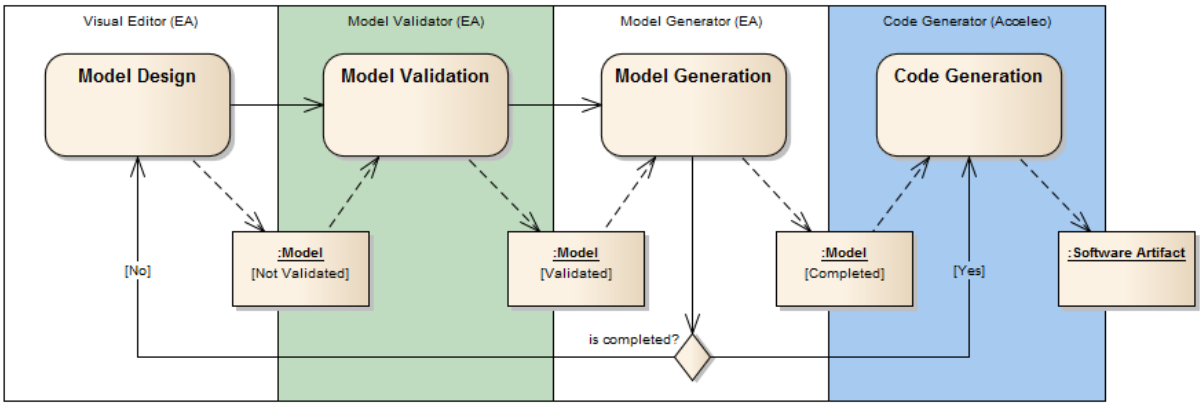



**Figure 27.** Suggested development process with the XIS-Mobile framework.

The XIS-Mobile framework is implemented using the Model Driven Generation (MDG) Technologies provided by Sparx Systems Enterprise Architect (EA) along with the Eclipse Modeling Framework (EMF). The use of these technologies leverages the environment they provide, as well as some compatible plug-ins. As can be noticed by the suggested development process, the XIS-Mobile framework is composed of four modules: (1) Visual Editor; (2) Model Validator; (3) Model Generator; and (4) Code Generator.

[17] The XIS-Mobile framework is available in https://github.com/xis-mobile.

First, the Visual Editor is implemented on top of EA through the use of an MDG Technology plug-in specifically created for the XIS-Mobile language. The use of EA's MDG Technologies allowed the definition of the XIS-Mobile language as a UML profile fully compliant with the OMG specification for UML2, and the creation of toolboxes, diagram types, model patterns and model templates customized to the XIS-Mobile language. In a preliminary version, the XIS-Mobile's Visual Editor has been implemented using Papyrus. However, Papyrus proved poorly suited to model with a language of the complexity of XIS-Mobile. Namely, it revealed usability problems, such as the lack of support of proper modeling of XisInteractionSpaces that contain several elements, and bugs that hinder the modeling process.

Second, the Model Validator is implemented as a plug-in using EA's Model Validation API[18]. It plays a crucial role avoiding errors by the designer, improving the quality of the models and, consequently, enhancing the quality of the generated models and code. When thinking about UML models validation, OCL(OMG, 2014) 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 would not be fruitful for now. Thus, despite not being a standard like OCL, this solution allows the definition of constraints or rules, custom error messages for each constraint, the assignment of severity levels (error or warning) to them, and the immediate navigation to the element that broke the rule and caused the message.

Third, the Model Generator is implemented as a plug-in leveraging EA's Automation Interface[19]. It permits accessing the repository containing the created diagrams and their elements, as well as creating new diagrams and elements. Thus, the Model Generator performs M2M transformations using the information of the UseCases, Domain, BusinessEntities and Architectural views. Usually a single instance of each one of these diagrams yields to multiple InteractionSpace View diagrams and a single NavigationSpace View diagram. The M2M transformation process is detailed in Section 5.2.

Fourth, the Code Generator is based on Acceleo], a template-based code generator framework available as an Eclipse plug-in. Acceleo implements the MOF Model to Text Language (MTL) standard (OMG, 2008) and allows defining code templates for any kind of model compatible with the EMF. The code templates are composed of regular text (static part of the template) and several annotations (dynamic part of the template) that are replaced by values of the model during generation time. For now, the XIS-Mobile framework supports the generation of applications for Android, and also early versions for Windows Phone and iOS have been developed. An important point is that whenever the user desires to add support for other platforms, he needs to define the corresponding code templates.

Additionally, the models designed in EA are exported in a XMI file before the code generation is performed. This file contains the representation of the designed models and consists in the input of the Code Generator. Because the XMI format used in EA is not directly compatible with the XMI format supported by the EMF and consequently expected by Acceleo, it was necessary to process the exported model before starting the code generation. This conversion has been implemented using a

[18] http://goo.gl/XwUvsx (Accessed on October 2014)
[19] http://goo.gl/r9jz0Z (Accessed on October 2014)

Java program that replaces EA's XMI namespaces by EMF's XMI namespaces, stores the positioning and size of each InteractionSpace View element by setting the respective tagged values (posX, posY, width and height of the XisWidget stereotype) using the layout information stored by EA's XMI format and finally removes additional irrelevant data stored by EA. Setting these tagged values is very important, because it is based on them that the layout source code files are generated. They assure that the positioning and size of the modeled XisWidgets is roughly the same as the generated application.

## 5.2. Visual Editor

As mentioned before, the Visual Editor of the XIS-Mobile Framework has been developed leveraging the features provided by EA Model Driven Generation (MDG) Technologies. More precisely, a MDG technology specifically created for XIS-Mobile has been created in order to tailor the EA environment to the XIS-Mobile language. Additionally to the XIS-Mobile profile definition, the implemented MDG technology contains the defined custom diagrams, custom toolboxes, model patterns, a project template and custom quicklinks. Each one of these components is described below.

Regarding the definition of the XIS-Mobile diagrams, EA allows the definition of custom diagram profiles[20] and their inclusion in a MDG technology. This custom diagram profile provides specific metaclasses for each type of diagram (e.g. the metaclass Diagram_Logical represents a Class diagram) and allows associating custom toolboxes to the custom diagrams. The Profile diagram of the XIS-Mobile's diagrams is represented in Figure 28.

The XIS-Mobile diagrams toolboxes have also been implemented through the creation profile diagrams. In this case were created six profile diagrams (one for each view of XIS-Mobile) that were then included in the implemented MDG technology. EA provides a custom toolbox profile[21] containing the metaclass ToolboxPage that allows defining the elements and associations of a custom diagram toolbox. It is also possible to associate icons to specific toolbox items through the use of the metaclass ToolboxImageItem. Figure 29 shows the profile diagram for the toolbox of the Architectural View of XIS-Mobile.

EA also allow the specification of model patterns[22] and their association to toolboxes, i.e., toolboxes can have model patterns as items. Basically a pattern consists in a fragment of model that is usually used. In the case of XIS-Mobile, there were define to patterns: one for XisMenus and other for XisLists. Both of them are available to use through the InteractionSpace View toolbox. Using a Design Pattern enables you to rapidly create template solutions for code structures that perform the same type of task in other situations, and to use items defined in the Pattern with the UML model.

[20] http://goo.gl/pVAwTx (Accessed on October 2014)
[21] http://goo.gl/Xsntgw (Accessed on October 2014)
[22] http://goo.gl/APOGMY (Accessed on October 2014)

Models of project templates[23] can also be created using EA MDG Technologies. After being added to the MDG technology, these templates can be chosen from the list of standard model templates already provided by EA. For XIS-Mobile, it was only created on model template that creates the package structure with all the views and respective diagrams of a XIS-Mobile project. Its use can be seen in Figure 30 and the project structure is highlighted in red.

At last, there were also defined custom Quick Linker menus[24] for the XIS-Mobile elements. This is specified by adding to the UML profile diagram a QuickLink Document Artifact element that contains the Quick Linker connections for each element and diagram in a CSV (Comma-separated values) format. This was also created in XIS-Mobile with the goal of reducing the errors, since it restricts the type of associations and elements that be created from a given element.

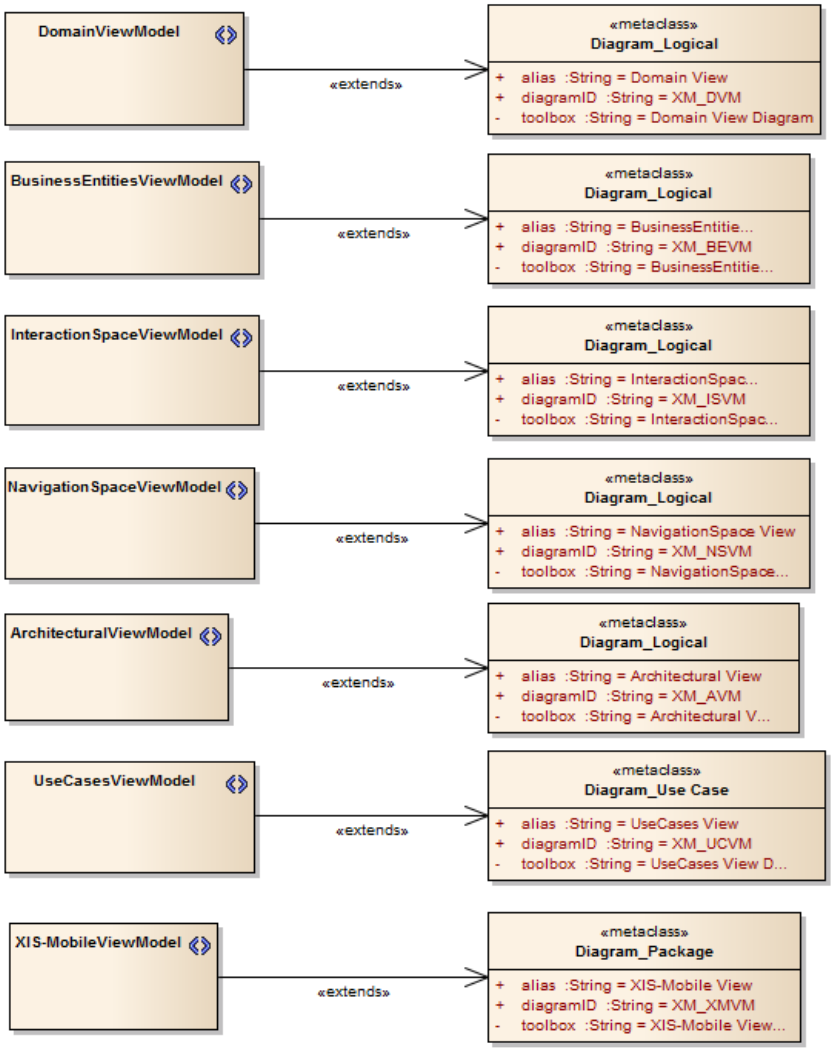



**Figure 28.** Profile diagram of XIS-Mobile's diagrams.

[23] http://goo.gl/GH81xe (Accessed on October 2014)
[24] http://goo.gl/VcG9pA (Accessed on October 2014)

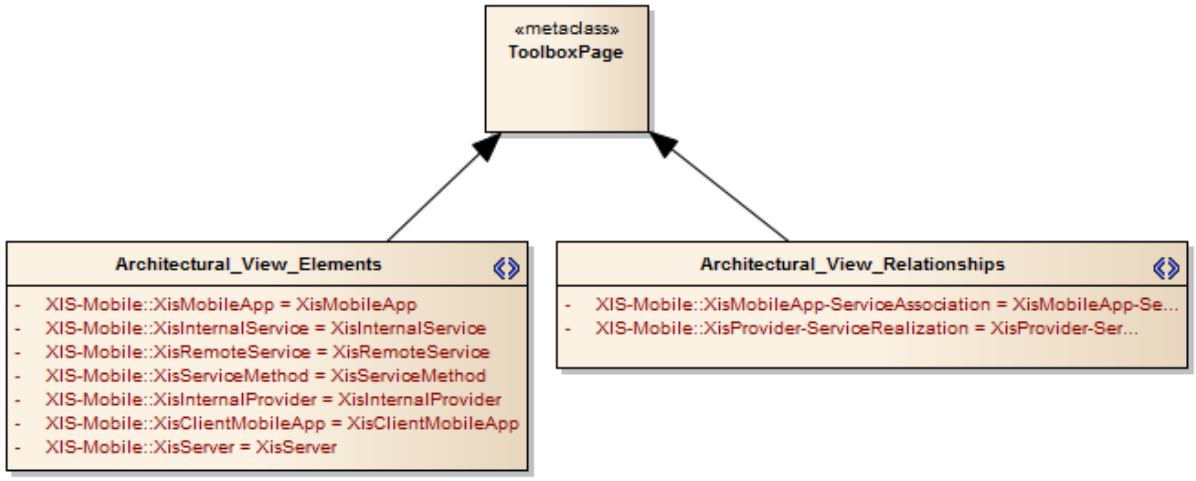



**Figure 29.** Profile diagram of the toolbox of the Architectural View of XIS-Mobile.

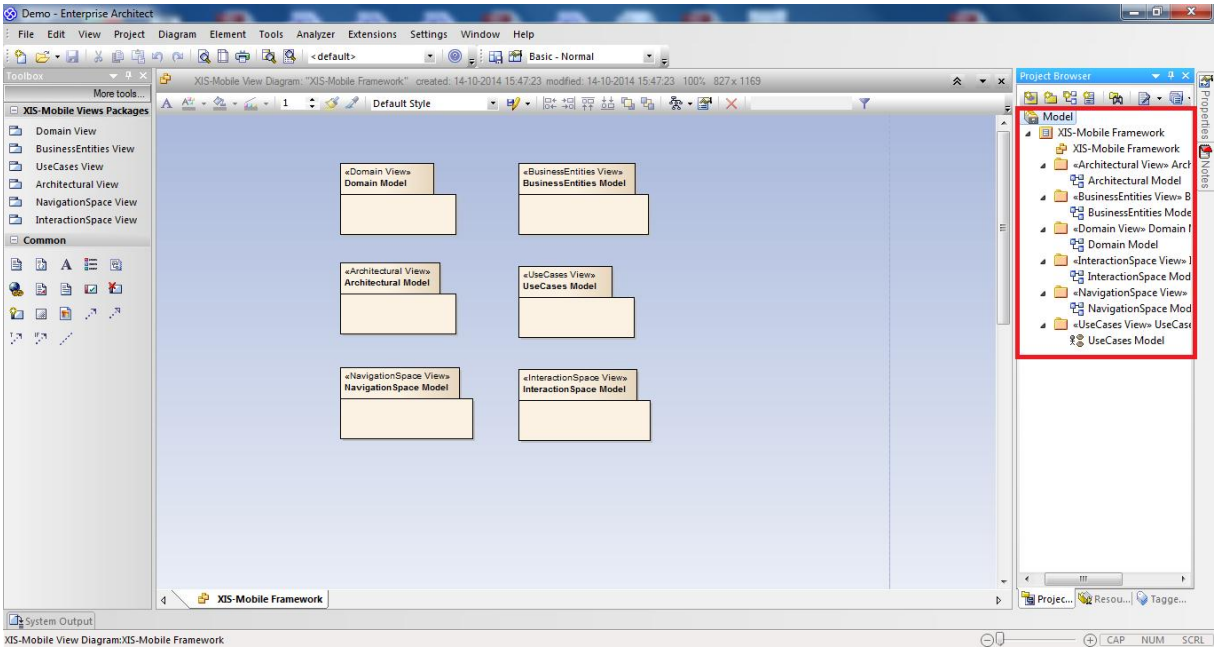



**Figure 30** Example of the XIS-Mobile project structure.

# 5.3. Model-to-Model Transformations

The XIS-Mobile's M2M transformations stage has been implemented with the goal of offering support to the proposed smart design approach and proving its feasibility, as proof of concept. This approach leverages M2M transformations to automatically generate the most laborious views of the XIS-Mobile language, the NavigationSpace and InteractionSpace views. In this case, the M2M transformations do not convert the designed models into other ones, but rather create additional new models based on them. Specifically, the M2M transformations use the information defined in the Domain, BusinessEntities, Architectural (if defined) and UseCases views to generate the NavigationSpace and InteractionSpace views. M2M transformation is triggered in the XIS-Mobile framework by selecting the option "Generate Models" in the XIS-Mobile Plugin menu. Figure 31 shows the launching of this

process. All the M2M transformations are implemented in C# code using EA's Automation Interface that allows manipulating the repository contain the models.

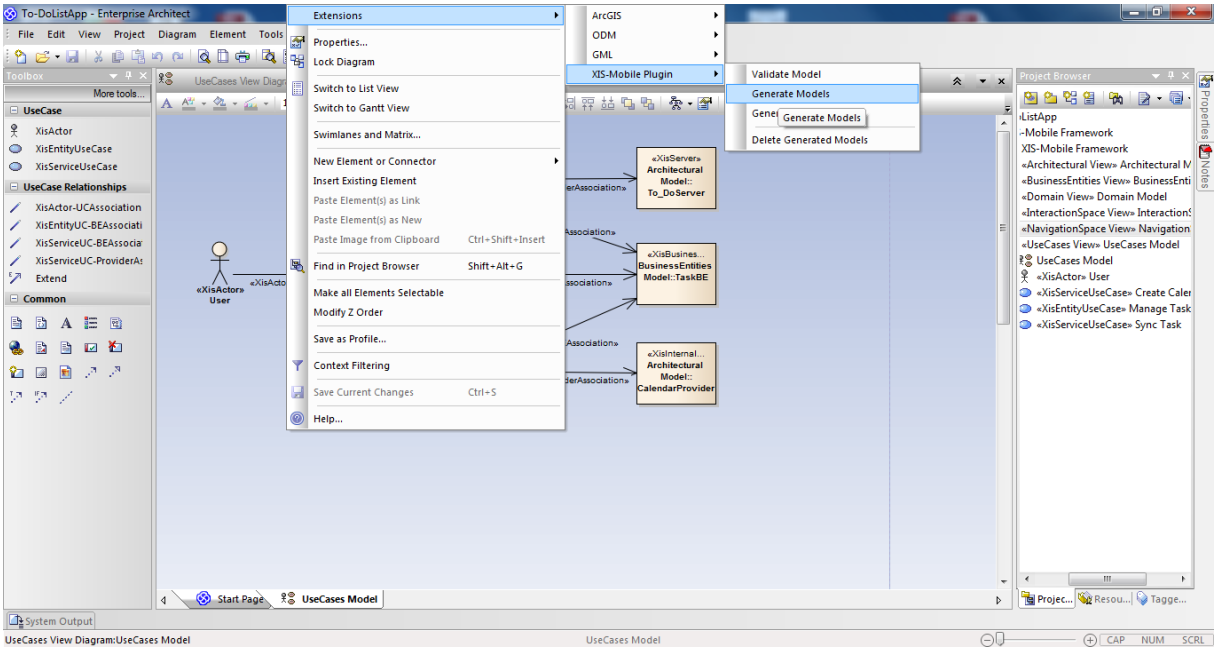



**Figure 31.** Launching of the M2M transformations in the XIS-Mobile framework.

The UseCases View is the "cornerstone" of the XIS-Mobile's M2M transformation stage, because it provides the abstractions (stereotypes, tagged values and associations) that configure this process. Namely, the XisUseCases and their relationships capture the UI patterns described in Section 2.3, and therefore determine the type of models that will be generated. The patterns of generation supported by specific XisUseCases are described subsequently in detail. The generated NavigationSpace diagrams have been omitted for reasons of simplicity, better comprehension and space-saving.

