The XIS-Web Technology

A Model-Driven Development Approach for Responsive Web Applications

João Eduardo dos Reis Peres de Seixas

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Dr. Alberto Manuel Rodrigues da Silva
Eng. André Filipe Oliveira Pinto Ribeiro

Examination Committee

Chairperson: Prof. Dr. António Manuel Ferreira Rito da Silva
Supervisor: Prof. Dr. Alberto Manuel Rodrigues da Silva
Member of the Committee: Prof. Dr. André Ferreira Ferrão Couto e Vasconcelos

June 2016

Acknowledgments

First, I would specially like to thank Professor Alberto Rodrigues da Silva, for his exceptional guidance through the development of this research work, that made me a better professional and also a better person. His knowledge and experience were a great contribution for this dissertation, and so, part of its quality is due to him.

I would also like to give a special thanks to my friend and colleague André Ribeiro. We shared many fun moments and he also provided support whenever I needed, even for the dumb questions (most of them were dumb questions) and this work would not have been possible without him, since he started this whole process with XIS-Mobile.

I would also like to thank all my friends at IST, specially those that shared with me the experience of developing the thesis, for all the moments of fun, but also for the moments of reflexion.

I would also like to thank my colleagues and students of Professor Alberto Rodrigues da Silva and all the other people that somehow participated in the development of this research work, namely during the evaluation phase.

A special thanks to my family, namely my mother, Mariana, and my father João who always provided support and believed that I could overcome the difficulties that were presented me and always stood by my side in this long journey. I dedicate this work to them.

Finally, I would also like to thank you, Sara, for putting up with me during all this time, for you encouraging words whenever I needed them and your everlasting patience. Part of this work is also yours, and I also dedicate it to you.

Abstract

This research work proposes a model-driven development approach for the design and creation of responsive web applications, called XIS-Web. This approach comprises a domain specific language, defined as a UML profile (the XIS-Web language) and a companion software framework (the XIS-Web framework).

The XIS-Web language captures concepts from the domain of web applications, enabling the specification of these systems in a way understandable by not necessary developer stakeholders. It divides the design process of web applications in six separate views, that promote a ”separation of concerns” principle, reducing the complexity of the whole process.

The XIS-Web framework provides support to the language by automatically generating the source code from the XIS-Web models, through Model-to-Text transformations. This framework also provides support for the design of the most complex models (the ones for the User Interface), by generating them automatically through Model-to-Model transformations. These transformation mechanisms reduce both the Time-to-Market for a web application and its production cost, ultimately increasing productivity.

Some preliminary evaluation of the approach as been made and the results are quite satisfactory. In a group of 12 users, using a scale of 1 to 5, users ranked XIS-Web language with a value of 4.1, XIS-Web framework with 4.53 and the general approach with 4.33.

XIS-Web is an innovative approach that uses the benefits of a model-driven development, to build responsive web applications, that can run on multiple devices.

Keywords: Model-Driven Development, Responsive Web Applications, Domain Specific Language, Device Fragmentation

Resumo

Esta dissertação propõe uma abordagem de desenvolvimento orientada por modelos, para o desenho e criação de aplicações *web responsive*, chamada XIS-Web. Esta abordagem inclui uma linguagem de domínio específico, definidas como um perfil UML (a linguagem XIS-Web) e uma ferramenta de suporte (a ferramenta XIS-Web).

A linguagem XIS-Web captura conceitos relacionados com o domínio das aplicações *web*, permitindo a especificação deste tipo de sistemas, de uma maneira compreensível por *stakeholders* que não são necessariamente programadores. Divide o processo de concepção de uma aplicação web em seis vistas diferentes, promovendo um princípio de "separação de preocupações", reduzindo a complexidade do processo em geral.

A ferramenta XIS-Web forneced suporte à linguagem ao gerar automaticamente o código fonte dos modelos XIS-Web, através de transformações Modelo-para-Texto. Esta ferramenta também fornece suporte à criação das vistas mais complexas da linguagem (as correspondentes à Interface do Utilizador), gerando as automaticamente através de transformações Modelo-para-Modelo. Estes mecanismos de transformação reduzem o *Time-to-Market* e o custo de produção das aplicações *web* e em última instância aumentam a productividade.

Foram realizadas avaliações preliminares a esta abordabem e os resultados foram bastante satisfatórios. Num grupo composto por 12 pessoas, utilizando uma escala de 1 a 5, os participantes da avaliação classificaram a linguagem XIS-Web em 4.1, a ferramenta XIS-Web com 4.53 e a abordagem geral com 4.33.

A tecnologia XIS-Web tem uma abordagem inovativa que utiliza as vantagens do desenvolvimento orientado por modelos para criar aplicações *web responsive*, que podem ser reproduzidas em multiplos dispositivos.

Palavras-Chave: Desenvolvimento Orientado por Modelos, Aplicações *Web Responsive*, Linguagens de Domínio Específico, Diversificação de Dispositivos

Contents

List of Tables

List of Figures

Acronyms

APD Abstract Presentation Diagram. 24

API Application Programming Interface. 19, 42

ATL ATLAS Transformation Language. 18, 23

CASE Computer-Aided Software Engineering. 19

CLOC Count Lines of Code. 51

CMOF Complete MOF. 16

CMS Content Management System. 59

CRUD Create, Read, Update, Delete. 35, 38, 44

CSS Cascading Style Sheets. 5, 20, 24, 26, 48, 59

DSL Domain Specific Language. 4, 5, 11, 13, 15–19, 25, 26, 31, 53, 59, 60

DSML Domain Specific Modeling Language. 15, 17

EA Enterprise Architect. 5, 9, 10, 41–43, 54

EMF Eclipse Modeling Framework. 5, 8, 10, 16, 42

EMOF Essential MOF. 16

GPML General Purpose Modeling Languages. 15, 18, 19

HPG Hera Presentation Generator. 23

HTML Hypertext Markup Language. 5, 6, 9, 18, 23, 24, 26, 48, 51, 57, 59

IDE Integrated Development Environment. 19

IDSS Information and Decision Support Systems. 4

INESC-ID Instituto de Engenharia de Sistemas e Computadores - Investigação e Desenvolvimento. 4

IoT Internet of Things. 3

JSON JavaScript Object Notation. 62

JSP Java Service Provider. 23

LOC Lines of Code. 51, 57

M2C Model-to-Code. 18

M2M Model-to-Model. 11, 14, 18, 24, 26, 27, 41–43, 57

M2T Model-to-Text. 14, 18, 23, 24, 26, 27, 38, 41, 47, 51

MDA Model-Driven Architecture. 13, 14, 16

MDD Model-Driven Development. 3, 5–8, 11, 13, 14, 18, 19, 23–26, 41, 57, 59, 60

MDE Model-Driven Engineering. 13, 14

MDG Model-Driven Generation. 42, 59

MOF Meta-Object Facility. 16–18, 42

MOFM2T MOF Modelt to Text Transformation Language. 16

MTL Model to Text Language. 42

MVC Model-View-Controller. 24

NAD Navigation Access Diagram. 24

NAD2APD Navigation Access Diagram to Abstract Presentation Diagram transformation. 24

OCL Object Constraint Language. 17, 18, 42, 47

OMG Object Management Group. 14–18, 42

OO Object-Oriented. 3, 16

OO-H Object-Oriented Hypermedia. 24

PHP PHP: Hypertext Preprocessor. 24

PIM Platform-Independent Model. 6, 14, 16, 24, 25, 56, 59

PRML Personalization Rule Modeling Language. 56

PSM Platform-Specific Model. 14

QVT Query-View-Transformation. 18

RAD Rapid Application Development. 24

RDF(S) Resource Description Framework (Schema). 23

RMM Relationship Management Methodology. 23

RWD Responsive Web Design. 20

SQL Structured Query Language. 5

TTM Time-To-Market. 3, 6, 55, 60

UI User Interface. 13, 21, 26, 38, 39, 43, 52, 53, 57, 60, 61

UML Unified Modeling Language. 3, 5, 8, 10, 11, 13, 15–19, 23–26, 42, 48, 52–54, 56, 57, 59, 60

UWE UML-based Web Engineering. 23, 24

WML Wireless Markup Language. 23

XMI XML Metadata Interchange. 10, 16, 42

XML eXtensible Markup Language. 10, 16

XP eXtreme Programming. 3

XSL eXtensible Stylesheet Language. 56

YACC Yet Another Compiler Compiler. 18

Chapter 1

Introduction

The management of complexity in software systems is one of the most important disciplines in Software Engineering. As time passes, systems tend to become more complex and sophisticated because they try to solve increasingly harder problems. As effect, the size of the project and the learning curve of the technologies used increases.

Fortunately, Software Engineering has made some key contributions regarding the software development process. Through the introduction of software development methodologies (e.g. Scrum, eXtreme Programming (XP) or Kanban) , software development approaches (e.g. Incremental, Iterative, Waterfal), mechanisms of abstraction that encapsulate the complexity (e.g. Object-Oriented (OO) programming and UML) and test-driven development approaches that assure quality of the software systems [5][6].

Since 2014[1], the number of mobile devices as surpassed the number of desktop computers, shifting the paradigm of software development. We are currently living the time of the Internet of Things (IoT). The Internet of Things can be defined as the interconnection of sensing and actuating devices, providing the ability to share information across platforms through a unified framework, developing a common operating picture for enabling innovative applications. This is achieved by seamless large scale sensing, data analytics and information representation using cutting edge ubiquitous sensing and cloud computing [7].

A device with web browsing capabilities nowadays can be virtually anything, from a watch, to a television to a refrigerator. Giving that these devices are fundamentally different from each other, typically the content produced for them follows one of two approaches: (1) native code for the underlying platform of the device, (2) web-based code, that is independent from the device that is running it. The first, despite achieving better performances causes lots of overhead in having to develop the same specification several times using different technologies, so the second one pulls ahead, by only requiring one specification to be developed. This ultimately reduces an application's cost and TTM [8].

The usage of Web technologies solves the problem of device fragmentation, leaving the problem of managing complexity in software systems. Software Engineering recommends the usage of documentation (like models, requirements and design documents) in order to handle the system's complexity, and to have a common understanding of it by all its stakeholders. But given that a system suffers many changes throughout its development process, managing simultaneously the source code and the documentation is usually a time-consuming and error-prone task [1]. One of Software Engineering's areas, Model-Driven Engineering and its development process, Model-Driven Development (MDD), tackle this issue by having models as the main artefact of the system and automatically generated everything else (from model transformations), namely the code and the documentation associated [9][3].

[1] http://www.internetlivestats.com

1.1 Context

This research work has been developed at the Information and Decision Support Systems (IDSS) of INESC-ID (Instituto de Engenharia de Sistemas e Computadores - Investigação e Desenvolvimento), under the supervision of Professor Alberto Rodriges da Silva, regarding the Master Degree in Information Systems and Computer Engineering at Instituto Superior Técnico, Universidade de Lisboa. This research work combines the interest of the area of MDD with the development of responsive web applications.

The Information and Decision Support Systems is responsible for a variety of projects in the area of Model-Driven Development and Web Applications Development, namely the projects related with the research initiatives of MDDLingo and RSLingo:

- **MDDLingo Initiative.** Aggregates projects that are related to Model-Driven Development, namely the ones of the XIS-* family like XIS, XIS-Mobile, XIS-CMS, XIS-Web and XIS-Analytics.
- **RSLingo Initiative.** Aggregates projects that are related with Requirements Specification, namely RSL-IL, RSLingo4Privacy, RSLingo-Studio and ReBox.

Moreover the XIS profile and the XIS-Mobile technology served as starting points for this research, due to the goal of using a higher level representation for applications (in the case of XIS) and the automatic generation of applications, from the a DSL based on the problem domain (in the case of XIS-Mobile) [10][1]. Unlike XIS or XIS-Mobile, this research work focuses on responsive web applications. For these reasons, we named the proposed solution of XIS-Web. XIS-Web is composed of a DSL and a supporting framework for responsive web application development. The relation with XIS and XIS-Mobile is described in more detail in Section 3.1

1.2 Problem Definition

Given the fragmentation of devices that run web applications (from watches to computers), developers often find themselves having to design the same applications multiple times to cater for the differences in size and shape of the devices. Ultimately this ends up introducing even more complexity on the software development process and is something that consequently brings higher costs to production and larger time to market, considering the amount of redundant work.