### XisEntityUseCase

A XisEntityUseCase is, as its name suggests, a use case that represents an action over domain entities (XisEntities). For this reason, a XisEntityUseCase must be connected to a business entity (XisBusinessEntity). In turn, the business entity allows the binding of the use case with the domain entities, since it aggregates one or more domain entities. The business entity also assigns roles to each one of the entities it aggregates. These roles define domain entities as Master, Detail and Reference. A Master domain entity is the main domain entity manipulated by the XisEntityUseCase and the one around which the User-Interfaces Models will be generated. Each business entity only defines one Master domain entity. Detail and Reference domain entities represent domains entities that are associated to the specified Master through aggregations and associations respectively. Thus, they are considered as additional information of their Master domain entity.

A XisEntityUseCase contains a set of tagged values representing the CRUD (Create, Read, Update and Delete) and the Search operations for the master, detail and reference entities. These tagged values function as flags and if they are set to true, they will result in the appearance of options (menus, menu items, buttons, and interaction spaces) that will be present in the generated models.

Moreover, a XisEntityUseCase has a tagged value that defines its type and consequently the types of interaction spaces that will be generated. A XisEntityUseCase has the following types: (1) EntityManagement and (2) EntityConfiguration. Figure 32 shows an example of a configuration of a XisEntityUseCase with EA with the tagged values view highlighted in red.

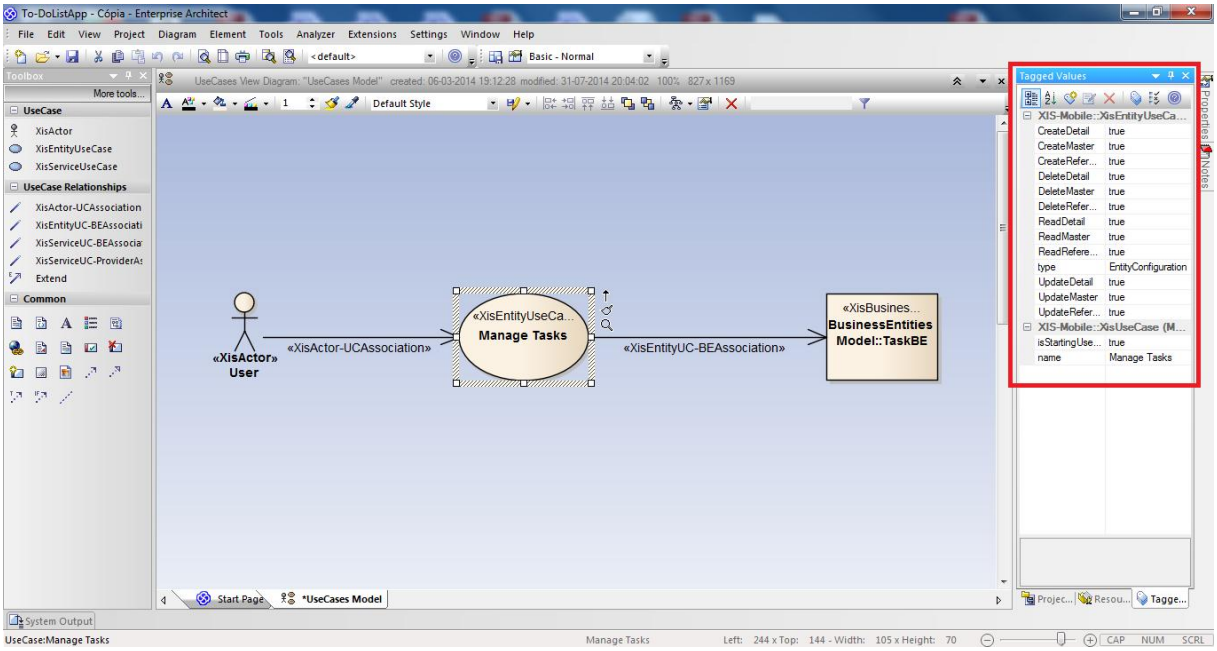



**Figure 32.** Example of a UseCases diagram containing a XisEntityUseCase in the XIS-Mobile framework.

A XisEntityUseCase of type “EntityManagement” will generate interaction spaces for managing a list of multiple instances of the Master entity (from the associated business entity). From now on, this type of M2M transformation is referred as Pattern 1. Using the To-Do List App diagrams and the UseCases diagram of Figure 32. (with the tagged value “type” set to “EntityManagement”) will result in the generation of four screens (XisInteractionSpaces). Figure 33 shows the first three XisInteractionSpaces generated in this case. The fourth screen has been omitted from this figure for simplicity and due to space and visibility restrictions, but it can be seen in Figure 35 that is described later. Moreover, the associations between interaction spaces and business entities have been omitted as well.

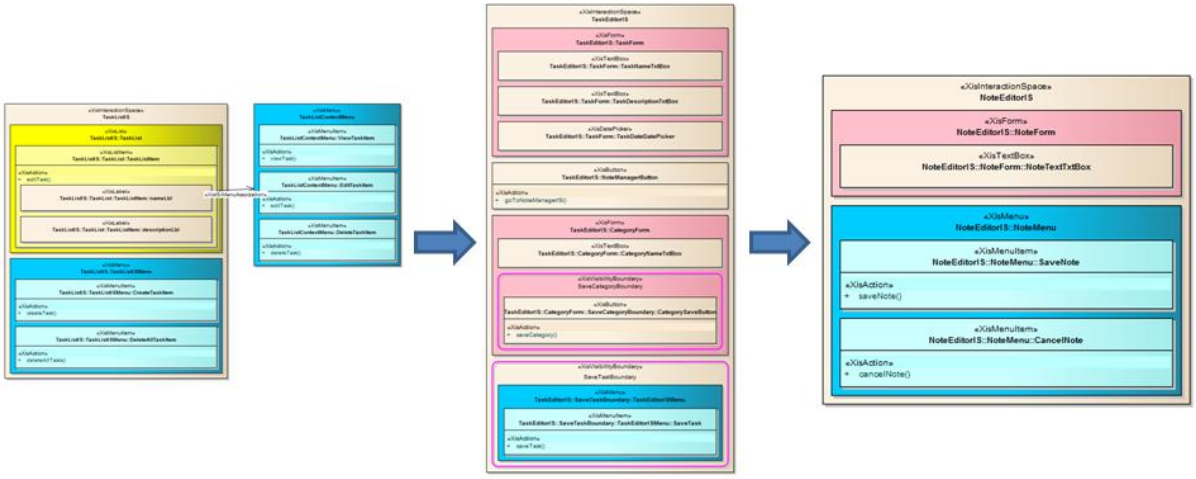


**Figure 33.** Example of the interaction spaces generated from a XisEntityUseCase of type “EntityManagement.”

The first screen allows managing all the instances of the Master entity (in this case, a Task). The second screen is an editor screen of the Master entity (Task) and its Detail entities (Note) and Reference entities (Category). Since a Task can have one or more Notes, the edition of Notes would potentially occupy great part of the screen space, which by itself is already small in mobile applications. Thus, in these cases are created buttons that allows navigating to dedicated editor screens of the Details or Reference entities. In this case of the To-Do List App is just required to apply this pattern for the Details, since a Task can only be associated at most with one Category. Therefore, the third screen allows managing the list of Notes associated to a Task. At last, the fourth screen allows editing a given Note of a given Task. Figure 34 shows the same three screens that will be generated from the models of Figure 33 for Android.


**Figure 34.** Example of the interaction spaces generated from a XisEntityUseCase of type “EntityManagement” in Android.

In turn, a XisEntityUseCase of type “EntityConfiguration” will generate interaction spaces to manage a single instance of the associated Master entity. This case consists in Pattern 2. In fact, the interaction spaces generated in this case are almost the same as the ones of Pattern 1. Specifically, the only

difference is that the first screen of Pattern 1 is not generated, because in Pattern 2 there is a single instance of the Master entity. Again, using the To-Do List App diagrams and the UseCases diagram of Figure 32 will result in the generation of three screens. The representation of these three generated XisInteractionSpaces is depicted in Figure 35. In turn, Figure 36 shows their representation in Android.

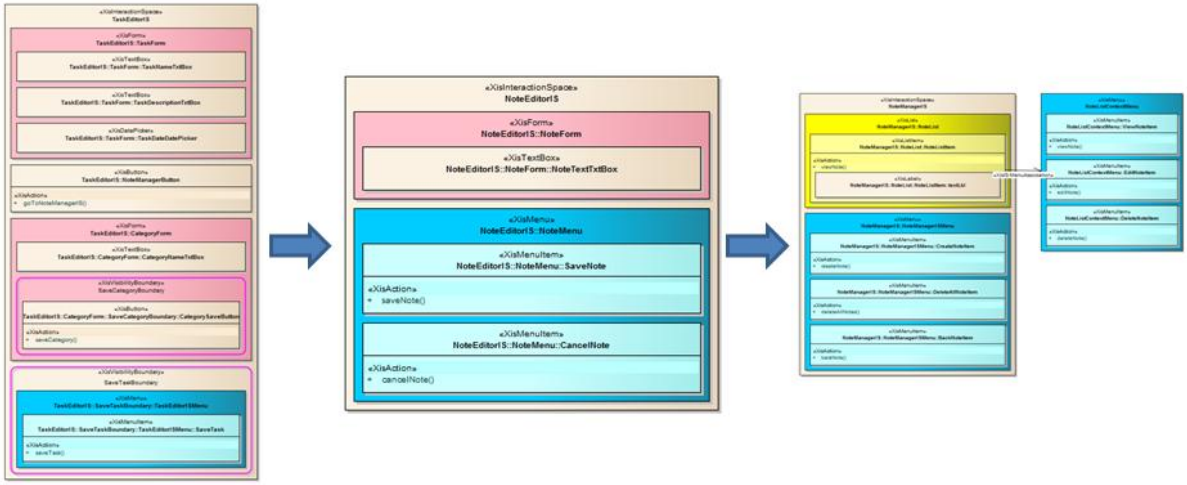


**Figure 35.** Example of the interaction spaces generated from a XisEntityUseCase of type “EntityConfiguration".


**Figure 36.** Example of the interaction spaces generated from a XisEntityUseCase of type “EntityConfiguration”.

### XisServiceUseCase

A XisServiceUseCase represents an action that uses the operations (XisServiceMethods) provided by a provider (XisProvider defined in the Architectural View). For this reason, a XisServiceUseCase must be connected to a provider and can also be connected to a business entity, representing that the operations have an effect over the business entity’s domain entities.

A XisServiceUseCase will generate an interaction space that makes use of the operations provided by the provider and its associated services. The generated interaction space is composed of an Options Menu (XisMenu of type “OptionsMenu”) with those operations as menu items (XisMenuItem).

If the XisServiceUseCase is associated to a business entity, the generated interaction space is not only composed of an Options Menu with the provider services operations represented as menu items, but also presents the details of the associated domain entities. Figure 37 shows an example of a UseCases diagram that uses this pattern in the To-Do List App context. In turn, Figure 38 shows the interaction space generated from the previous diagram.

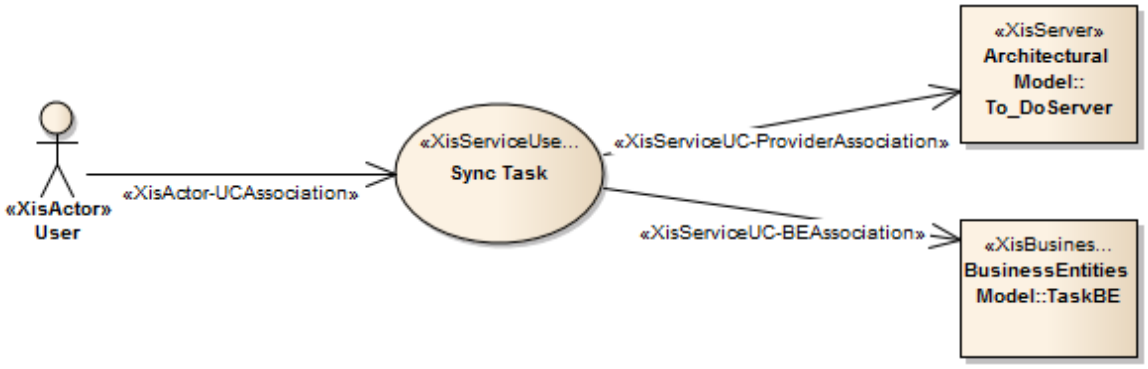



**Figure 37.** Example of UseCases diagram with a XisServiceUseCase.

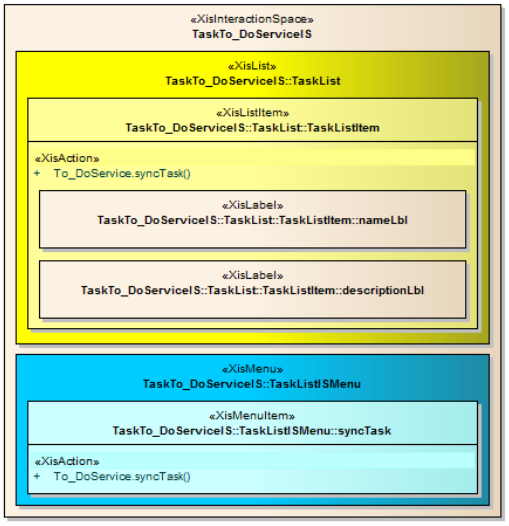



**Figure 38.** Example of an interaction space generated from a XisServiceUseCase.

These two patterns of M2M generation using a XisServiceUseCase need to be improved, possibly through the definition of types of XisServiceUseCase, similarly to what is done with a XisEntityUseCase. Thus, this represents a future research direction.

### XisEntityUseCase extension with XisServiceUseCases

XIS-Mobile also allows the extension of a XisEntityUseCase with one or more XisServiceUseCases. This causes the addition of options (menu items) that use the provider services operations to the menus (Context and Options) generated in context of a XisEntityUseCase. Figure 39 illustrates an example of such a UseCases diagram for the To-Do List App where a XisEntityUseCase is extended by two XisServiceUseCases. In turn, Figure 40 shows an interaction space that was generated from

the previous diagram. The extensions made by the two XisServiceUseCases to this interaction space are highlighted in red.

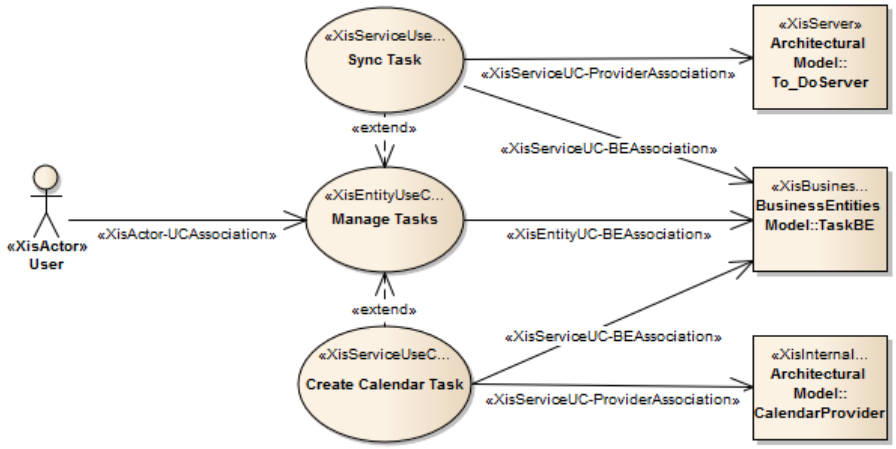



**Figure 39.** Example of the extension of a XisEntityUseCase by two XisServiceUseCases.

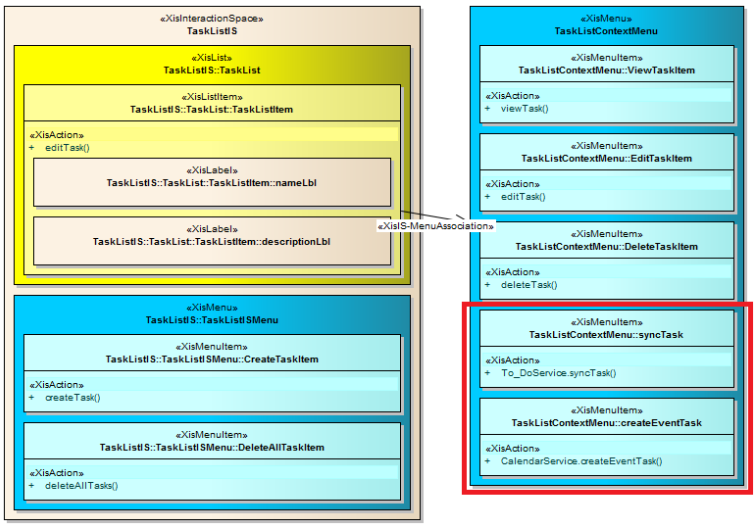



**Figure 40.** Example of an interaction space generated from the extension of a XisEntityUseCase by two XisServiceUseCases.

### Multiple independent XisUseCases

Another pattern of generation considered in XIS-Mobile is applied when one or more XisUsesCases (either XisEntityUseCase or XisServiceUseCase) are not related. This pattern obliges the specification of a navigation pattern that will define the home screen (first interaction space) of the application. To deal with this pattern, the XIS-Mobile framework opens a window before the M2M transformation process starts, where the user must select the navigation pattern he wants to apply (see Figure 41). Otherwise, it would be not possible to navigate to the interaction spaces generated from non-related XisUseCases and therefore the application navigation would be broken. For now, the navigation patterns applied are: Springboard and List. Both of them have been described in Section 2.3. In the future, we plan to add support for further navigation patterns, namely for the Tab Menu also described in Section 2.3. Figure 42 illustrates a UseCases diagram that comprises three independent

XisUseCases in the context of the To-Do List App. In turn, the two possible home interaction spaces generated from the previous diagram are depicted in Figure 43 and Figure 44.