The main problems addressed in this research work are: Complexity Management in Software Development and Device Fragmentation. Considering these problems, the following research questions summarize the situation to address:

- **R1.** How to specify a responsive web application in a platform-independent way?
- **R2.** What are the key concepts needed to specify a responsive web application?
- **R3.** Which User Interface patterns are the more suitable for responsive web applications?
- **R4.** How to make the responsive web application device independent?

In the ideal scenario, the web application is developed once and works seamlessly on any device achieving what is known as portability. Some approaches have been made to address this issue, namely through the use of cross-platform development tools and new versions of web technologies that bring the necessary flexibility for it. Also MDD approaches like the one proposed in this dissertation are examples of solutions focused on solving these problems.

1.3 Proposed Solution

This research work proposes XIS-Web, a DSL with a companion framework, as a solution to the problem presented previously, adopting an MDD approach.

The XIS-Web Language is a graphical DSL that is a UML Profile specifically build for the modeling of responsive web applications, putting to use concepts specific to the web applications domain. It has a multi-package organization, and packages have one or more views (of the system) inside. Currently it supports two design approaches: the dummy approach and the smart approach, being the main difference between them the usage of Model-to-Model transformations (by the smart approach) to automatically generate the views that correspond to the User Interface (these that are the most complex ones). The dummy approach requires the manual creation of all views.

In order to generate the responsive web applications, the supporting MDD-based framework was developed using Sparx Systems Enterprise Architect[2] (Enterprise Architect (EA)) and the Eclipse Modeling Framework[3] (EMF [11]), namely through the plugin Acceleo[4]. The XIS-Web framework can be divided in four major components (see Figure 1.1): (1) the Visual Editor, that allows the creation of the web application using the XIS-Web language, (2) the Model Validator, that applies a set of rules specifically design for the XIS-Web language, to see if the models produced by the Visual Editor are compliant with the constraints, (3) the Model Generator, that is able to generate the more complex models, taking as input the other models created by the user, using Model-to-Model transformations and (4) the Code Generator, that automatically generates the source code of the responsive web application that was built by the designer, using Model-to-Text transformations that are based on code templates specified using Acceleo.

In order to achieve the desired behavior of responsiveness, XIS-Web generates its applications using the latest web technologies, namely HTML5, JavaScript, WebSQL[5] and CSS3[6] (via the Bootstrap library[7]). HTML5 defines the markup of the mage, while CSS3 defines the styles for each element defined in HTML5. XIS-Web makes use of the Bootstrap library for the layout because it offers the desired behaviour in terms of responsiveness for the widgets present in the language. The logic of the application is developed using JavaScript, that manipulates the elements defined in the HTML5 markup. For data layer we chose WebSQL because it allows for an SQL Database inside the browser of the device.

XIS-Web pretends to be a tool that helps not only people that do not have experience with software development, namely responsive web application development, but also experienced developers by generating the skeleton of responsive web applications. The developer can start from the already generated code from the framework and modify it as he sees fit (like improving the User-Interface). Currently XIS-Web is more suited for applications that are typically business (and form) oriented.

The goals of this dissertation are as follows:

- **G1.** Design and implement a DSL that allows the development of responsive web applications.
- **G2.** Implement the tools that support such language:
  - A visual editor that allows the authoring of the application's models using a UML profile;
  - A model validator enforces the language's constraints on the user created models;
  - A model generator that automatically generates the models related with the User Interface;

[2] http://www.sparxsystems.com.au/products/ea/
[3] http://eclipse.org/modeling/emf/
[4] http://www.eclipse.org/acceleo
[5] http://www.w3.org/TR/webdatabase/
[6] http://developer.mozilla.org/en-US/docs/Web/CSS/CSS3
[7] http://getbootstrap.com

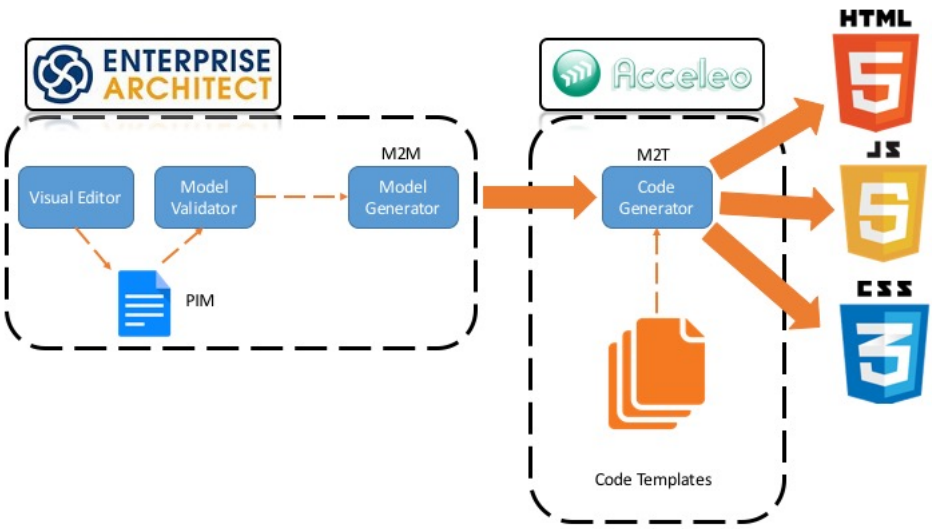


Figure 1.1: Different technologies of the XIS-Web framework and simplified development approach (adapted from [1]).

  - A code generator that automatically generates the application's source code, from the existing models.

- **G3.** Evaluate the quality of the proposed system in the following ways:
  - The usability and usefulness of the language and framework through a pilot-user test session;
  - The feasibility of the system through case study applications;
  - The system's added value through comparation with related work.

- **G4.** Automatically generate more than 70% of an application's source code.

1.4 Thesis Claim

This dissertation's thesis claims that combining an MDD-approach with the usage of web technologies like HTML5 and JavaScript, results in: (1) productivity gains, namely by reducing the complexity of the system that is being developed, its Time-to-Market (TTM) and its cost; (2) the portability of the application, created with a stack of technologies that allow it to have responsive design and independence from the device that is running on.