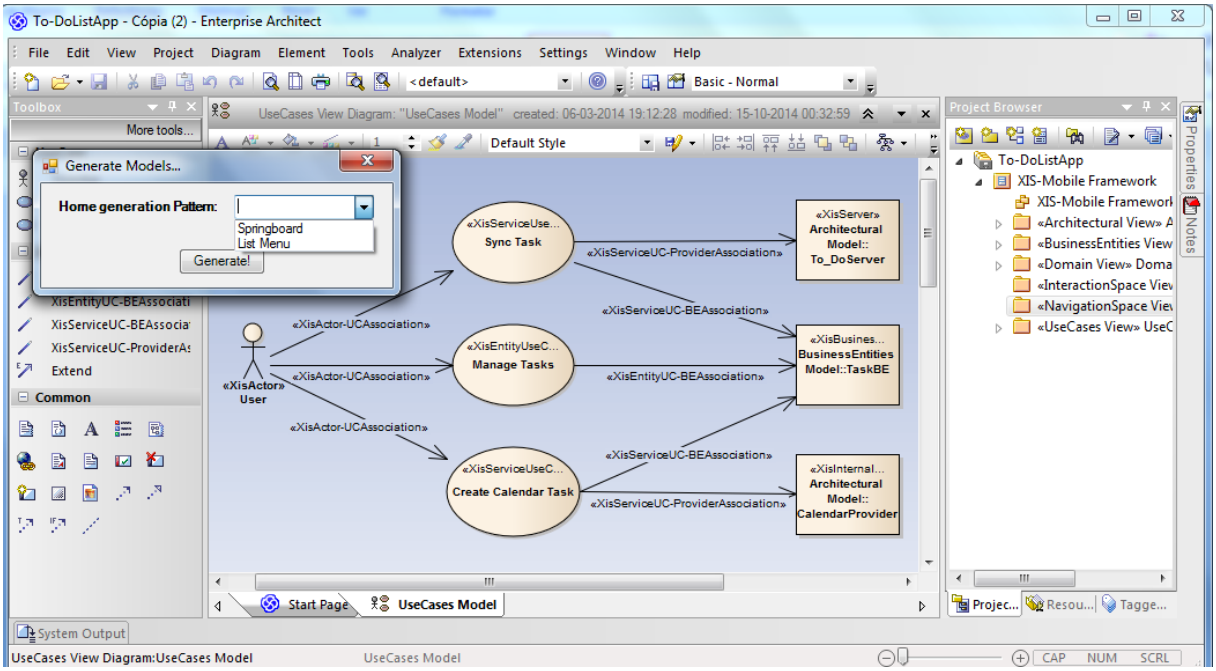



**Figure 41.** Selection of the navigation pattern to apply in M2M transformations in the XIS-Mobile framework.

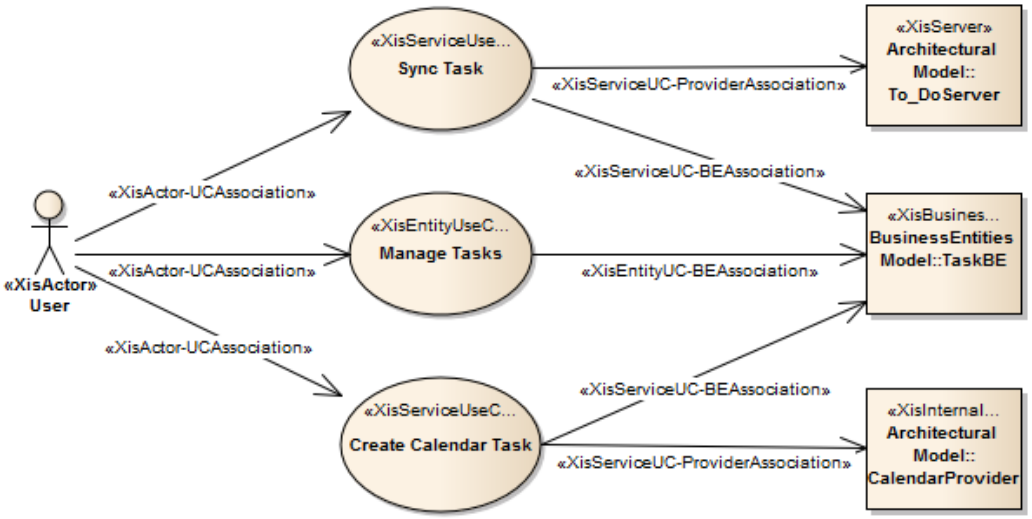



**Figure 42.** Example of a UseCases diagram with three independent XisUseCases.

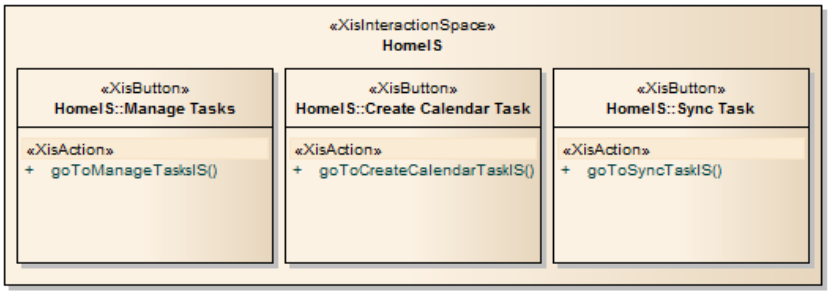



**Figure 43.** Example of an interaction space using the Springboard pattern.

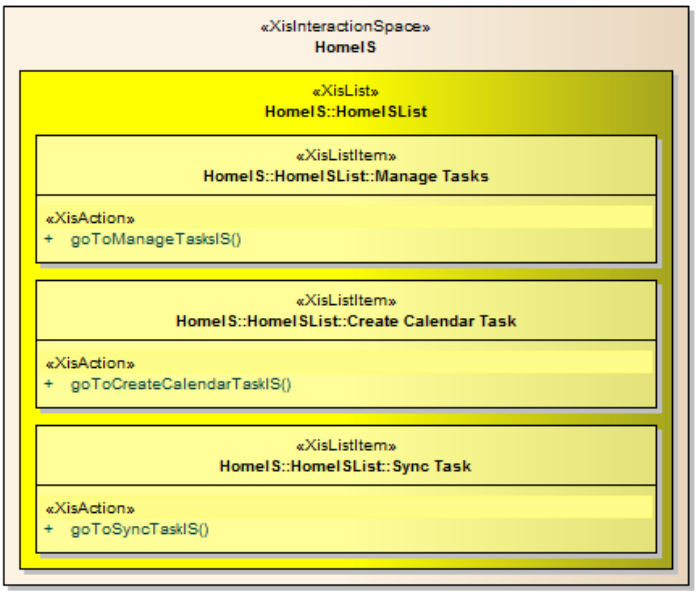



**Figure 44.** Example of an interaction space using the List pattern.

# 5.4. Model-to-Text Transformations

The XIS-Mobile's M2T transformations stage is fully implemented using Acceleo, a plugin provided by Eclipse. Acceleo is an open source framework that allows the creation of template-based code generators. Acceleo was created by Obeo[25], a french company specialized in providing software solutions to create and transform complex systems like industrial systems (e.g. avionics, space or defense), or IT applications. Acceleo was released in 2006 as a plugin for Eclipse, and in 2009 included as in Eclipse's Model To Text (M2T) project. Acceleo implements the MOF Model to Text Language (MTL) standard. Acceleo allows defining code templates (also known as modules and MTL files) for any kind of EMF-based model, including UML models. As mentioned before, the code generation mechanism is based on templates. A template is a special file composed of regular text (static part of the template) and several annotations (dynamic part of the template). These annotations are substituted by values of the model elements during generation time.

Acceleo templates are composed of two major types of constructs: templates and queries. Templates are sets of Acceleo statements used to generate any kind of text. They are delimited by the [template …][/template] tags. In turn, queries do not generate text. They are used to extract information from the model. They always return values and are specified with [query …/] tags that use OCL inside. Additionally, Acceleo allows the use of java service wrappers when it is not possible to define the intended template using templates or queries constructs. A java service wrapper is a java file that allows navigating the model. The Acceleo non-standard library provides a service invoker which allows invoking the java service as traditional query. Table 4 shows an excerpt of an Acceleo template for generating java classes. In this case, it was assumed that the input UML model contains one class named "Task" with three attributes (title, description and date).

[25] http://www.obeo.fr/en/ (Accessed on October 2014)

**Acceleo Code Template:**

```
[module generateClass('http://www.eclipse.org/uml2/4.0.0/UML')]
[template public generateClass(c : Class)]
[comment @main /]
[file (c.name.toUpperFirst().concat('.java'), false, 'UTF-8')]
     public class [c.name.toUpperFirst()/] {
     [for p : Property | c.attribute separator('\n')]
           private [p.type.name] [p.name];
     [/for]

           public [c.name.toUpperFirst()]() { }
     }
[/file]
[/template]
```

**Sample Output:**

```
public class Task {
     private String title;
     private String description;
     private Date date;

     public Task() { }
}
```

**Table 4.** Example of a simple Acceleo template and a possible output.

Moreover, Acceleo provides a template editor that allows editing templates in an easier and assisted way, a profiler that allows analyzing the quality of the generation, and a debugger to test, place breakpoints and to move between instructions of templates during generation time.

The code templates defined for the XIS-Mobile framework have been developed following Acceleo's best practices, particularly in terms of naming conventions, project and module organization and java services use. Figure 45 shows the organization of the Acceleo project of the XIS-Mobile's code generator for Android. The naming convention follows the Acceleo recommendation: <project name><kind of input><input metamodel name>gen<output language name>. The differences between the packages are described below:

- **common:** This package contains the utility modules, i.e., the modules with the queries commonly used by templates. For example, this package contains a file with several queries that check the XIS-Mobile stereotype of a certain UML element;
- **files:** This package contains all the modules that will generate files, i.e., modules like the one exemplified in Table 4;
- **main:** This package contains the modules with the main templates and their matching launcher class, i.e., the entry points of the generator;
- **services:** This package contains all the modules that make use of Java service wrappers, as well as the corresponding Java files.

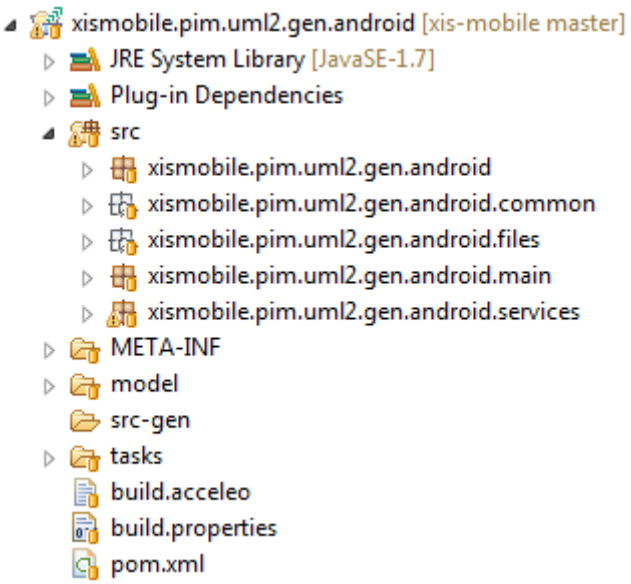



**Figure 45.** Organization of the Acceleo project for Android.

For now, the XIS-Mobile framework supports the generation of applications for Android, and also early versions for Windows Phone and iOS have been developed. This means that the code generator for Android is more complete and mature than the others. The XIS-Mobile framework generates projects ready to be imported by the IDEs used in the development of each platform. The XIS-Mobile framework generates different kinds of files according to the type of platform, ranging from Java, C#, Objective-C, XML, XAML or even configuration files specific of the IDE used. Additionally, it is important to emphasize that whenever the user desires to add support for other platforms, he needs to define the corresponding code templates.

Acceleo proved to be a wise choice due to its complete and updated support of UML profile-based models, like the ones produced using XIS-Mobile. The features mentioned before (editor, profiler, debugger, templates, java services), as well as Acceleo's extensive and good documentation (guides, videos and tutorials) greatly aided the development process. In comparison to other technologies, like JET, Apache Velocity and Xpand, described in Section 2.1.7, Acceleo is a much more active project. An evidence of that, are the constant posts related to Acceleo in the Eclipse dedicated forum for the M2T project[26].

[26] https://www.eclipse.org/forums/index.php/f/24 (Accessed on October 2014)

# 6. Evaluation

This chapter presents and discusses a threefold evaluation performed to XIS-Mobile. First, in Section 6.1, the XIS-Mobile M2T capabilities are evaluated through the analysis of the percentage of lines of code (LOC) generated for case studies A and B (see Section 3.4) for the three supported mobile platforms, Android, iOS and Windows Phone. Then, Section 6.2 presents the results of an assessment session of XIS-Mobile by third parties. This assessment focused on three aspects of XIS-Mobile: (1) the Language, (2) the Framework and (3) the General Approach. Section 6.3 compares XIS-Mobile with some related work and at last Section 6.4 summarizes the chapter describing the main conclusions obtained from this threefold evaluation.

## 6.1. Case Study Implementation

This section presents and analyzes the results obtained from the implementation of two case studies presented in Section 3.4: (A) the To-Do List App and (B) the Tourism App, using XIS-Mobile. Table 5 presents the quantitative evaluation by comparing the generated code against the complete manual implementation for the Android, iOS and Windows Phone platforms. Case study B was not implemented for Windows Phone and for that reason the ratio was not calculated in this case. The metric used to perform this evaluation was the ratio between the lines of code (LOC) generated and the lines of code of the full version of the application (including with data initialization). This ratio allows measuring the amount of code generated for the two case studies, in percentage. The measurements were performed using the open source project CLOC – Count Lines of Code[27].

| | To-Do List App | | | Tourism App | | |
|---|---|---|---|---|---|---|
| **Platform** | **Generated** | **Overall** | **Ratio (%)** | **Generated** | **Overall** | **Ratio (%)** |
| Android | 1132 | 1280 | 88.4 | 2089 | 2922 | 71.5 |
| iOS | 675 | 1053 | 64.1 | 1266 | 1561 | 81.1 |
| WP | 1197 | 1502 | 79.7 | | | |

**Table 5.** LOC results for the different platforms and case studies.

Regarding case study A, the results showed that the generation for iOS underperformed, since only 64.1% of the application has been generated. In turn, the percentage of code generated for Android and Windows Phone was quite positive in this case study, because both of them presented a ratio greater than to 79%. This can be explained, because XIS-Mobile generates an optimized version of the application for iOS, i.e., compared to the manual implemented version by Mats Sandvoll (Sandvoll, 2014), the generated one combines into a single editor screen the screens to create, edit and view the details of a task. This is a point that must be fixed in the future, in order to better assess the generation for iOS. Figure 46 shows the home screen of case study A in Android, iOS and Windows Phone.

[27] http://cloc.sourceforge.net (Accessed on October 2014)

Regarding case study B, the results for the Android platform were not so good (under 72%). This happens, because this case study has much more screens, domain entities and, consequently, a more complex business logic that could not be generated. Namely, the initialization and manipulation of all the POIs, categories and tours caused this decrease when compared with case A. Also more enhancements in the UI layout were needed, due to the type of screens it contains (e.g. with map views, images and search boxes). For instance, the drawing of the tours in the map could not be generated. However, the percentage of code generated for the iOS platform was quite good (above 81.1%). This can be due to the fact of the templates of the iOS generator are much more static and less generic than the ones for Android. Figure 47 shows the home screen of case study B in Android and iOS.

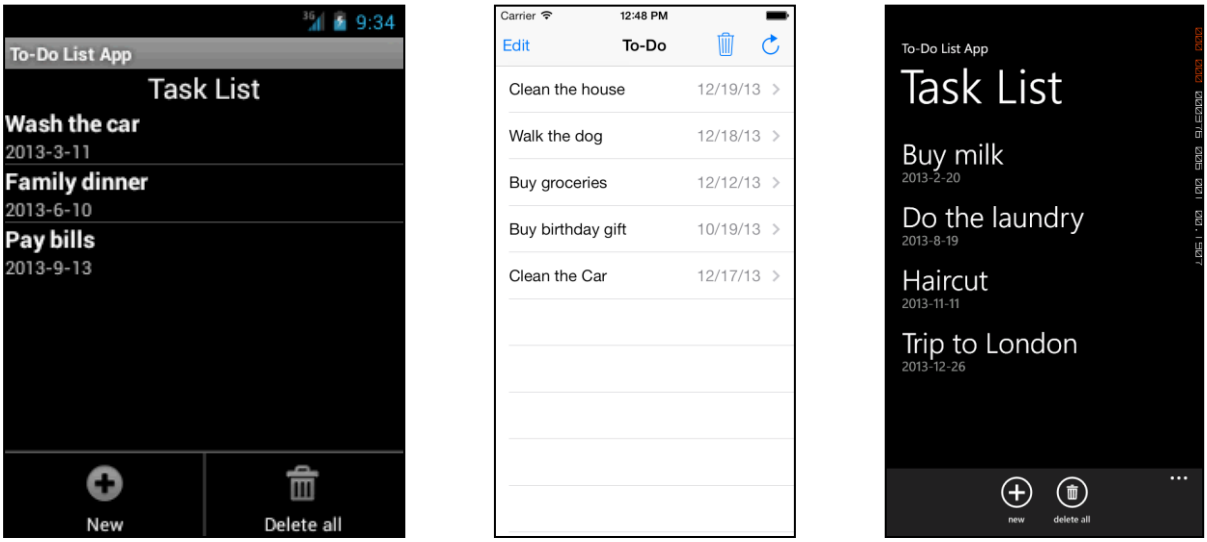



**Figure 46.** Home screen of the case study A in Android, iOS and Windows Phone (from left to right).

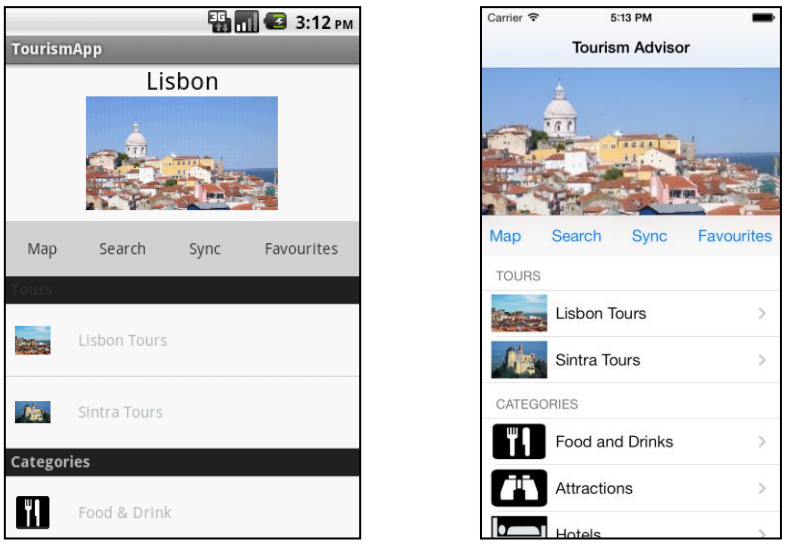



**Figure 47.** Home screen of the case study B in Android (left) and iOS (right).

Summarizing, the results were good on average, especially for a proof of concepts. However, it can be noticed that the percentage of code generated for the case study A was greater than the one for B,

except for iOS. Therefore, this may allow us to conclude that case study A was better captured by XIS-Mobile than the case B.

During the modeling of both case studies we could verify that case A fits well in the XisEntityUseCase type “EntityManagement” and thus a great majority of the models used to generate the code, were the ones generated. Therefore, case study A has taken a good advantage of the smart design approach. Concerning the case study B, the results were not so encouraging, because does not exist a use case type closer to the requirements of this application. Thus, the modeling of case B has been mostly performed manually, using a mix of the smart and dummy approaches. The goal in the near future is to extract patterns behind the case study B, in order to move from a mostly dummy based approach to the smart approach.

During the implementation of the code generator for iOS, it was found a constraint in the generation of each screen. Each screen’s information cannot be generated in a dedicated file, like in Android and Windows Phone, which use XML and XAML files, respectively. Therefore, all the details related with the UI of an iOS application generated with XIS-Mobile are performed programmatically in code.

## 6.2. User Session Assessment

To better evaluate XIS-Mobile, receive feedback from people not directly involved in this research 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 participants also had experience with mobile application development and 7 had professional experience in Information Technology.

The user session was conducted under the following conditions:

- Session took place in the laboratory (controlled environment);
- The assigned tasks were performed without previous use and learning (for the first time);
- The user must had a computer running Windows, Java and previously installed the Sparx Enterprise Architect (version 7.5, 10 or above);
- Direct Observation, i.e., while users performed the assigned tasks, 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. Also, a demonstration of the development of a simple case study using XIS-Mobile from model design until code generation has been performed. Thereafter, they had a script (see Appendix B) describing the Flight Reservation App described in Section 3.4 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. In the end, participants

were asked to fill in a questionnaire to rate the XIS-Mobile language, its supporting framework and the overall approach. The analysis of the results gathered from these questionnaires is described below.

**Questionnaire Analysis**

The questionnaire used in the user session 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 6 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 one 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-Interfaces View more complex and harder to learn than the other views. This can be explained by the its 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 the 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.

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

**Table 6.** 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 7 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. Questions 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. As seen in Section 5.3, use cases can have different configurations with different types, tagged values and associations to business entities and providers.

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

**Table 7.** 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 8 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 higher when compared to the traditional approach.

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

**Table 8.** 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 9, the results were generally encouraging with positive scores in all three analyzed aspects. Nevertheless, it was observed that the XIS-Mobile Framework and namely its model generation process had the lowest score and possibly need to be refined in order to improve their simplicity. Moreover it was observed that the learnability of the XIS-Mobile language should be improved, namely in terms of the InteractionSpace View concepts representation.

It can be stated that the number of participants of the session is relatively small. We believe that number is sufficient to take meaningful conclusions, because usability experts have noted that a group of 5 testers is enough to uncover over 80% of the usability problems (Nielsen & Landauer, 1993). 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. The complete results of the questionnaires are provided in Appendix C.

| | XIS-Mobile | | |
|---|---|---|---|
| | Language | Framework | General Approach |
| **Average** | 3.89 | 3.73 | 4.17 |

**Table 9.** Questionnaire's average score (in a scale of 0-5) for each XIS-Mobile Framework aspect.

# 6.3. Related Work Discussion

After presenting the results of the evaluation through case studies implementation and through a user session, it is time to compare XIS-Mobile with some of the related research work.

Similarly to XIS-Mobile, some works propose and justify the relevance of using UML profiles and MDD to abstract the application development and to achieve platform independency. Unfortunately, none of them represent a real alternative to XIS-Mobile. For example, (Boudaa, Camp, Hammoudi, & Chikh, 2012) is more focused on mobile context-aware applications, while MAM-UML (Belloni & Marcos, 2004) targets mobile-agent applications addressing concepts like code mobility.

Several Cross-Platform Tools for mobile application development, like PhoneGap or Appcelerator Titanium, have appeared in the last years and are widely used to overcome the mobile platform fragmentation. Surveys like the ones presented in (Ribeiro & da Silva, 2013)(Singh & Palmieri, 2011)(Paananen, 2011) 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 (Ranabahu, Maximilien, Sheth, & Thirunarayan, 2011) 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) (Gamma, Helm, Johnson, & Vlissides, 1995)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 (Kramer, Clark, & Oussena, 2010) proposes to achieve portability through the use of a virtual machine (VM) and thus, it 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 common 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 and possible integration with other tools. In addition, MobiA states is focus on the development of mobile health monitoring applications for non-expert users (Balagtas-Fernandez, Tafelmayer, & Hussmann, 2010).

The Cameleon Reference Framework (Calvary, 2002) 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 M2T transformations in XIS-Mobile. A lot of work has been done based on the Cameleon Reference Framework, being UsiXML (Vanderdonckt, 2005), IDEALXML (Vanderdonckt & Simarro, 2010), UsiXML4ALL(Trindade & Pimenta, 2007) and MARIA (Patternò, Santoro, & Spano, 2009) 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 M2M 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 can generate a considerable part of it, mainly through its concept of XisAction. MARIA focuses on service-oriented applications in ubiquitous environments. Like XIS-Mobile, MARIA takes into account notions like gesture 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 (Wolber, App inventor and Real-World Motivation, 2011), highly abstracted 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. It has also been discontinued by Google and its support transferred to the MIT Center for Mobile Learning.

## 6.4. Summary

This chapter presented the results of the threefold evaluation performed to XIS-Mobile. First, through the comparison in terms of LOC between the manual implementation and the automatic generation of case studies A and B described in Section 3.4. This evaluation was essentially a quantitative evaluation because it inspects the percentage of code of the overall application code that could be generated. The results obtained were quite satisfactory with percentages of generation above 75%, on average. This confirmed not only the feasibility of the XIS-Mobile approach, but also partially demonstrated that the technologies used to implement that approach were the correct, namely Acceleo that is responsible for the code generation. However, there were identified some points of improvements both in terms of the code templates and in terms of the model representations (especially in the UseCases and InteractionSpaces views).

The second part of the evaluation focused on the assessment of XIS-Mobile by third parties not involved in this research work. This assessment was done by conducting a user session where the participants had to create a mobile application using the XIS-Mobile framework, from model until code. In the end of the session, the participants filled in a questionnaire focused on three aspects of XIS-Mobile: (1) the Language, (2) the Framework and (3) the General Approach. The results obtained from the questionnaires were generally encouraging with positive scores in all three analyzed aspects. However, it was observed that the participants found the M2M transformation process of the XIS-

Mobile framework the hardest part to understand in the framework. Thus, this suggests that the M2M transformation process may need to be revised or enhanced in order to improve their simplicity and learnability. In addition, it was observed that the learnability of the XIS-Mobile language should be improved, especially in terms of the InteractionSpace View concepts representation.

At last, there were compared some related research work with XIS-Mobile. Despite none of these works constitute a real and effective alternative to XIS-Mobile, it was possible to identify future research direction for this work, namely its extension to support Mobile Cloud Computing (MCC) and context awareness scenarios, the creation of a code generator for a cross-platform tool of type web-to-native wrapper and the evaluation of XIS-Mobile in the academic field and for teaching purposes.

These evaluation results support the thesis of this dissertation, which advocates that a MDD-based approach can mitigate the software development complexity and the mobile platform fragmentation. Moreover, these results showed preliminary evidences that demonstrate XIS-Mobile's usefulness and feasibility.

# 7. Conclusion

Mobile computing is everyday more present in our daily routines. This can be explained by the great evolution that occurred in this area over the last decade and caused the emergence of a variety of new mobile devices and operating systems, increasingly more sophisticated and powerful. The common tasks of making calls and sending text messages are being obfuscated by others that make use of additional built-in features (e.g. Internet, GPS, accelerometer, video) of the mobile device. These built-in features along with the applications that use them make mobile devices, like smartphones and tablets, very desirable and popular nowadays. The effort made by the mobile industry companies (e.g. Google, Apple and Microsoft) to popularize their respective products also contributed to the ubiquity of mobile devices in our lives and to the variety of mobile platforms and devices available. However, the rapid growing of the mobile market was also responsible for the creation of a certain fragmentation of the mobile platforms, since each company provides its own platform with specific language, SDK and application market. This can be a problem when the goal is to develop mobile applications to several platforms. Traditionally, the differences between each platform would imply rewriting the application for each one of the intended platforms, making the development of mobile applications a complex and time-consuming task. Thus, given these problems, the ideal scenario would be to develop the application once and then run it in as many platforms as desired.

Fortunately, some work has been conducted over the last years to tackle both problems presented previously: the software development complexity and the mobile platform fragmentation. The use of web technologies (e.g. HTML5, CSS3 and JavaScript libraries), cross-platform tools and frameworks, or approaches based on Model-Driven Development (MDD), like the one presented in this dissertation, are examples of solutions focused on solving these problems. MDD approaches seek to mitigate these problems by considering models as the center of the development process. Other deliverables, such as source code or documentation, are generated automatically from those models through model transformations. Apart from making automatic the repetitive tasks of managing the source code and the documentation, MDD further allows developers to express applications in a platform-independent way using domain concepts and therefore generate source code for multiple platforms from a single model specification.

In this dissertation, we have presented XIS-Mobile (language and framework) as an MDD-based approach to develop mobile applications and to mitigate the problems of software development complexity and mobile platform fragmentation. The XIS-Mobile language is a DSL (defined as a UML profile) focused on the mobile applications domain that allows defining mobile applications in a platform-independent way. The XIS-Mobile language has a multi-view organization that adheres to the “separation of concerns” principle and fosters the use of domain concepts, such as widgets, gestures or providers. This fact represents an advantage, because it allows 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. Moreover, XIS-Mobile supports two design approaches: the dummy approach and the smart approach.

Along with the XIS-Mobile language is also proposed a supporting framework based on Sparx Systems Enterprise Architect MDG Technologies 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 modules, this framework suggests developing a mobile application in four steps whenever possible: defining the required views using the Visual Editor, validating them using the Model Validator, generating the User-Interfaces View models with the Model Generator, and finally generating the application's source code through the Code Generator. This way, the developer takes advantage of the MDD benefits, namely increasing his productivity by using a single specification of the system with a PIM, by avoiding the implementation of boilerplate code and reducing errors.

XIS-Mobile has been developed over the last nineteen months following the Action Research Methodology and has been evaluated in order to assess its usefulness and adequacy to the purpose of modelling mobile applications. This evaluation process has been done in an iterative and gradual way through the implementation of case study applications and by conducting a user session. The implementation of case study applications allowed to assess the percentage of code generated of the using the lines of code (LOC) as metric. The results achieved were quite satisfactory with over 75% of the code being generated, on average. The user session focused on the assessment of XIS-Mobile by third parties, i.e., people not directly involved in its development. 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 questionnaire focused on three aspects of XIS-Mobile: the Language, the Framework and the General Approach. The questionnaires collected positive results and showed preliminary evidences that demonstrate XIS-Mobile's usefulness, feasibility and support the thesis behind this dissertation. Additionally, it is important to note that all the goals established in Section 1.3 have been accomplished during the course of this research work.

The remaining of this chapter describes the main contributions of this research work, in Section 7.1, and indicates future research directions, in Section 7.2.

# 7.1. Main Contributions

We believe that the proposed XIS-Mobile approach provides an innovative way to develop mobile applications. The use of domain concepts (e.g. widgets, gestures, providers) and their representation in a graphical way alleviate the development complexity and allow non-developers, like business or functional analysts, to actively participate in the implementation of a mobile application. Also, by generating the boilerplate code, the XIS-Mobile framework avoids unnecessary waste of time by the developers in the realization of such tasks, with potential gains in productivity and even satisfaction, and lower costs and shorter TTM. Additionally, by providing a DSL and a supporting framework, the

XIS-Mobile approach can constitute and interesting tool to introduce the learning of the UML profile mechanism, MDD approach and also the generation of cross-platform mobile applications.

Further contributions of this work can be aligned with the research questions formulated in Section 1.2. Thus, this research work also provides the following contributions aligned with the research questions:

**RQ.1:** Defines a DSL in the form of a UML profile focused on the mobile applications domain.

**RQ.2:** The XIS-Mobile language uses concepts specific of mobile applications, such as gestures, providers and some widgets.

**RQ.3:** Provides a DSL with six views that proved sufficient to specify a mobile application.

**RQ.4:** Applies the set of UI patterns defined in Section 2.3 to generate and model mobile applications.

**RQ.5:** Generates code for multiple platforms through the use of PIMs defined in the XIS-Mobile language which are then transformed to code using Acceleo.

## 7.2. Future work

This section presents the future work and directions that can be followed in a future research related to this work. During the course of this work several ideas have emerged, but either due to time restrictions, for research strategy or due to their complexity, they were considered too ambitions and so were not implemented. It is important to emphasize that none of these research directions undermine the achievement of the goals of this work. Thus, some of these research directions are summarized and explained subsequently:

**Add new patterns to Model-to-Model transformations:** As it was seen in this work, the XIS-Mobile framework uses M2M transformations to apply the smart approach and therefore generate the User-Interfaces View package. Despite that, XIS-Mobile provides a limited set of possibilities, namely in the use cases configuration that control the M2M process. Therefore, would be interesting to enlarge the set of supported M2M generation patterns even to make the use of M2M transformations easier to understand.

**Enhance Stereotypes representation:** A more appealing and closer to the reality representation of the XIS-Mobile stereotypes would enhance the usability and learnability of the XIS-Mobile language. Namely, a better representation of the interaction spaces would benefit the adoption of XIS-Mobile.

**Design and evaluation of more complex and real-world applications:** We intend to further evaluate XIS-Mobile in more complex and real-world scenarios. Mobile Cloud Computing (MCC) and context awareness are two topics very popular nowadays in the mobile computing area. Similarly, the use of CMS-based systems is becoming very popular to support the development of web applications. Thus, it would be interesting to create some extensions of XIS-Mobile to support such scenarios. Testing the use of XIS-Mobile in the academic field and for teaching purposes is another possible research direction

**Improve the code templates:** The code templates for the three different platforms are not in the same state of completeness. The code generation for Android is much more developed than the code generation for iOS and Windows Phone that only have preliminary versions.

**Support incremental model and code generations:** XIS-Mobile supports the generation of models and code, but does not support incremental generations. This means if the user changes a piece of code or models and then needs to perform a new generation, the changes he made will be lost after the generation.

**Integration with other language workbenches:** For now, the XIS-Mobile language and framework can only be used in EA, but allowing its use for other commonly used workbenches or editors would be beneficial to increase its acceptance and expand the set of users.

**Generation for other platforms:** XIS-Mobile supports generation for Android, iOS and Windows Phone, but would also be interesting to have support for a cross-platform tool, namely a web-to-native wrapper, which relies on web technologies (e.g. HTML5, CSS and JavaScript).

**Representation and generation of documentation reports:** The generation of well-defined documentation reports based on specific and possible standard document files templates. The generation of these reports would make documentation using XIS-Mobile diagrams more easily accepted.

# 8. References


Allen, S., Graupera, V., & Lundrigan, L. (2010). *Pro Smartphone Cross-Platform Development: iPhone, Blackberry, Windows Mobile and Android Development and Distribution.* New York: Appress.

Ameller, D. (2009). *Considering Non-Functional Requirements in Model-Driven Engineering (Master Thesis).* Barcelona: Universitat Politècnica de Catalunya.

Balagtas-Fernandez, F., Tafelmayer, M., & Hussmann, H. (2010). Mobia Modeler: easing the creation process of mobile applications for non-technical users. *Proceedings of the 15th International Conference on Intelligent user interfaces (IUI '10)* (pp. 269-272). Hong Kong: ACM.

Baskerville, R. (1999). Investigating Information Systems with action research. *Communications of the Association for Information Systems.* Atlanta: Association for Information Systems.

Belloni, E., & Marcos, C. (2004). MAM-UML: an UML profile for the modeling of mobile-agent applications. *24th International Conference of the Chilean Computer Science Society (SCCC 2004)* (pp. 3-13). Arica: IEEE Computer Society.

Book, M., Beydeda, S., & Gruhn, V. (2005). *Model-driven Software Development.* Springer.

Boudaa, B., Camp, O., Hammoudi, S., & Chikh, M. (2012). Model-Driven Development of Context-Aware Services: Issues, Techniques and Review. *International Conference on IT and e-Services (ICITeS)* (pp. 1-8). Sousse: IEEE Computer Society.

Bourguignon, J.-P. (n.d.). Structuring for managing complexity. *Managing Complexity in Software Engineering*, 217-224.

Brambilla, M., Cabot, J., & Wimmer, M. (2012). *Model-Driven Software Engineering in Practice (1st Edition).* Morgan & Claypool Publishers.

Calvary, G. (2002). *The CAMELEON Reference Framework, Deliverable 1.1.* CAMELEON Project.

Cetinkaya, D., & Verbraeck, A. (2011). Metamodeling and Model Transformations in Modeling and Simulation. *Proceedings of the 2011 Winter Simulation Conference (WSC)* (pp. 3043-3053). Phoenix: IEEE Computer Society.

Charland, A., & LeRoux, B. (2011). Mobile Application Development: Web vs. Native. *Communications of the ACM*.

Costagliola, G., Deufemia, V., Ferrucci, F., & Gravino, C. (2006). Constructing Meta-CASE Workbenches by Exploiting Visual Language Generators. *IEEE Transactions on Software Engineering*, 156-175.

Czarnecki, K., & Helsen, S. (2003). Classification of Model Transformation Approaches. *OOPSLA'03 Workshop on Generative Techniques in the Context of Model-Driven Architecture.* Anaheim.

da Silva, A. R. (2004). *O Programa de Investigação "ProjectIT".* Lisbon.

da Silva, A., Saraiva, J., Ferreira, D., & Silva, R. (2007). Integration of RE and MDE Paradigms: The ProjectIT Approach and Tools. *IET Software Journal*, 294-314.

da Silva, A., Saraiva, J., Silva, R., & Martins, C. (2007). XIS-UML Profile for eXtreme Modeling Interactive Systems. *Proceedings of the 4th International Workshop on Model-based Methodologies for Pervasive and Embedded Software (MOMPES 2007)* (pp. 55-66). Braga: IEEE Computer Society.

de Almeida Ferreira, D., & da Silva, A. R. (2012). RSLingo: An information extraction approach toward formal requirements specifications. *Model-Driven Requirements Engineering Workshop (MoDRE)* (pp. 39-48). Chicago: IEEE Computer Society.

Erdweg, S., van der Storm, T., Völter, M., Boersma, M., Bosman, R., Cook, W. R., & Gerritsen, A. (2013). The State of the Art in Language Workbenches. *Software Language Engineering, 8225*, 197–217.

Fernando, N., Loke, S., & Rahayu, W. (2013). Mobile Cloud Computing: A Survey. *Future Generation Computer Systems*, 84-106.

Fling, B. (2009). *Mobile Design and Development.* O'Reilly Media.