This claim can be achieved using Platform-Independent Models (PIM) create with the XIS-Web language, and subsequent transformation using the XIS-Web framework. The process starts with the transformation of the simple models created by the user, to more complex models that represent the User Interface and its navigation flow (through Model-to-Model transformations). Then, using Model-to-Text transformations, the generation of the source code of the corresponding responsive web application. Considering that this approach can generate applications that are comparable to a custom made application, I also claim that the use of XIS-Web contributes to an improvement of productivity.

Moreover, three case studies were developed using the proposed method, from model to source code. There was a pilot-user test session that contributed to assess the acceptance of the technology and made the participants aware of the enhancements to productivity that it can offer.

1.5 Methodology

This research work followed the methodology that is typically used in the research group, that is the Action Research methodology, depicted in Figure 1.2 [2]. This methodology, suggest an iterative approach that can be divided in five steps, executed in a client-system infrastructure:

- **Diagnosing:** Identification the problem domain and its relevance;
- **Action Planning:** Definition of the proposed solution and the steps required to solve the problem identified in the previous step;
- **Action Taking:** Implementation of the solution planned in the previous step;
- **Evaluating:** Assessment of the solution development in the previous step (see if the actions performed have succeeded to solve the problem);
- **Specifying Learning:** The lessons learned during the cycle and preparation for the next action research cycle.

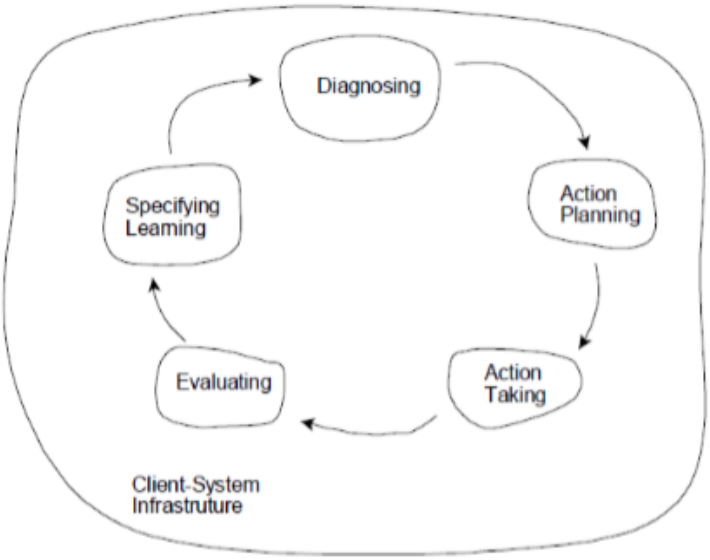


Figure 1.2: Action Reserach Methodology cycle, extracted from [2]

This research work, that was developed over the course of 12 months, can be divided in five iterations of the Action Research methodology.

First Iteration - Month 1 to Month 3:

This iteration consists on this research work planning. The earlier tasks involved getting familiar with the research work done by other members of the research group, in particular with the XIS-Mobile technology. Afterwards, some investigation was made, to understand how web technologies could fit in an MDD-based approach like XIS-Mobile and what would be the added value of this research work.

At first, it was decided that XIS-Web would act as an extension of XIS-Mobile, with the purpose of generating mobile applications for any platform, doing so using web technologies and it was called XIS-Mobile: HTML5. The remainder of the research done in these months, revolved around determining the feasibility, (namely which technologies to use) and usefulness of such task and also about investigating similar frameworks (i.e., frameworks that could generated code for several platforms that were somehow

using web technologies to do so). Next, the five steps of the Action Research Methodology for this iteration are described.

- **Diagnosing:** Getting to know the research developed in the research group and understanding how to integrate web technologies in XIS-Mobile.
- **Action Planning:** Planning of a preliminary version of the XIS-Mobile: HTML5 language and planning of the XIS-Mobile: HTML5 framework (both extensions to the existing XIS-Mobile technology).
- **Action Taking:** Decisions regarding the technologies to use for code generation (namely: HTML5, JavaScript and CSS3). Investigation of related technologies and preliminary definition of the XIS-Mobile: HTML5 language.
- **Evaluating:** Validation with other members of the research group, about the added value and purpose of the XIS-Mobile: HTML5 technology.
- **Specifying Learning:** From the previous step, it was possible to understand that the added value of XIS-Mobile: HTML5 would be reduced, since it would be attacking the same problems that XIS-Mobile was, that was the reduction in complexity of software development and the platform fragmentation of the mobile market. Instead, we decided to change the purpose of the language and the focus of the research work by creating an MDD-based approach that allows for the creation of responsive web applications, that run seamlessly on any device. By keeping the MDD-based approach proposed by XIS and used by XIS-Mobile, XIS-Web can also contribute for the reduction in software development complexity, while the usage of web technologies that have a responsive behavior depending on the device, can handle the issue of device fragmentation.

Second Iteration - Month 4 to Month 6:

The second iteration began with the definition of the XIS-Web language, no longer XIS-Mobile: HTML5. Starting from the XIS-Mobile language (a successful implementation of the XIS approach), changes to the language were made, in order to adapt the concepts to the domain of web applications. While mobile applications are somewhat related to web applications, some concepts are specific to each other.

With the closing of the language profile, the work shifted to the code generation. This preliminary work performed on code generation, was made entirely inside the Eclipse Modeling Framework (EMF), due to the short development cycle of: (1) creating a model using Papyrus, (2) applying it the stereotypes defined in the XIS-Web language, (3) proceeding to code generation based on the developed templates (using Acceleo) and (4) debug the generated code and repeat the process once again. In order to help with the code templates for Acceleo, a case study was manually developed that would serve as goal for the generated code.

At this stage it was also developed the case study application of ”dDocs”, that served as proof of concept for both the feasibility and usefulness of XIS-Web, considering it was real-world application that was being developed by a colleague in the research group, in a business scenario.

With the code generator in a more stable state, the language profile was specified in EA in order to prepare the work in the areas of Model Validation and Model-to-Model transformations. Next, the five steps of the Action Research Method for this iteration are described.