Fondement, F., & Silaghi, R. (2004). Defining Model Driven Engineering Processes. *Proceedings of the 3rd Workshop in Software Model Engineering (WiSME'04).* Lisbon.

Forman, G. H., & Zahorjan, J. (1994). The Challenges of Mobile Computing. *IEEE Computer*, 38-47.

Fowler, M. (2005). *Language Workbenches: The Killer-App for Domain Specific Languages?*

Frankel, D. (2003). *Model Driven Architecture: Applying MDA to Enterprise Computing.* Indianapolis: Wiley Publishing, Inc.

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

Gonzalez-Perez, C., & Henderson-Sellers, B. (2007). Modelling software development methodologies: A conceptual foundation. *Journal of Systems and Software*, 1778–1796.

Greenfield, J., & Short, K. (2004). *Software Factories: Assembling Applications with Patterns, Models, Frameworks and Tools.* Indianapolis: Wiley Publishing Inc.

Hartmann, G., Stead, G., & DeGani, A. (2011). *Cross-platform mobile development.* Cambridge: Mobile Learning Environment.

Hennig, S., Braune, A., & Koycheva, E. (2010). Towards a model driven approach for development of visualization applications in industrial automation. *2010 IEEE Conference on Emerging Technologies and Factory Automation (ETFA)* (pp. 1-6). Bilbao: IEEE Computer Society.

Hoober, S., & Berkman, E. (2011). *Designing Mobile Interfaces.* O'Reilly Media, Inc.

Jouault, F., Allilaire, F., Bézivin, J., & Kurtev, I. (2008). ATL: A model transformation tool. *Science of Computer Programming*, 31-39.

Kieburtz, R., McKinney, L., & Bell, J. (1996). A software engineering experiment in software component generation. *Proceedings of the 18th international conference on Software engineering (ICSE '96)* (pp. 542-552). Washington: IEEE Computer Society.

Kleppe, A., Warmer, J., & Bast, J. (2003). *MDA Explained: The Model Driven Architecture: Practice and Promise.* Boston: Addison-Wesley Longman Publishing.

Kramer, D., Clark, T., & Oussena, S. (2010). MobDSL: A Domain Specific Language for multiple mobile platform deployment. *2010 IEEE International Conference on Networked Embedded Systems for Enterprise Applications (NESEA)* (pp. 1-7). Suzhou: IEEE Computer Society.

Kuhn, T., Gotzhein, R., & Webel, C. (2006). Model-Driven development with SDL - process, tools, and experiences. *Proceedings of the 9th international conference on Model Driven Engineering Languages and Systems (MoDELS'06)* (pp. 83-97). Springer-Verlag.

Langlands, M. (2010). *Inside the Oval: Use-Case Content Patterns (Technical Report).* Planet Project.

Meeker, M., Devitt, S., & Wu, L. (2010). *Internet Trends.* Morgan Stanley Research.

Mendoza, A. (2013). *Mobile User Experience: Patterns to Make Sense of it All .* Morgan Kaufmann.

Mitchell, R. (1990). *Managing Complexity in Software Engineering.* Philadelphia.

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

Nielsen, J., & Landauer, T. (1993). A Mathematical Model of the Finding of Usability Problems. *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems (CHI '93)* (pp. 206-213). Amsterdam: ACM.

Nilsson, E. G. (2009). Design Patterns for User interface for Mobile Applications. *Advances in Engineering Software*, 1318-1328.

OMG. (2003). *Object Management Group - MDA Guide, Version 1.0.1.* OMG.

OMG. (2008). *Object Management Group - MOF Model to Text Transformation Language (MOFM2T), Version 1.0.* OMG.

OMG. (2011). *Object Management Group - Meta Object Facility (MOF) 2.0 Query/View/Transformation (QVT) Specification, Version 1.1.* OMG.

OMG. (2011). *Object Management Group - Unified Modeling Language (UML): Superstructure Specification, Version 2.4.1.* OMG.

OMG. (2014). *Object Management Group - Meta Object Facility (MOF) Core, Version 2.4.2.* OMG.

OMG. (2014). *Object Management Group - Object Constraint Language (OCL) Specification, Version 2.4.* OMG.

OMG. (2014). *Object Management Group - XML Metadata Interchange (XMI), Version 2.4.2.* OMG.

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

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

Ranabahu, A., Maximilien, M., Sheth, A. P., & Thirunarayan, K. (2011). A Domain Specific Language for Enterprise Grade Cloud-Mobile Hybrid Applications. *Proceedings of the compilation of the co-located workshops on DSM'11, TMC'11, AGERE!'11, AOOPES'11, NEAT'11, & VMIL'11 (SPLASH '11 Workshops)* (pp. 77-84). Portland: ACM.

Ribeiro, A., & da Silva, A. (2013). Survey on Cross-Platforms and Languages for Mobile Apps. *Proceedings of the 2012 Eighth International Conference on the Quality of Information and Communications Technology (QUATIC '12)* (pp. 255-260). Lisbon: IEEE Computer Society.

Ribeiro, A., & da Silva, A. (2014). Evaluation of XIS-Mobile, a DSL for Mobile Applications. *Journal of Software Engineering and Applications*.

Ribeiro, A., & da Silva, A. (2014). XIS-Mobile: A DSL for Mobile Applications. *Proceedings of 29th Symposium on Applied Computing (SAC'14)* (pp. 1316-1323). Gyeongju: ACM.

Sandvoll, M. (2014). *MobileApps4Tourism (Research Project).* Lisbon: Instituto Superior Técnico.

Saraiva, J. d., & da Silva, A. R. (2008). Evaluation of MDE Tools from a Metamodeling Perspective. *Journal of Database Management*, 21-46.

Saraiva, J. d., & da Silva, A. R. (2010). Web-Application Modeling with the CMS-ML Language. *Actas do II Simpósio de Informática (INForum 2010)*, (pp. 461-472). Lisbon.

Saraiva, J., & da Silva, A. (2009). CMS-Based Web-Application Development Using Model-Driven Languages. *Proceedings of the 4thInternational Conference on Software Engineering Advances* (pp. 21-26). Porto: IEEE Computer Society.

Satyanarayanan, M. (1996). Fundamental Challenges in Mobile Computing. *Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing (PODC '96)* (pp. 1-7). Philadelphia: ACM.

Savić, D., da Silva, A. R., Vlajić, S., Lazarević, S., Antović, I., Stanojević, V., & Milić, M. (2014). Preliminary Experience Using JetBrains MPS to Implement a Requirements Specification Language. *Proceedings of the 9th International Conference on the Quality of Information and Communications Technology (QUATIC).* Guimarães: IEEE Computer Society.

Schmidt, D. (2006). Guest Editor's Introduction: Model-Driven Engineering. *IEEE Computer*, 25-31.

Sendall, S., & Kozaczynski, W. (2003). Model transformation: the heart and soul of model-driven software development. *Software IEEE*, 42-45.

Singh, I., & Palmieri, M. (2011). *Comparison of cross-platform mobile development tools.* Västerås: IDT: Mälardalen University.

Stahl, T., & Voelter, M. (2005). *Model-Driven Software Development: Technology, Engineering, Management.* Hoboken: John Wiley & Sons.

Steinberg, D., Budinsky, F., Paternostro, M., & Merks, E. (2008). *EMF: Eclipse Modeling Framework.* Addison-Wesley Professional.

Trindade, F., & Pimenta, M. (2007). *Prototyping Multi-platform Software Using the UsiXML4ALL Tool (Technical Report).*

van Deursen, A., Klint, P., & Visser, J. (2000). Domain-Specific Languages: An Annotated Bibliography. *SIGPLAN*, 26-36.

Vanderdonckt, J. (2005). A MDA-compliant environment for developing user interfaces of information systems. *Proceedings of the 17th international conference on Advanced Information Systems Engineering (CAiSE'05)* (pp. 16-31). Porto: Springer-Verlag.

Vanderdonckt, J., & Simarro, F. (2010). Generative Pattern-Based Design of User Interfaces. *Proceedings of the 1st International Workshop on Pattern-Driven Engineering of Interactive Computing Systems (PEICS '10)* (pp. 12-19). Berlin: ACM.

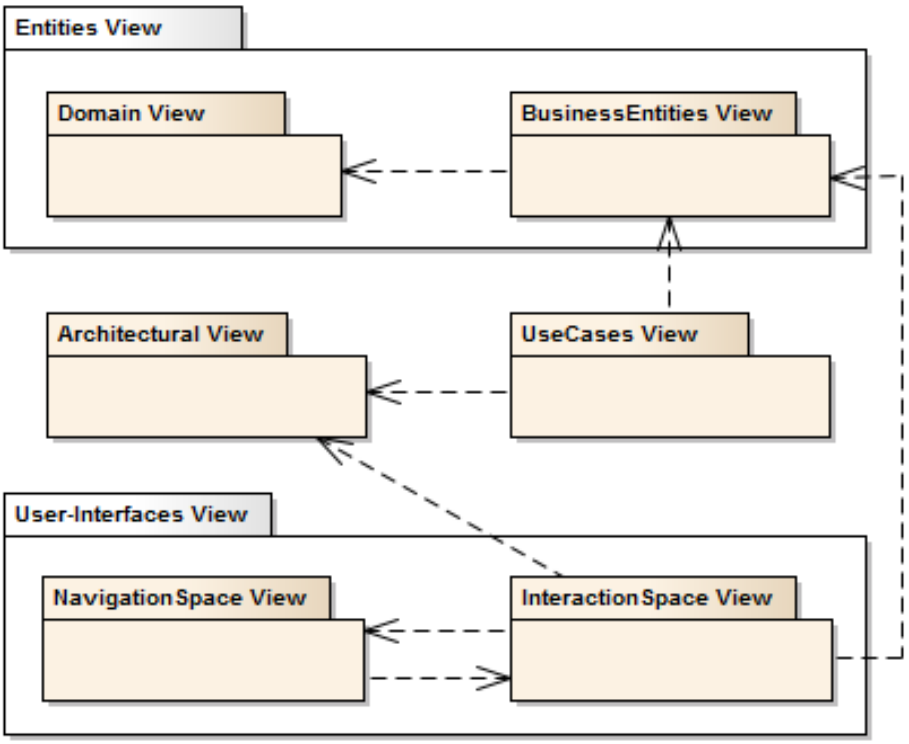
Weiser, M. (1993). Some Computer Science Issues in Ubiquitous Computing. *Communications of the ACM*, 75-84.

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

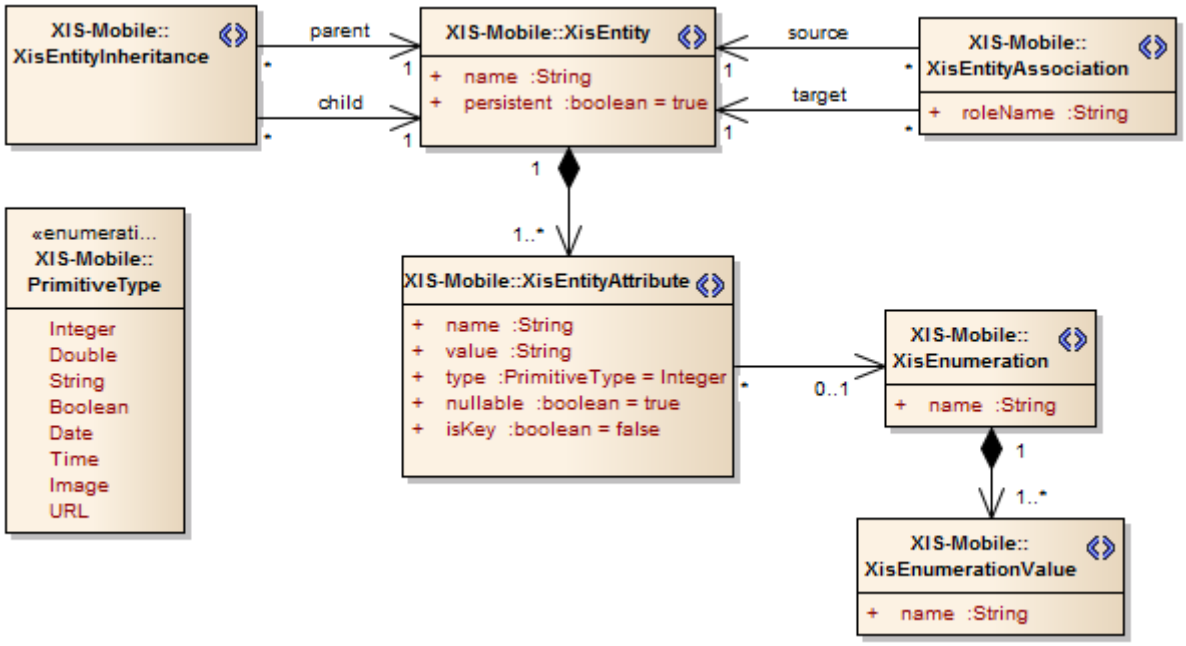
Wolber, D. (2011). App inventor and Real-World Motivation. *Proceedings of the 42nd ACM Technical Symposium on Computer science Education (SIGCSE '11)* (pp. 601-606). Dallas: ACM.

# A. XIS-Mobile Profile Specification

The XIS-Mobile UML Profile comprises six views: Domain View, BusinessEntities View, Architectural View, UseCases View, NavigationSpace View and InteractionSpace View. Each one of them is detailed in the next sections.



# A.1. Domain View




| Stereotype | Metaclass |
|---|---|
| XisEntity | Class |
| XisEntityAttribute | Property |
| XisEnumeration | Enumeration |
| XisEnumerationValue | EnumerationLiteral |
| XisEntityAssociation | Association |
| XisEntityInheritance | Generalization |

| XisEntity | | |
|---|---|---|
| Represents a domain entity. | | |
| **Tagged Value** | **Type** | **Description** |
| name | String | Specifies the name of the domain entity. |
| persistent | Boolean | Specifies if the entity should be persisted or not. |

| XisEntityAttribute | | |
|---|---|---|
| Represents an attribute or property of a domain entity (XisEntity). | | |
| **Tagged Value** | **Type** | **Description** |
| name | String | Specifies the name of the attribute. |
| value | String | Assigns a value to the entity attribute. |
| type | PrimitiveType | Specifies the type of the domain entity. |
| nullable | Boolean | Specifies if the attribute can be null. |
| isKey | Boolean | Specifies if the attribute is a key. |

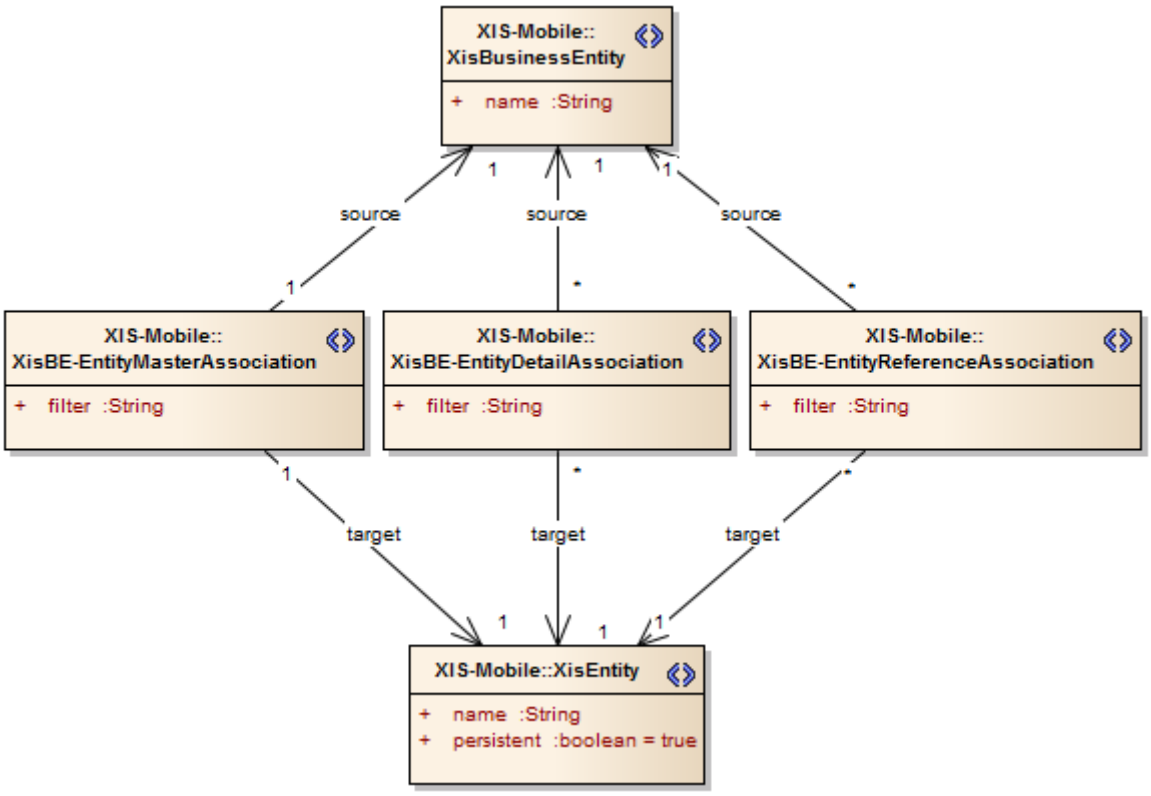
| XisEnumeration | | |
|---|---|---|
| Represents an enumeration that can be used as an attribute of a domain entity. | | |
| **Tagged Value** | **Type** | **Description** |
| name | String | Specifies the identifier of the enumeration. |

| XisEnumerationValue | | |
|---|---|---|
| Represents an enumeration literal, i.e., a possible value of an enumeration (XisEnumeration). | | |
| **Tagged Value** | **Type** | **Description** |
| name | String | Specifies the identifier of the enumeration value. |

| XisEntityAssociation | | |
|---|---|---|
| Associates two domain entities (XisEntity). | | |
| **Tagged Value** | **Type** | **Description** |
| roleName | String | Specifies the identifier of the association. |

| XisEntityInheritance |
|---|
| Represents a generalization between two domain entities (a parent and its child). |

# A.2. BusinessEntities View




| Stereotype | Metaclass |
|---|---|
| XisBusinessEntity | Class |
| XisBE-EntityMasterAssociation | Association |
| XisBE-EntityDetailAssociation | Association |
| XisBE-EntityReferenceAssociation | Association |

| XisBusinessEntity | | |
|---|---|---|
| Represents a business entity, which consists in an aggregation of domain entities: a main domain entity and a set of entities (detail and reference domain entities) associated to it that contribute to its definition. | | |
| **Tagged Value** | **Type** | **Description** |
| name | String | Specifies the identifier of the business entity. |

| XisBE-EntityMasterAssociation | | |
|---|---|---|
| Defines the main domain entity of the business entity (XisBusinessEnity). | | |
| **Tagged Value** | **Type** | **Description** |
| filter | String | Specifies the domain entity's attributes aggregated by the business entity. |

| XisBE-EntityDetailAssociation | | |
|---|---|---|
| Defines the set of detail domain entities[28] related to the master domain entity. | | |
| **Tagged Value** | **Type** | **Description** |
| filter | String | Specifies the domain entity's attributes aggregated by the business entity. |