- **Diagnosing:** Analysis of the issues from the previous iteration, namely the purpose and focus of the technology being developed. Decision to change the purpose to the automatic development of responsive web applications using the XIS-Web language defined as a UML Profile, based on the XIS-Mobile language.

- **Action Planning:** Definition of the concepts that should be redefined from the XIS-Mobile language, to be added to the XIS-Web language and the new concepts that take part in this language. Specification of a case study to serve as goal for the code generation and learning how to implement the code generation process.

- **Action Taking:** Specification of the XIS-Web language, and preliminary implementation of the code generator for HTML5 and JavaScript. Manual implementation of the To-Do List App case study to serve as goal for the code generation and later, the development of the "dDocs" application to work as a proof of concept for the feasibility and usefulness of XIS-Web. Later, the development environment was changed to include Sparx Systems Enterprise Architect.

- **Evaluating:** The evaluation of the code generation process was made by comparing the code generated to the one that was manually developed. The state of the work was evaluated with the implementation of the "dDocs" application, an application being manually developed by a colleague from the research group, in a business scenario.

- **Specifying Learning:** Success on reaching the desired state of code generation (where the application that was manually developed could be automatically generated) and success in demonstrating the features that XIS-Web possessed at that current time with the demonstration of the "dDocs" application. Now there was the need to work on the modeling part, namely in the Model Validation and the Model-to-Model generation.

Third Iteration - Month 7 to Month 9:

This iteration consisted in the definition of the Model-to-Model and Model Validation transformation rules and customization of the Visual Editor. First, since XIS-Web builds applications with the purpose of them being responsive to any device, there was the need to define patterns that could abide to this concept. Once the models that are the result of the Model-to-Model transformation are defined and the framework could apply the intended transformations, some Model Validation rules were defined, in order to apply the constraints that these new concepts brought.

Once these two tasks were completed, and in order to allow the developers to properly model applications using the XIS-Web language, there was the need to customize the Visual Editor (Enterprise Architect) in order to support the language. This was done by creating a custom project template for XIS-Web, that would automatically provide a six view organization for the project and create stub diagrams inside each view. Custom diagrams were also created, with custom toolboxes that allowed for the stereotypes available for each diagram to be used. This was then integrated in the EA environment using its MDG Technology mechanism. Next, the five steps of the Action Research Method for this iteration are described.

- **Diagnosing:** With the code generation mechanism in a stable state, the work in this iteration revolved around modeling of the application.

- **Action Planning:** Definition of the Model-to-Model and Model Validation rules to be implemented in the XIS-Web framework, and the customizations to be made in the Visual Editor.

- **Action Taking:** Implementation of the Model-to-Model and Model Validation rules and customization of Enterprise Architect using its MDG Technologies mechanism.

- **Evaluating:** The evaluation of the Model-to-Model rules was made via observation of the generated model and by comparison to models that were manually created and compliant with the stereotypes of the XIS-Web language. The validation of the Model Validation rules was performed by introducing errors in the specified models and seeing if the Validator would catch them.

- **Specifying Learning:** After some iterations on the Model-to-Model and Model Validation mechanisms, we got a successful implementation of both. The customization of EA in order to provide support for modeling applications using the XIS-Web language was also successful. Currently the modeling part (Visual Editor, Model Validator and Model Generator) were working in EA and the Code Generation part was working in Eclipse. The following task was to have these two parts working together.

Fourth Iteration - Month 10 to Month 11:

In this fourth iteration, the main task was to have the XIS-Web framework working as a whole and to accomplish this, there was the need to have the Code Generation part to work as standalone (to be used by EA). To accomplish this task, a XML parser had to be developed, in Java, using Eclipse. This XML parser was the responsible for parsing the XMI generated by EA and convert it into XMI that the standalone code generator could understand. This discrepancy happens because EA and EMF have different ways of representing concepts in UML. Once the XML parser was successfully developed, a new case study, the TimeSlot Booking App was developed in order to test the framework as a whole. Next, the five steps of the Action Research Method for this iteration are described.

- **Diagnosing:** From the last iteration, there was the problem of the framework not being able to work as a whole, due to discrepancies in the XMI generated by EA and the XMI accepted by EMF.

- **Action Planning:** Planning of the XML parser that would convert the XMI from EA into XMI understandable by EMF.

- **Action Taking:** Development of the XML parser that converts XMI from EA into XMI understandable by EMF.

- **Evaluating:** The evaluation of this iteration was the fact that the framework was able to work as a whole, i.e., the translation of XMI generated by EA to XMI understood by EMF was a success.

- **Specifying Learning:** Successful implementation of the XML parser. Given that both the language and framework were in a stable state, work began on the evaluation of XIS-Web.

Fifth Iteration - Month 12:

Finally, in this iteration, the tasks revolved around preparing the evaluation for XIS-Web.

The evaluation method was one that is commonly used in this research group. A pilot-user test session with participants not associated with the research group. In order to do so, it was needed to package the XIS-Web platform into an executable that would run on the participant's computers, provided that they had Java and Enterprise Architect installed.

This session would focus on the evaluation of three aspects of XIS-Web: the Language, the Framework and the General Approach. In order to do so, the participants were asked to follow a script, in which they were given the proper indications in order to build an application using XIS-Web. The application built corresponded to the TimeSlot Booking App. After the session the participants were asked to fill a questionnaire regarding those three aspests of the framework. Next, the five steps of the Action Research Methodology for this iteration are described.

- **Diagnosing:** Identification of the evaluation method to be applied for XIS-Web.

- **Action Planning:** Planning of the pilot-user test session that included: defining the aspects to be evaluated, how the session would occour and defining the script for the participats.

- **Action Taking:** Specification of the script to be used by the participants (the "User Session Guide"), that described the case study to be modeled and the session rules. Elaboration of the participation questionnaire.
- **Evaluating:** Evaluation session with the pilot-users.
- **Specifying Learning:** Analysis of the results obtained from the participants' answers to the questionnaire, as well as feedback provided by them during the session. Overall, the results collected were positive and serve to demonstrate XIS-Web's usefulness and feasibility as a proof of concept. This session also provided future research directions, like the addition of more M2M patterns for generation and the reduction of the number of stereotypes that makes the language not so easy to learn.