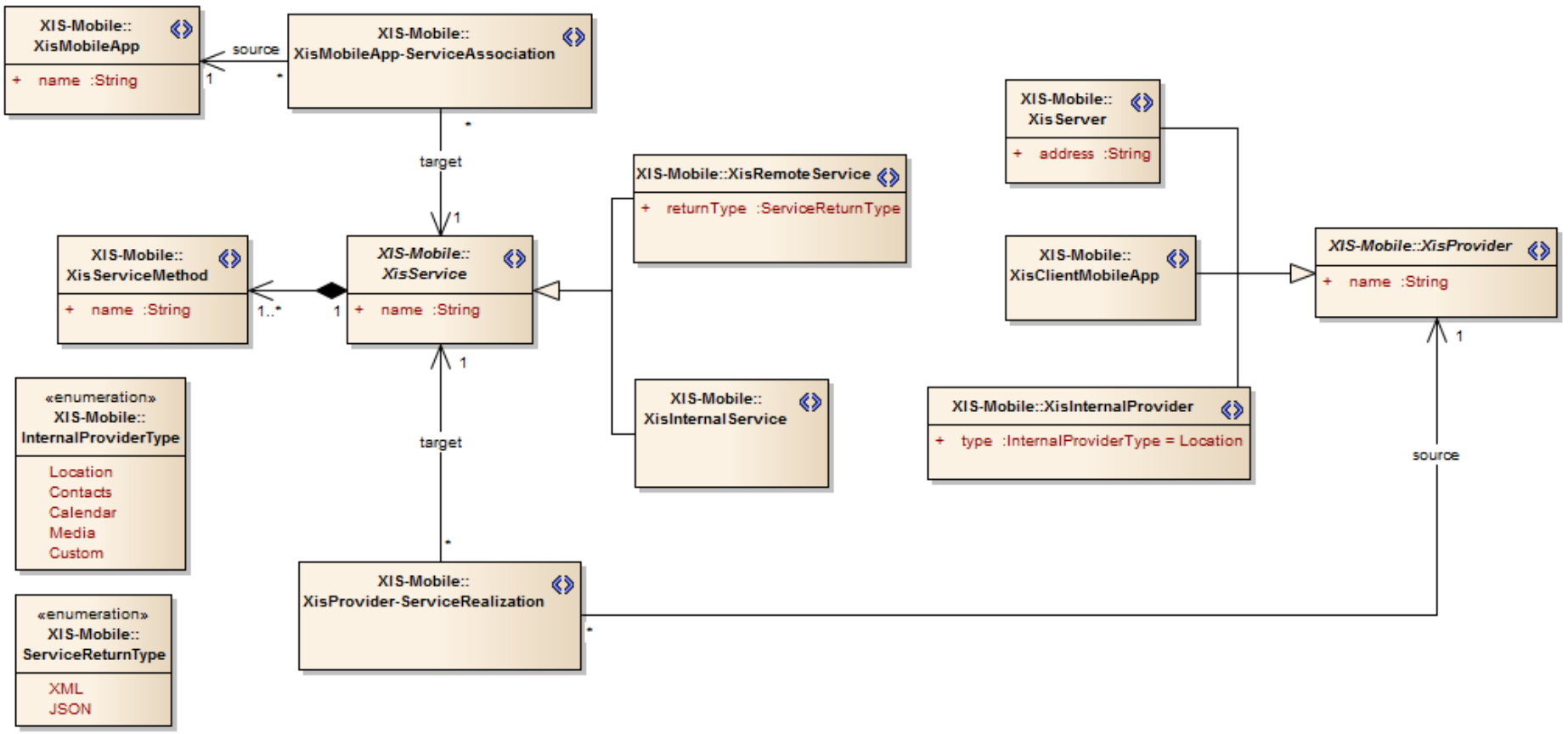
| XisBE-EntityReferenceAssociation | | |
|---|---|---|
| Defines the set of reference domain entities[29] related to the master domain entity. | | |
| **Tagged Value** | **Type** | **Description** |
| filter | String | Specifies the domain entity's attributes aggregated by the business entity. |

The filter tagged value can have the following values:

- If empty, all domain entity's attributes will be aggregate by the business entity;
- If one entity attribute, should be in the format <DomainEntityName>.<DomainEntityAttributeName>;
- If more than one entity attribute, they should be in the previous format and separated by a semicolon (;).

[28] Detail domain entities are entities associated to the master domain entity through an aggregation or composition in the Domain View.
[29] Reference domain entities are entities associated to the master entity through an association in the Domain View.

## A.3. Architectural View



| Stereotype | Metaclass |
|---|---|
| XisMobileApp | Class |
| XisMobileApp-ServiceAssociation | Association |
| XisService | Interface |
| XisServiceMethod | Operation |
| XisRemoteService | XisService |
| XisInternalService | XisService |
| XisProvider-ServiceRealization | Realization |
| XisProvider | Class |
| XisServer | XisProvider |
| XisClientMobileApp | XisProvider |
| XisInternalProvider | Class |

| **XisMobileApp** | | |
|---|---|---|
| Represents the mobile application detailed in the other views. | | |
| **Tagged Value** | **Type** | **Description** |
| name | String | Specifies the name of the mobile application. |

| **XisMobileApp-ServiceAssociation** |
|---|
| Associates the mobile application (XisMobileApp) to a service (XisService). |

| **XisService** | | |
|---|---|---|
| Represents a service composed by operations (XisServiceMethod) that can be invoked by the mobile application (XisMobileApp). | | |
| **Tagged Value** | **Type** | **Description** |
| name | String | Specifies the identifier of the service. |

| **XisServiceMethod** | | |
|---|---|---|
| Represents an operation provided by a service (XisService). | | |
| **Tagged Value** | **Type** | **Description** |
| name | String | Specifies the identifier of the operation. |

| XisRemoteService | | |
|---|---|---|
| Represents a service provided by an external entity. | | |
| **Tagged Value** | **Type** | **Description** |
| returnType | ServiceReturnType | Specifies the type of the response: XML or JSON. |

| XisInternalService |
|---|
| Represents a service provided by an entity within the mobile application's operating system. |

| XisProvider-ServiceRealization |
|---|
| Represents a realization relationship of a service (XisService) by a provider (XisProvider). |

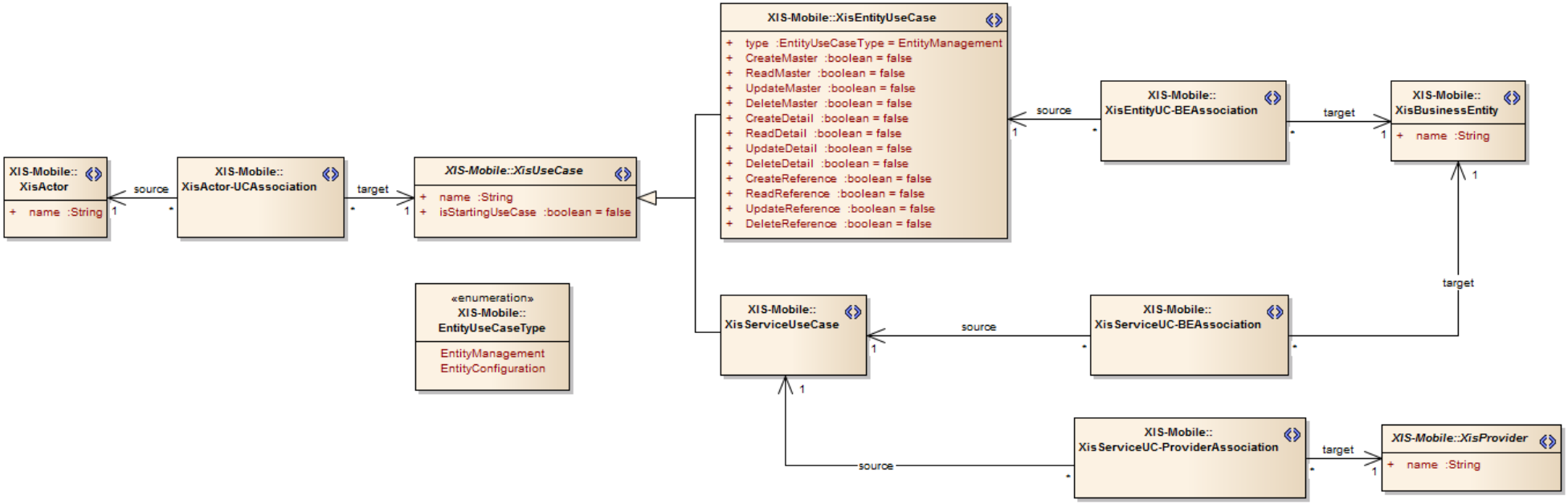
| XisProvider | | |
|---|---|---|
| Represents a content provider. | | |
| **Tagged Value** | **Type** | **Description** |
| Name | String | Specifies the identifier of the provider. |

| XisServer | | |
|---|---|---|
| Represents a remote server that interacts with the main mobile application (XisMobileApp). | | |
| **Tagged Value** | **Type** | **Description** |
| Address | String | Specifies the IP address of the server. |

| XisClientMobileApp |
|---|
| Represents a mobile application that interacts with the main mobile application (XisMobileApp). |

| XisInternalProvider | | |
|---|---|---|
| Represents an internal provider, which allows interacting with the mobile operating system's native features. | | |
| **Tagged Value** | **Type** | **Description** |
| Type | InternalProviderType | Specifies the type of the internal provider: Location, Contacts, Calendar, Media or Custom. |

# A.4. UseCases View



| Stereotype | Metaclass |
|---|---|
| XisActor | Actor |
| XisActor-UCAssociation | Association |
| XisUseCase | UseCase |
| XisEntityUseCase | XisUseCase |
| XisServiceUseCase | XisUseCase |
| XisEntityUC-BEAssociation | Association |
| XisServiceUC-BEAssociation | Association |
| XisServiceUC-ProviderAssociation | Association |

| XisActor | | |
|---|---|---|
| Represents an Actor. | | |
| **Tagged Value** | **Type** | **Description** |
| name | String | Specifies the identifier of the actor. |

| XisActor-UCAssociation |
|---|
| Associates an actor (XisActor) to a use case (XisUseCase). |

| XisUseCase | | |
|---|---|---|
| Represents a use case. | | |
| **Tagged Value** | **Type** | **Description** |
| name | String | Specifies the identifier of the use case. |
| isStartingUseCase | Boolean | Specifies if the use case is the first that should be considered during the Model-to-Model transformation stage. |

| XisEntityUseCase | | |
|---|---|---|
| Represents a use case that performs a set of operations over a business entity. | | |
| **Tagged Value** | **Type** | **Description** |
| type | EntityUseCaseType | Specifies the type of pattern associated to the use case: EntityManagement or EntityConfiguration |
| CreateMaster | Boolean | Specifies if the use case will generate models which allow the creation of the associated |

| | | |
|---|---|---|
| | | master entity. |
| ReadMaster | Boolean | Specifies if the use case will generate models which allow the visualization of the associated master entity. |
| UpdateMaster | Boolean | Specifies if the use case will generate models which allow the edition of the associated master entity. |
| DeleteMaster | Boolean | Specifies if the use case will generate models which allow the deletion of the associated master entity. |
| CreateDetail | Boolean | Specifies if the use case will generate models which allow the creation of the associated detail entities. |
| ReadDetail | Boolean | Specifies if the use case will generate models which allow the visualization of the associated detail entities. |
| UpdateDetail | Boolean | Specifies if the use case will generate models which allow the edition of the associated detail entities. |
| DeleteDetail | Boolean | Specifies if the use case will generate models which allow the deletion of the associated detail entities. |
| CreateReference | Boolean | Specifies if the use case will generate models which allow the creation of the associated reference entities. |
| ReadReference | Boolean | Specifies if the use case will generate models which allow the visualization of the associated reference entities. |
| UpdateReference | Boolean | Specifies if the use case will generate models which allow the edition of the associated reference entities. |
| DeleteReference | Boolean | Specifies if the use case will generate models which allow the deletion of the associated reference entities. |

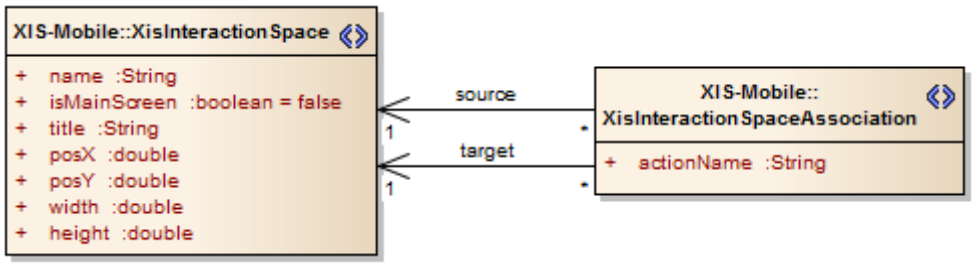
| **XisServiceUseCase** |
|---|
| Represents a use case that uses a set of services provided by a XisProvider and can be connected to a business entity (XisBusinessEntity). |

| **XisEntityUC-BEAssociation** |
| --- |
| Associates a XisEntityUseCase to a business entity (XisBusinessEntity). |

| **XisServiceUC-BEAssociation** |
| --- |
| Associates a XisServiceUseCase to a business entity (XisBusinessEntity). |

| **XisServiceUC-ProviderAssociation** |
| --- |
| Associates a XisServiceUseCase to a provider (XisProvider): XisClientMobileApp, XisServer or XisInternalProvider. |

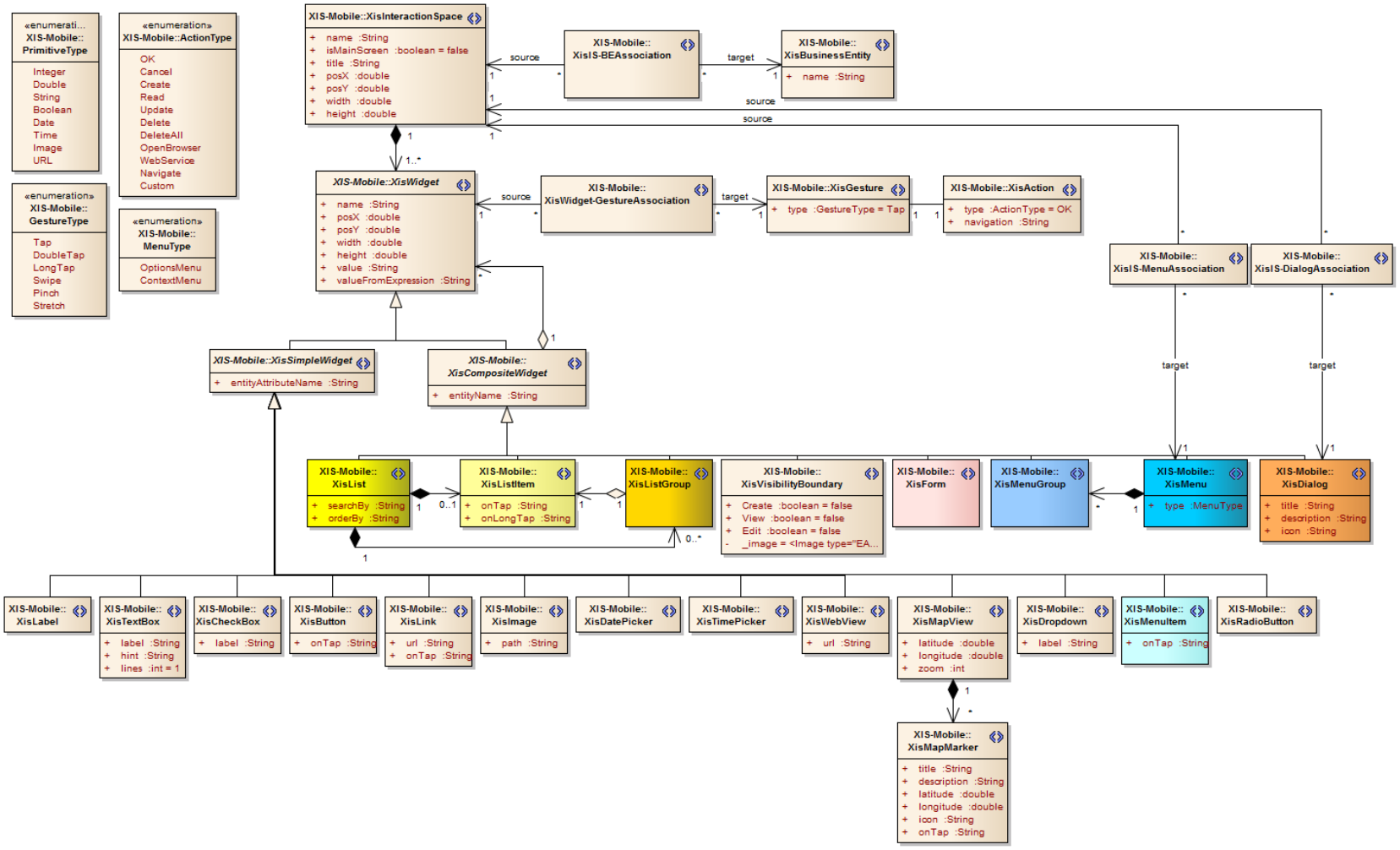
# A.5. NavigationSpace View




| Stereotype | Metaclass |
|---|---|
| XisInteractionSpaceAssociation | Association |

| XisInteractionSpaceAssociation | | |
|---|---|---|
| Defines the transitions between two interaction spaces (XisInteractionSpace). | | |
| **Attribute** | **Type** | **Description** |
| actionName | String | Specifies the name of the action that triggered the navigation. |

# A.6. InteractionSpace View



| Stereotype | Metaclass |
|---|---|
| XisInteractionSpace | Class |
| XisIS-BEAssociation | Association |
| XisWidget | Class |
| XisWidget-GestureAssociation | Association |
| XisGesture | Class |
| XisAction | Operation |
| XisSimpleWidget | XisWidget |
| XisCompositeWidget | XisWidget |
| XisLabel | XisSimpleWidget |
| XisTextBox | XisSimpleWidget |
| XisCheckBox | XisSimpleWidget |
| XisButton | XisSimpleWidget |
| XisLink | XisSimpleWidget |
| XisImage | XisSimpleWidget |
| XisDatePicker | XisSimpleWidget |
| XisTimePicker | XisSimpleWidget |
| XisWebView | XisSimpleWidget |
| XisMapView | XisSimpleWidget |
| XisMapMarker | XisSimpleWidget |
| XisDropdown | XisSimpleWidget |
| XisMenuItem | XisSimpleWidget |
| XisRadioButton | XisSimpleWidget |
| XisList | XisCompositeWidget |
| XisListItem | XisCompositeWidget |
| XisListGroup | XisCompositeWidget |
| XisVisibilityBoundary | XisCompositeWidget |
| XisForm | XisCompositeWidget |
| XisMenuGroup | XisCompositeWidget |
| XisMenu | XisCompositeWidget |
| XisDialog | XisCompositeWidget |
| XisIS-MenuAssociation | Association |
| XisIS-DialogAssociation | Association |

| XisInteractionSpace | | |
|---|---|---|
| Represents an interaction space, i.e., an application screen. | | |
| **Tagged Value** | **Type** | **Description** |
| name | String | Specifies the identifier of the interaction space. |
| isMainScreen | Boolean | Specifies if the screen is the first one presented when the mobile application starts. |

| XisIS-BEAssociation |
|---|
| Associates an interaction space to a business entity. |

| XisWidget | | |
|---|---|---|
| Represents a UI widget or control of an interaction space (XisInteractionSpace). | | |
| **Tagged Value** | **Type** | **Description** |
| name | String | Specifies the identifier of the widget. |
| posX | Double | Horizontal position of the widget. |
| posY | Double | Vertical position of the widget. |
| width | Double | Specifies the width of the widget. |
| height | Double | Specifies the height of the widget. |
| value | String | Assigns a value to the widget. |

| XisWidget-GestureAssociation |
|---|
| Represents a textbox. |

| XisGesture | | |
|---|---|---|
| Represents a gesture attached to a UI widget (XisWidget). | | |
| **Tagged Value** | **Type** | **Description** |
| name | String | Specifies the identifier of the gesture. |
| type | GestureType | Specifies the type of the gesture: Tap, DoubleTap, LongTap, Swipe, Pinch or Stretch. |

| XisAction | | |
|---|---|---|
| Represents an action triggered by a gesture. | | |
| **Tagged Value** | **Type** | **Description** |
| name | String | Specifies the name of the action. |
| type | ActionType | Specifies the type of the action: OK, Cancel, Create, Read, Update, Delete, DeleteAll, OpenBrowser, WebService, Navigate or Custom. |
| navigation | String | Determines if the action causes navigation to another interaction space. If so, its value is the name of the target interaction space. |

| XisSimpleWidget | | |
|---|---|---|
| Represents a textbox. | | |
| **Tagged Value** | **Type** | **Description** |
| label | String | Specifies the label of the textbox. |

| XisCompositeWidget | | |
|---|---|---|
| Represents a textbox. | | |
| **Tagged Value** | **Type** | **Description** |
| label | String | Specifies the label of the textbox. |

| XisLabel |
|---|
| Represents a label. |

| XisTextBox | | |
|---|---|---|
| Represents a textbox. | | |
| **Tagged Value** | **Type** | **Description** |
| label | String | Specifies the label of the textbox. |
| hint | String | Specifies the hint of the textbox. |
| lines | Integer | Specifies the number of lines of the textbox. |

| XisCheckBox | | |
|---|---|---|
| Represents a checkbox. | | |
| **Tagged Value** | **Type** | **Description** |
| label | String | Specifies the label of the checkbox. |

| XisButton | | |
|---|---|---|
| Represents a button. | | |
| **Tagged Value** | **Type** | **Description** |
| onTap | String | Specifies if the widget has a tap gesture attached. If so, its value is the name of the operation it triggers. |

| XisLink | | |
|---|---|---|
| Represents a link. | | |
| **Tagged Value** | **Type** | **Description** |
| url | String | Specifies the link's destination. |
| onTap | String | Specifies if the widget has a tap gesture attached. If so, its value is the name of the operation it triggers. |

| XisImage | | |
|---|---|---|
| Represents an image. | | |
| **Tagged Value** | **Type** | **Description** |
| path | String | Specifies the image's location. |

| XisDatePicker |
|---|
| Represents a date picker. |

| XisTimePicker |
|---|
| Represents a time picker. |

| XisWebView | | |
|---|---|---|
| Represents a web view. | | |
| **Tagged Value** | **Type** | **Description** |
| url | String | Specifies the address of the web site accessed. |

| XisMapView | | |
|---|---|---|
| Represents a Google Maps view. | | |
| **Tagged Value** | **Type** | **Description** |
| latitude | Double | Specifies the latitude of the center of the map. |
| longitude | Double | Specifies the longitude of the center of the map. |
| zoom | Integer | Specifies the zoom level of the map. |

| XisMapMarker | | |
|---|---|---|
| Represents a marker of a Google Maps view (XisMapView). | | |
| **Tagged Value** | **Type** | **Description** |
| title | String | Specifies the title of the marker. |
| description | String | Specifies the description of the marker. |
| latitude | Double | Specifies the latitude of the marker. |
| longitude | Double | Specifies the longitude of the marker. |
| icon | String | Specifies the path of the marker's icon. |
| onTap | String | Specifies if the marker has a tap gesture attached. If so, its value is the name of the operation it triggers. |

| XisDropdown |
|---|
| Represents a dropdown. |

| XisMenuItem | | |
|---|---|---|
| Represents a menu item. | | |
| **Tagged Value** | **Type** | **Description** |
| onTap | String | Specifies if the menu item has a tap gesture attached. If so, its value is the name of the operation it triggers. |

| XisRadioButton |
|---|
| Represents a radio button. |

| XisList | | |
|---|---|---|
| Represents a list. | | |
| **Tagged Value** | **Type** | **Description** |
| searchBy | String | Specifies the domain entity's attributes that can be used to filter the items of the list. |
| orderBy | String | Specifies the domain entity's attributes that can be used to order the items of the list. |

| XisListItem | | |
|---|---|---|
| Represents a list item. | | |
| **Tagged Value** | **Type** | **Description** |
| onTap | String | Specifies if the list item has a tap gesture attached. If so, its value is the name of the operation it triggers. |
| onLongTap | String | Specifies if the list item has a long tap gesture attached. If so, its value is the name of the operation it triggers. |

| XisListGroup |
|---|
| Represents a group of list items. |

| XisVisibilityBoundary | | |
|---|---|---|
| Represents a boundary that defines when the UI widgets it includes are visible. | | |
| **Tagged Value** | **Type** | **Description** |
| Create | Boolean | Specifies if the widgets included are visible in Create mode. |
| View | String | Specifies if the widgets included are visible in View mode. |
| Edit | String | Specifies if the widgets included are visible in Edit mode. |

| XisForm |
|---|
| Represents a form that groups a set of widgets. |

| XisMenuGroup |
|---|
| Represents a group of menu items. |

| XisMenu | | |
|---|---|---|
| Represents a menu. | | |
| **Tagged Value** | **Type** | **Description** |
| type | MenuType | Specifies the type of the menu: OptionsMenu or ContextMenu. |

| XisDialog | | |
|---|---|---|
| Represents an information dialog or popup. | | |
| **Tagged Value** | **Type** | **Description** |
| title | String | Specifies the title of the dialog. |
| description | String | Specifies the description of the dialog. |
| icon | String | Specifies the path of the dialogs's icon. |

| XisMenuAssociation |
|---|
| Associates an interaction space to a menu. |

| XisDialogAssociation |
|---|
| Associates an interaction space to a dialog. |

# B. User Session Guide

## Simple Test Session – User Guide, v1.0, 2014/July

The main goal of this user test session is to perform an evaluation of XIS-Mobile (language and framework) by users not familiarized with it, in order to detect eventual bugs and possible user limitations. The evaluation will be performed through the development of a very simple case study application: “Flight Reservation App”. The gathered results will be used to assess the XIS-Mobile language usability as well as its future enhancements. The case study application is described below.

**Case Study – Flight Reservation App**

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

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

**Conditions:**

This user test session will be conducted under the following conditions:

- The tests are conducted in the laboratory (controlled environment);
- The tasks must be performed without previous use and learning (for the first time);
- The user must have a computer running Windows and previously installed Sparx Enterprise Architect (version 7.5, 10 or above)[30];
- Direct Observation, i.e., while users perform the assigned task, their behavior and performance can be logged;
- Users can think out loud and share ideas if they want;
- The evaluator does not interact with the users until the tests are finished (except in case of blocking errors);
- The session will last 50 minutes (at most).

[30]For IST users, there are free and full licensed versions available in https://delta.ist.utl.pt/software/ea.php. For other users, there is a 30 day trial version that can be downloaded from http://www.sparxsystems.com/products/ea/trial.html.

**Instructions:**

1. Download the XIS-Mobile EA Plugin installer from: https://github.com/xis-mobile/XIS-Mobile/raw/master/XIS-MobileEAPlugin_Setup.msi
2. Install the XIS-Mobile EA Plugin. This plugin extends EA with the XIS-Mobile MDG (Model Driven Generation) Technology which contains the information about the XIS-Mobile profile, its custom diagrams, toolboxes and project template. The plugin also provides EA with the XIS-Mobile menu which allows triggering the model validation of a XIS-Mobile project, model generation and code generation.
3. Open Enterprise Architect and create a new Project.
4. When the Model Wizard window opens up select the "XIS-Mobile Framework" technology, then check the "XIS-Mobile Framework model" option and confirm using the "OK" button. After that, there should be a view containing a Package Diagram with 6 packages (one per each XIS-Mobile view).
5. Delete the diagrams of the UI views (InteractionSpace and NavigationSpace views). To do so right-click on the Model Viewer or Project Browser, go to the "Extension" menu option, "XIS-Mobile Plugin" option and select the "Delete Generated Models" option.
6. Model the Domain view:
   - 6.1 Create a XisEnumeration named "FlightClass" and add to it three XisEnumerationValues: "Economy", "Business" and "First".
   - 6.2 Create a XisEntity named "Passenger" and add to it the following XisEntityAttributes:
     - 6.2.1 "passportID" with type "String"
     - 6.2.2 "name" with type "String"
     - 6.2.3 "age" with type "int"
     - 6.2.4 "country" with type "String"
   - 6.3 Create a XisEntity named "Reservation" and add to it the following XisEntityAttributes:
     - 6.3.1 "oneWay" with type "boolean"
     - 6.3.2 "departureDate" with type "Date"
     - 6.3.3 "returnDate" with type "Date"
     - 6.3.4 "class" with type "FlightClass"
   - 6.4 Create a XisEntity named "Airport" and add to it a XisEntityAttribute called "name" with type "String".
   - 6.5 Connect the "Reservation" entity to the "Passenger" entity through a "XisEntityAssociation":
     - 6.5.1 Double click on the association and select the "Source Role" left link. Then set the "Multiplicity" to 1
     - 6.5.2 Select the "Target Role" left link and set the "Multiplicity" to 1..*
   - 6.6 Connect the "Reservation" entity to the "Airport" entity through a "XisEntityAssociation" for the "destinationAirport" concept:
     - 6.6.1 Double click on the association and select the "Source Role" left link. Then set the "Multiplicity" to *
     - 6.6.2 Select the "Target Role" left link and set the "Multiplicity" to 1
7. Model the BusinessEntities view:
   - 7.1 Create the following XisBusinessEntities: "ReservationBE", "AirportBE", and "PassengerBE".
   - 7.2 Copy the "Reservation", "Airport" and "Passenger" entities from the Domain view and paste them as links.
   - 7.3 Connect the "ReservationBE": to the "Reservation" through a "XisBE-EntityMasterAssociation"; to the "Passenger" through a "XisBE-EntityReferenceAssociation"; and to the "Airport" through a "XisBE-EntityReferenceAssociation".
   - 7.4 Connect the "PassengerBE" to the "Passenger" through a "XisBE-EntityMasterAssociation".
   - 7.5 Connect the "AirportBE" to the "Airport" through a "XisBE-EntityMasterAssociation".
8. Model the Architectural view:
   - 8.1 Create a "XisMobileApp" named "FlightReservationApp".

8.2 Create a “XisRemoteService” named “FlightReservationService” and add to it a “XisServiceMethod” named “syncAirports”.
8.3 Create a “XisServer” named “FlightServer”.
8.4 Connect the “FlightReservationApp” to the “FlightReservationService” through a “XisMobileApp-ServiceAssociation”.
8.5 Connect the “FlightServer” to the “FlightReservationService” through a “XisProvider-ServiceRealization”.

9. Model the UseCases view:
   9.1 Create a “XisActor” named “User”.
   9.2 Create a “XisEntityUseCase” named “Manage Reservations”; Double-click on it and select the “XIS-Mobile” left link to see its tagged values:
      9.2.1 Make sure the “type” tagged value is set to “Entity Management”
      9.2.2 Set all boolean tagged values to true except: “CreateDetail”, “DeleteDetail”, “ReadDetail” and “UpdateDetail”
   9.3 Create a “XisEntityUseCase” named “Manage Passengers”; Double-click on it and select the “XIS-Mobile” left link to see its tagged values:
      9.3.1 Make sure the “type” tagged value is set to “Entity Management”
      9.3.2 Set the following boolean tagged values to true: “CreateMaster”, “DeleteMaster”, “ReadMaster” and “UpdateMaster”
   9.4 Create a “XisServiceUseCase” named “Sync Airports”.
   9.5 Connect the “User” actor to the use cases through “XisActor-UCAssociation” associations.
   9.6 Copy the “ReservationBE”, “AirportBE” and “PassengerBE” entities from the BusinessEntities view and paste them as links.
   9.7 Associate the “Manage Reservations” use case to the “ReservationBE” entity through a “XisEntityUC-BEAssociation”.
   9.8 Associate the “Manage Passengers” use case to the “PassengerBE” entity through a “XisEntityUC-BEAssociation”.
   9.9 Associate the “Sync Airports” use case to the “AirportBE” entity through a “XisServiceUC-BEAssociation”.
   9.10 Copy the “FlightServer” class from the Architectural view and paste it as a link.
   9.11 Associate the “Sync Airports” use case to “FlightServer” through a “XisServiceUC-ProviderAssociation”.
10. Validate the model. To do so right-click on the Model Viewer or Project Browser, go to the “Extension” menu option, “XIS-Mobile Plugin” option and select the “Validate Model” option.
11. (If the model is valid) Apply the Model-to-Model generation (according to the “Smart Approach”) to automatically produce the diagrams associated to the NavigationSpace and InteractionSpace views. To do so right-click on the Model Viewer or Project Browser, go to the “Extension” menu option, “XIS-Mobile Plugin” option and select the “Generate Models” option.
12. Then a model generation window will appear. Select the option “Springboard” as “Home generation Pattern” and click on the “Generate!” button.
13. Check if the InteractionSpace and NavigationSpace packages content has changed.
14. (Optional if you have time) Change slightly one of the interaction space model
15. Validate the model repeating step 10.
16. (If the model is valid) Apply the model-to-code generation for the Android platform to automatically generate the Java source code for the selected target platform. Otherwise, check the errors triggered, fix them and try again.
17. To do so right-click on the Model Viewer or Project Browser, go to the “Extension” menu option, “XIS-Mobile Plugin” option and select the “Generate Code” option.
18. Then a generation window will appear. Choose a destination folder to where the generated code will be placed. Select “Android” as the target platform and click on the “Generate!” button.
19. Check if the target generation folder contains the application’s source code.
20. Congratulations! You have succeeded to design your first XIS-Mobile app!

**For More Info:**

- Consult https://github.com/xis-mobile/XIS-Mobile/
- Read the paper: André Ribeiro, Alberto Rodrigues da Silva, XIS-Mobile: A DSL for Mobile Applications, in Proceedings of ACM SAC'2014 Conference, 2014, ACM. http://isg.inesc-id.pt/alb/static/papers/2014/c120-ar-SAC-2014.pdf

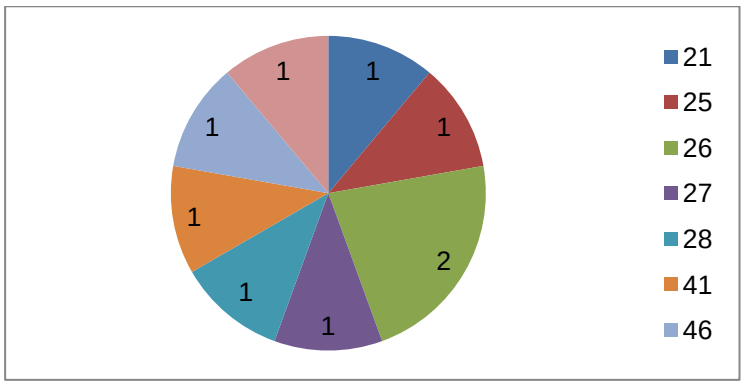
**Please, fill in the "Simple Test Session – Questionnaire" available online:**

http://goo.gl/M8shB1

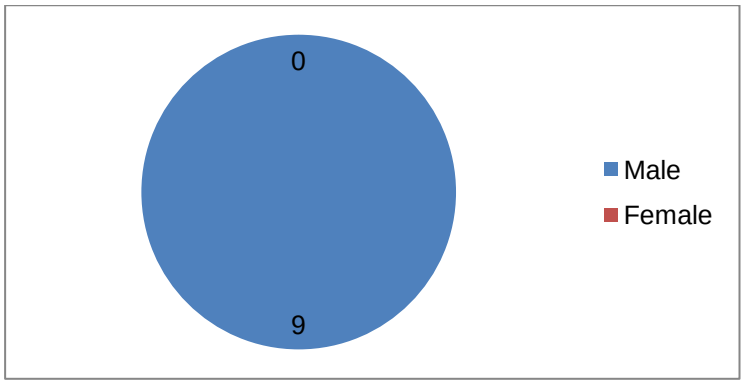
# C. User Session Questionnaire Results

**Total Number of Participants:** 9

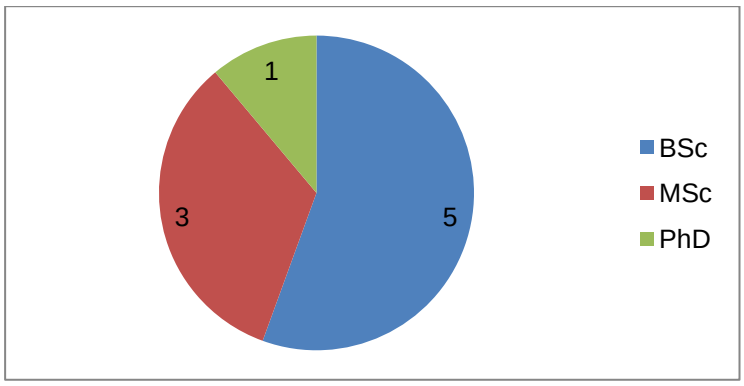
**Age:**



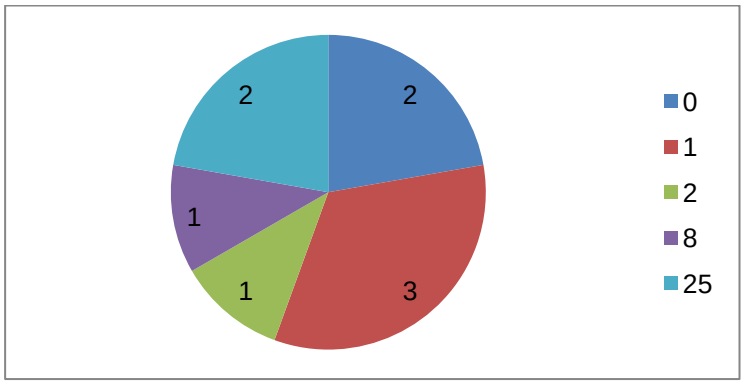

**Gender:**



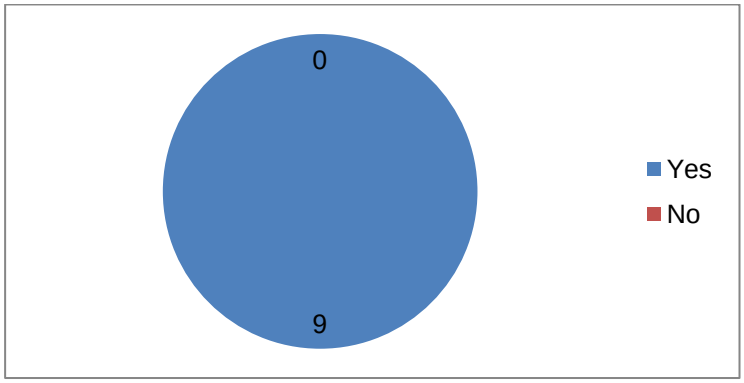

**Degree:**



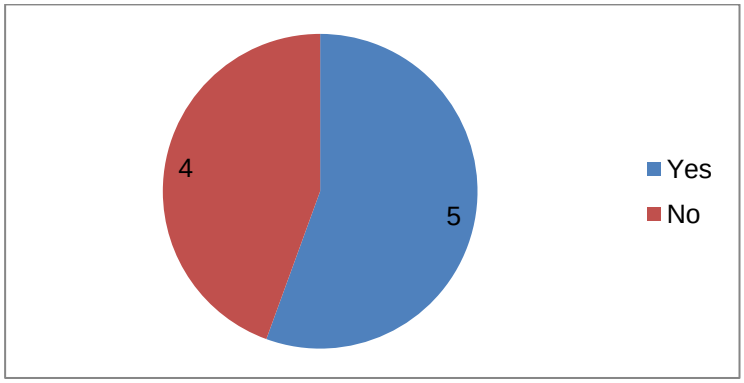
**Number of Years of Professional Experience:**