1.6 Outline

The remainder of this dissertation is organized as follows:

Chapter 2. In this chapter we provide the main background concepts that are related to this work with particular emphasis on two main topics: Model-Driven Development (MDD) and Responsive Web Application Development. Also it surveys technology somehow related to this research work.

Chapter 3. In this chapter we present an overview of the proposed approach.

Chapter 4. In this chapter we present the XIS-Web language, a DSL that is a UML Profile, that allows the creation of responsive web applications in a platform-independent way.

Chapter 5. In this chapter we describe the XIS-Web framework. A support framework for the language, that allows for automatic generation of the applications created with it, enforcing a MDD approach.

Chapter 6. In this chapter we present and analyze the evaluation performed to XIS-Web.

Chapter 7. In this chapter we discuss the main conclusions of this work, along with future work perspectives.

Chapter 2

Background

This chapter introduces the concepts that are related to XIS-Web, namely Model-Driven Development (MDD) and Web Applications.

Section 2.1, provides a description for MDD and some concepts that derive directly from it, namely Model-Driven Engineering and Model-Driven Architecture an initiative for MDD. Considering the contents of this work it is also important to explain some concepts like modeling languages and metamodeling, UML profiles and Domain Specific Languages (DSLs).

Moreover, Section 2.2 explains concepts related to Web Applications, taking particular emphasis on a sub-set of these, the Responsive Web Applications. It also describes the development process, UI Design Patterns.

Finally, Section 2.3 presents a survey of technologies that can be compared to this research work, focusing on languages that allow the modeling of web applications and frameworks that enable rapid web application development.

2.1 Model-Driven Development

This section explains concepts from Model-Driven Development (MDD) that are also related with this research work. First, Section 2.1.1 gives a description of Model-Driven Engineering (MDE) and Section 2.1.2 provides explanation for Model-Driven Architecture (MDA). Then, Section 2.1.3 introduces the concepts of modeling language and metamodeling, while in Section 2.1.4 and Section 2.1.5, the concepts of UML Profile and DSL are defined, respectively. Other concepts related to MDD that are important to mention are the Model Transformations and the Language Workbench in Sections 2.1.6 and 2.1.7 respectively.

2.1.1 Model-Driven Engineering

Model-Driven Engineering (MDE) is a methodology that focuses on designing the problem domain, components and all the topics in models, being these the primary entities to the process. MDE can be applied to many context besides software development, through the create of models to activities, process or requirements [12].

A concrete initiative of MDE is the Model-Driven Development (MDD) that focuses on the MDE methodology subset of models that represent the development process. This approach makes a similar claim as the one from MDE, stating that models used during development of the software system are primary artifacts, and the usage of these through the steps of the development process (from design to

maintenance) will directly improve the quality of the system by providing an environment less error-prone and more productive [13].

In conclusion, Figure 2.1 depics MDE related terminology and some concrete approaches.

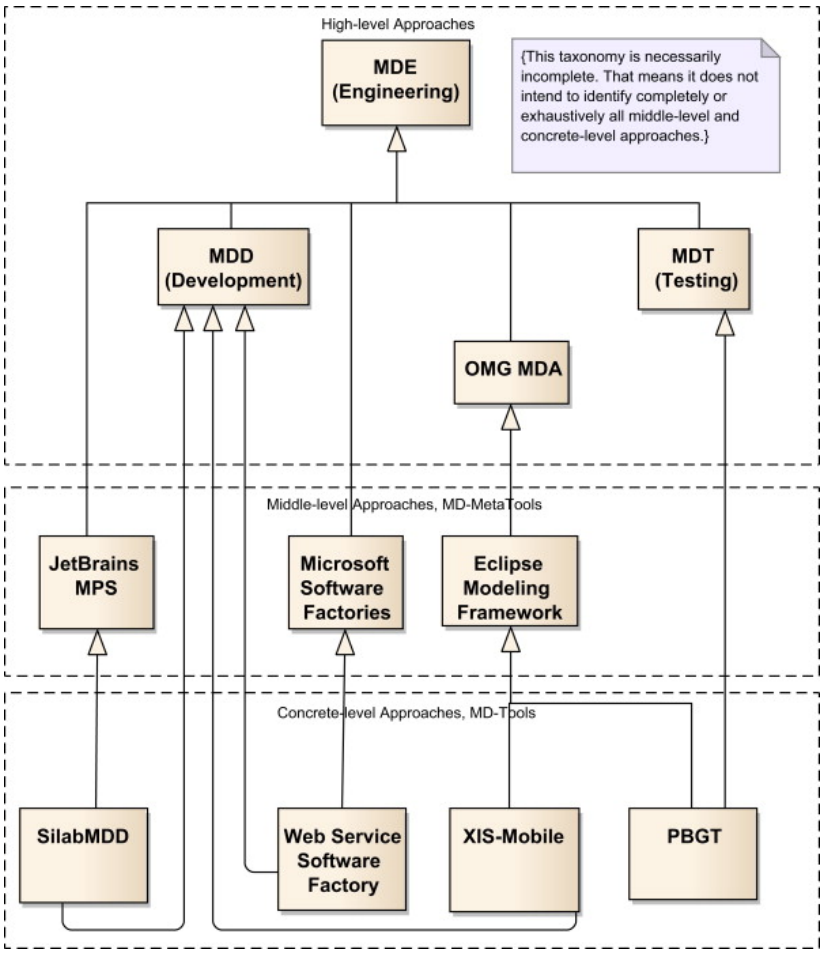


Figure 2.1: MDE related terminology and concrete approaches (extracted from [3]).

2.1.2 Model-Driven Architecture

The MDD initiative proposed by the Object Management Group (OMG) is Model-Driven Architecture (MDA). This software development process provides a set of guidelines and principles that allow the complete specification of a system based on models. Models for the same system can present different levels of abstraction, depending of the part that they are characterizing. Two types of model are considered for this initiative: (1) Platform-Specific Models (PSM) and (2) Platform-Independent Models (PIM). A PSM is a specification of a system that is built for a particular platform, using concepts that belong to it. A PIM is also a specification of a system but designed with an higher level of abstraction that allows for the concepts of the domain problem to be expressed in a platform-independent way [14]. A PIM can then be translated to one or more PSMs via Model-to-Model (M2M) transformations. The ultimate goal of either PIM or PSM models is to generate the system's source code (that needs to be platform specific), being these transformations are called Model-to-Text (M2T) transformations [15].