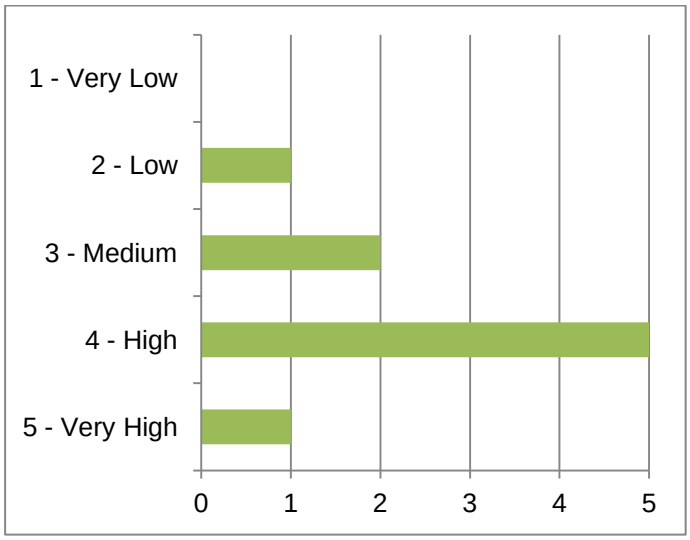

**Knowledge and Experience with UML:**



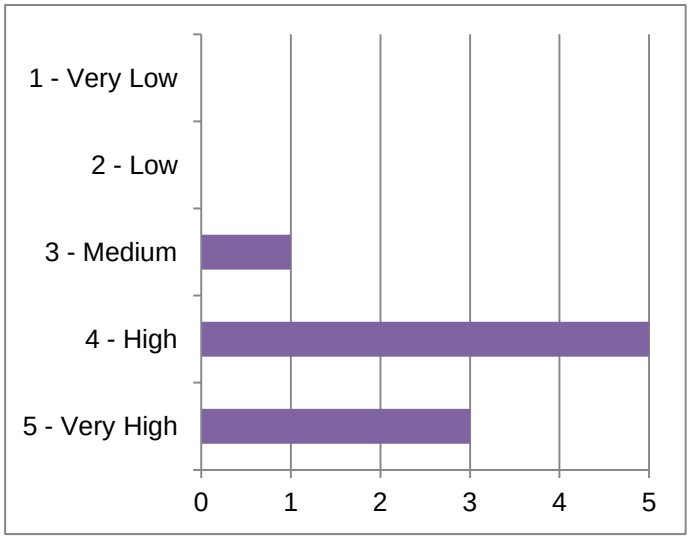

**Knowledge and Experience with Mobile Apps Development:**



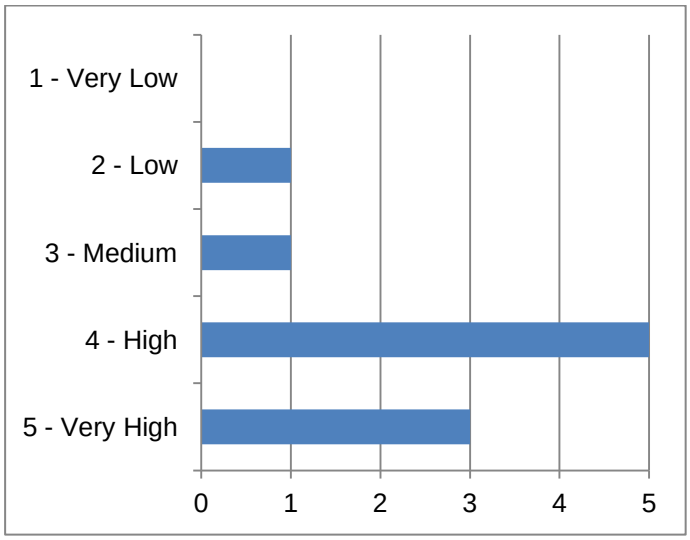
**XIS-Mobile Language [How suitable is the size (number of concepts) of the language?]:**



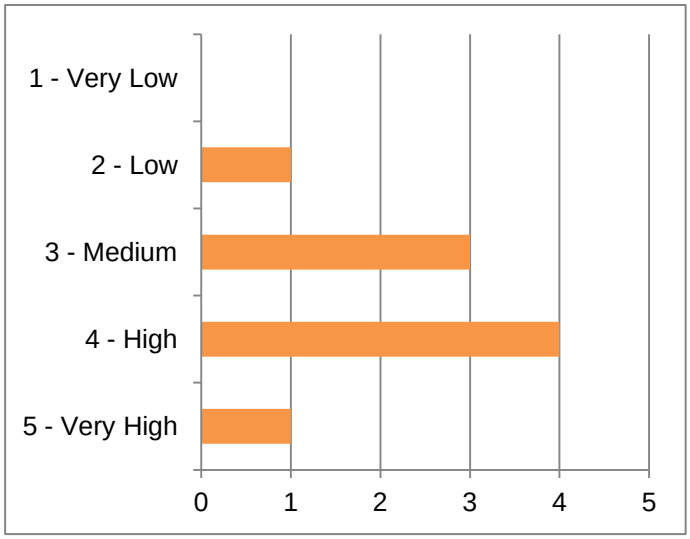

**XIS-Mobile Language [How easy to use is the notation used (defined as a UML Profile)?]:**



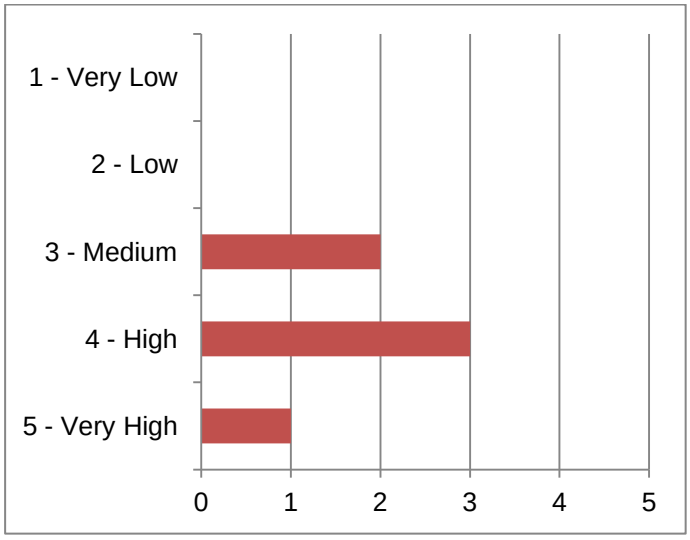
**XIS-Mobile Language [How easy to learn is the language without the UI concepts?]:**



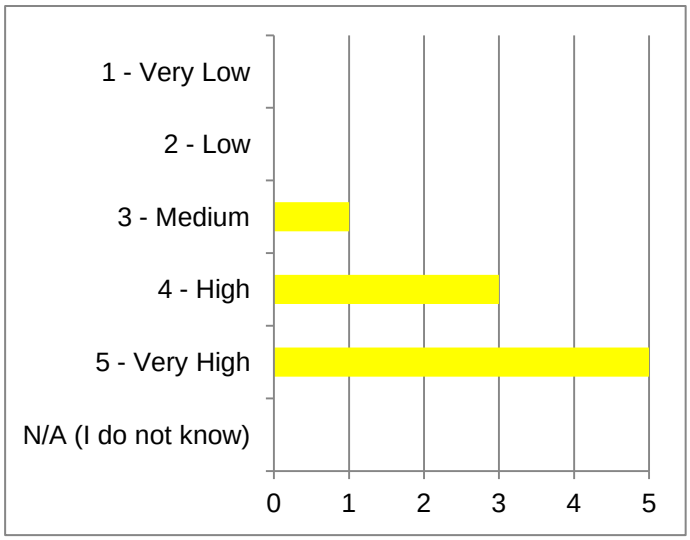

**XIS-Mobile Language [How easy to learn is the language with the UI concepts?]:**



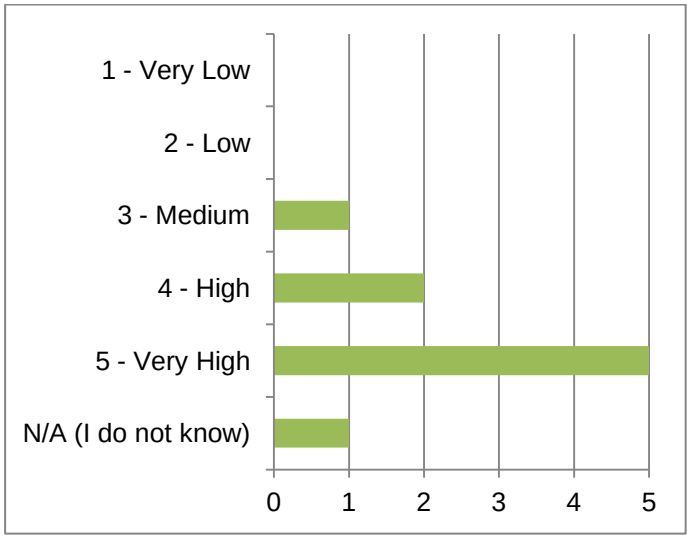
**XIS-Mobile Language [How suitable is the language for the Mobile Apps development domain?]:**



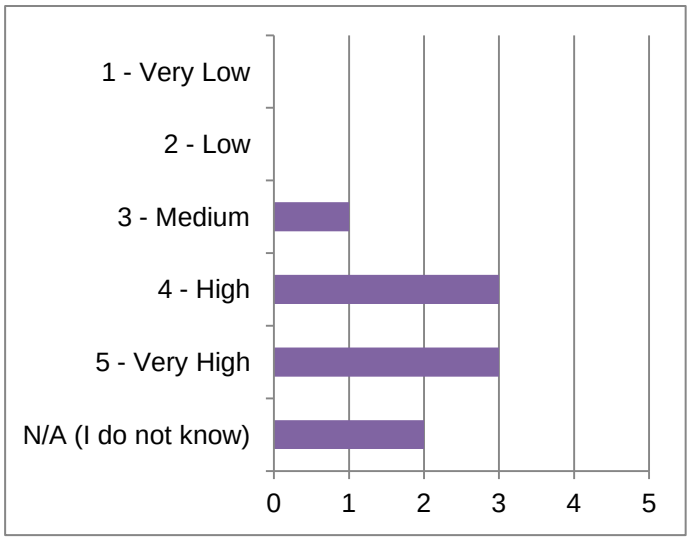

**XIS-Mobile Framework [How do you rate the usability of EA with the XIS-Mobile plugin?]:**



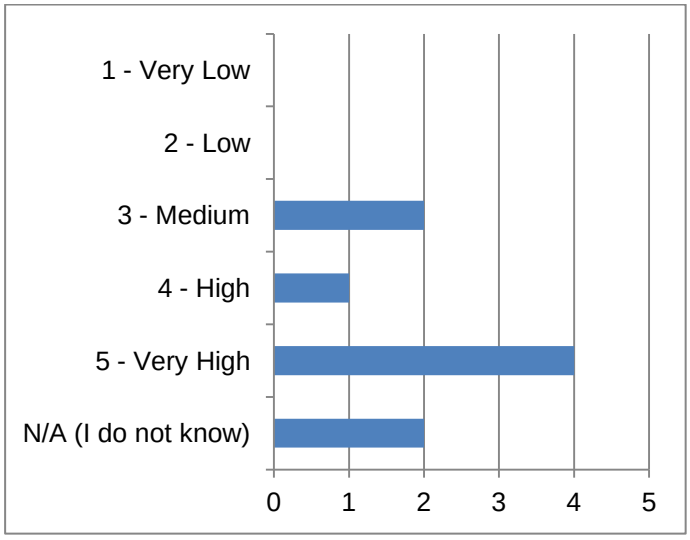
**XIS-Mobile Framework [How do you rate the usability of the Model Editor (Stereotypes, Toolboxes, Project template)?]:**



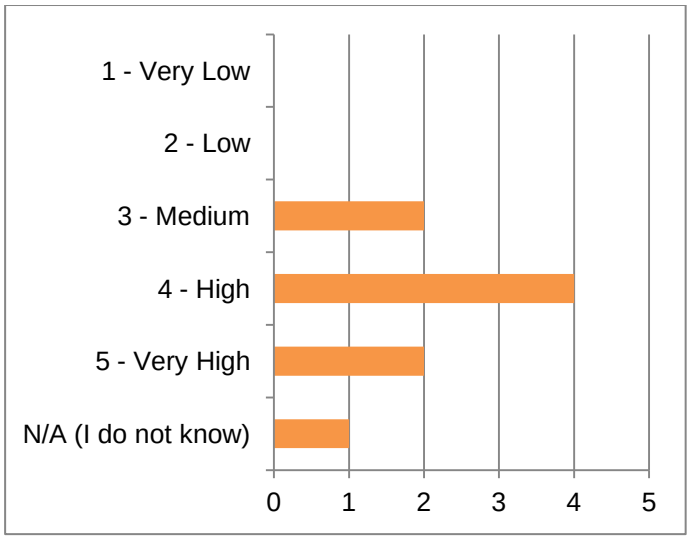

**XIS-Mobile Framework [How do you rate the usability of the Model Validator?]:**



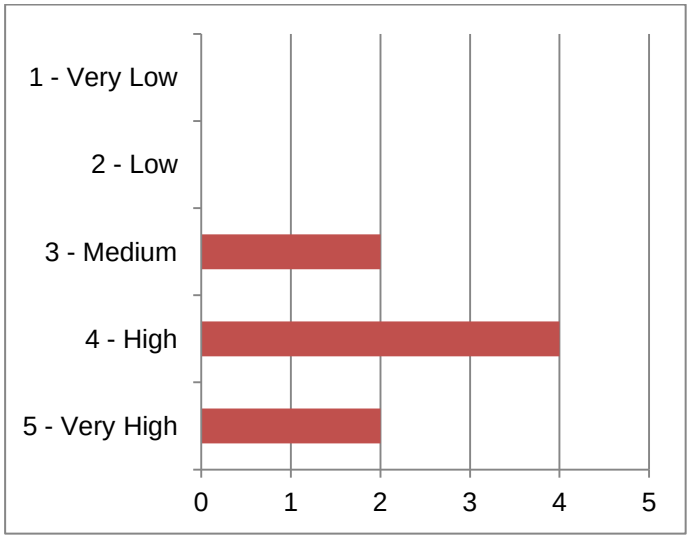
**XIS-Mobile Framework [How do you rate the simplicity of the Model-to-Model transformation (Model generation) process?]:**



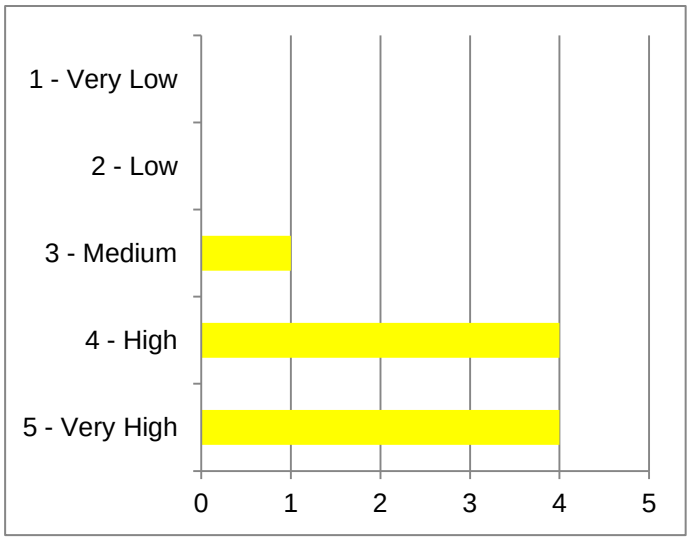

**XIS-Mobile Framework [How do you rate the simplicity of the Model-to-Text transformation (Code generation) process?]:**



**XIS-Mobile General Approach [How do you rate the productivity with XIS-Mobile comparing to the traditional software development process?]:**




**XIS-Mobile General Approach [Would you use such a tool on your own Mobile App projects?]:**




**Additional Comments (Suggestions, Problems, Bugs):**

Just minor bugs about the connectors. Apart from that, it's very easy for anyone who has already worked with EA.

Just necessary bug fixes and on to the road!

XIS-Mobile is definitively a tool that could address problems mobile apps development, especially the domain-independent feature. Allowing the user to think at a business-level, generates software more suited to the business requirements. On the other hand, I think the process, particularly the one from EA to Eclipse should be perfected along with some code bugs that should be refined. Nevertheless, I think the idea is great and the strategy is adequate.