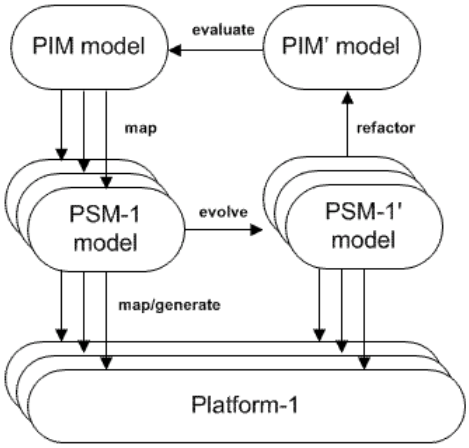


Figure 2.2: Example of M2M and M2T transformations in the MDA process.

2.1.3 Modeling languages and metamodeling

There is not a consensual definition of metamodels. However, some authors come up with the following definitions: (1) metamodel is a model that defines the language in which a model is expressed; (2) a metamodel is a model of a language of model or (3) a metamodel is a specification model for which systems under study being specified are models in a certain modeling language [3][16].

From these definitions we can understand that a Model is an element of a Modeling Language, which in turn is defined by a Metamodel. A modeling language can be defined as: the "set of all possible models that are conformant with the modeling language's abstract syntax, represented by one or more concrete syntaxes and that satisfy a given semantics. Additionally, the pragmatics (of a modeling language) helps and guides how to use it in the most appropriate way." [17].

To clarify the concepts presented, it is important to explain the meaning of abstract and concrete syntax. The abstract syntax of a language starts with the capture and identification of the concepts, abstractions and relations underlying the application domain. The abstract syntax defines all the concepts of a language, and their respective relations, so it is important that those names are related to the application domain in order to be easily understood by its users [18]. On the other hand, the concrete syntax of a modeling language is its notation, i.e., the way users will learn and use it. Since users have direct experience with the concrete syntax of a modeling language, its success will depend on the right balance between simplicity and expressiveness. Concrete syntaxes can have different notations, like graphical, textual, tabular, form-based or combinations of them. So, one modeling language can have multiple concrete syntaxes [19].

Modeling languages belong to one of two categories: (1) General-Purpose Modeling Languages (GPML) or Domain Specific Modeling Languages (DSL/DSML). The first type of modeling language typically presents a larger number of generic constructs, encouraging a widespread use in different fields of application. The Unified Modeling Language (UML [20]) specified by the OMG, is an example of one of those languages and is used primarily for specifying and documenting software systems in a object-oriented way. Concurrently, DSLs provide a more accurate description of the domain in which they are applied, by using language elements that are closely connected to the constructs of the application domain. While UML be used to model any system in a generic way, a DSL by being closely related to the problem domain is able to capture more details and do it in a more expressive way [18]. Nonetheless, UML allows for the extension of its concepts, through the usage of a Profile. With a specific Profile, it is possible to specialize the UML, thus creating a DSL, maintaining the best features of both options: the wide usage of UML due to its many applications, and the ability to get more details of a DSL due to its

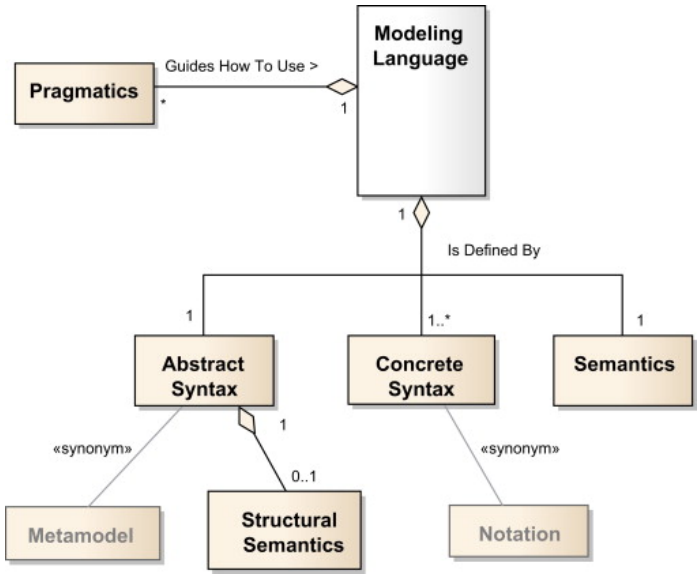


Figure 2.3: Modeling language definition (extracted from [3]).

close relation with the problem domain. Both UML Profiles and DSLs are further developed in Section 2.1.4 and Section 2.1.5, respectively [21].

The OMG proposes the Meta Object Facility (MOF [4]) as the standard in metamodeling. There many other modeling approaches, but the use of a standardized approach reduces the learning curve and fosters the adoption of the modeling language to be developed. Some examples of MOF-based languages are UML, XML Metadata Interchange (XMI) and MOF Model to Text Transformation Language (MOFM2T). MOF has two variants: (1) Essential MOF (EMOF) and Complete MOF (CMOF). EMOF is only a part of MOF, providing the constructs related with Object-Oriented (OO) programming languages and XML (e.g, EMOF is the metamodel of XMI). On the other hand, CMOF is the result of the merging CMOF with all its extensions. CMOF is the metamodel used to specify UML2.

MOF is designed as a pyramid diagram (see Figure 2.4), in a four-layered metamodel architecture, where each layer conforms to the one above it. The top layer, the M3 Level is the meta-metamodel layer and MOF is its own metamodel . The M2 layer represents the metamodel. The metamodel should be an instance of MOF (like UML). The M1 layer represents the model layer. In this layer, concepts made available in the metamodel are put to use, to define the user model. If UML is the metamodel in layer M2, then UML concepts should be used to define this model. Finally, M0 layer represents the system itself, that should be compliant with the model provided in M1.

Considering that MDA is the standard proposed by OMG, it relies on MOF to define its models (PIMs and PSMs). UML is a commonly used language to specify the PIMs. This guarantees that the models are compliant with the MOF architecture, and so can be store in a MOF-compliant repository, parsed and transformed by MOF-compliant tools, which is then rendered into XMI for network transportation [22].

In the XIS-Web language, we work with two separate variants of MOF, UML and Ecore. Ecore has been defined in the context of the Eclipse Modeling Framework (EMF) [11]. Ecore is similar to EMOF and is used as a metamodel to models and their runtime support in EMF.

2.1.4 UML Profile

As previously stated, a UML Profile is an extension of the UML language. Since UML does not allow the direct edition UML metamodel, this becomes an important mechanism for creating DSLs for a specific

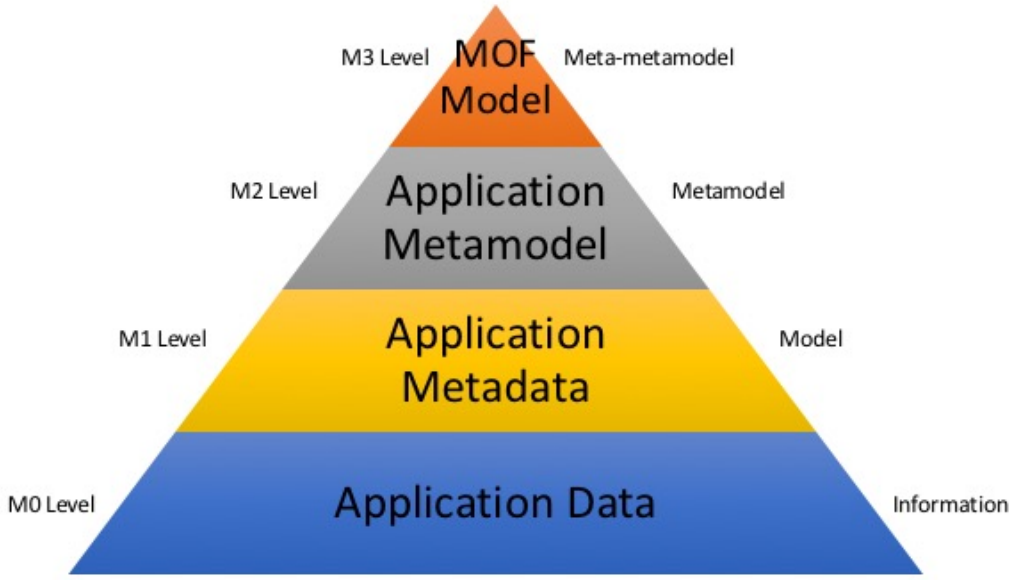


Figure 2.4: Adapted from OMG's MOF four-layered metamodel architecture [4].

domain, that are extensions of the generic concepts provided by UML. In order to properly define a UML Profile, one must use the following elements: (1) Stereotypes, (2) Tagged Values (or Tags) and (3) Constraints.

Stereotypes are metaclasses that extend UML metaclasses, i.e., they allow designers to extend the vocabulary of UML in order to create new model elements derived from existing ones. For example, a Stereotype can be an extension of an Interface, an Operation or a Package (metaclasses from UML). Besides refining the meaning of a concept (through its extension), a stereotype can also have a custom appearance.

Classes have attributes, meta-classes (or stereotypes) have tagged values (meta-attributes). They have the same purpose of Class attributes, that is to offer more expressiveness by adding more information to stereotypes.

A constraint is a restriction or rule, that defines the allowed behavior of stereotypes and relationships between them. Constraints defined in a UML Profile can not weaken the constraints present in the UML metamodel. The standard language for definition of constraints is the Object Constraint Language (OCL), proposed by OMG and can be used with any MOF meta-model.

Considering that UML is widely used, and that it allows for its extension via the UML profile mechanism, ultimately causes that a DSL implemented using this method has a lower learning curve (than a DSL implemented using a lesser known meta-model) and a high support from existing tools that already support the standard UML meta-model [20].

2.1.5 Domain Specific Language

A Domain Specific Language (DSL) is a language that, using specific notations and abstractions, represents concepts from a particular problem domain. A sub-set of DSLs, the Domain Specific Modeling Languages (DSMLs), like the one proposed in this dissertation, provide an even greater abstraction from the concepts in the domain, given that it offers a graphical representation for them.

A DSL is a powerful communication mechanism, because given that it represents domain concepts, it allows for an understanding of the system that is being built, by different types of stakeholders (like

developers, business experts, clients). Case studies show that DSLs can improve maintainability, productivity, reliability and portability [21]. Given that DSLs are specific to a problem domain, the learning curve for it is typically higher, when comparing to a GPML like UML. Some popular DSLs are YACC [23] grammars for parser creation and HTML for web page markup.

2.1.6 Model Transformations

A model transformation is a process that receives one or more models as input and then produces output based on information from those models. MDD recognizes two types of model transformation, namely Model-to-Model (M2M) and Model-to-Text (M2T), that convert models into other models, or into text (in this case source code) respectively [24]. Both of these transformations play a key part in the XIS-Web technology. A more detailed explanation is given in Sections 5.2 and 5.3.

2.1.6.1 Model-to-Model Transformations

When the output of the model transformation is another model, then it is a M2M transformation. One or more models can be provided as input and one or more models can be generated as output (e.g., the generation of a relational model for a database starting from a standard class diagram). In this work, we propose an MDD framework that performs M2M transformation that have multiple diagrams as input and generates multiple diagrams as output (many-to-many transformations). ATL and QVT (two popular M2M transformation languages) are described next.

ATL (ATLAS Transformation Language) [25] is a M2M transformation language that is developed and maintained by AtlanMod and Obeo. ATL is a rule-base language and makes use of OCL to provide M2M transformations. In this language, the transformation has a textual representation and are unidirectional (i.e., ATL transformations only allow reading from the source model and writing on the target model) [9].

QVT (Query-View-Transformation) [26] is the standard language for M2M transformations proposed by the OMG. It comprises three languages that have MOF as meta-model, namely: (1) QVT-Relations; (2) QVT-Operational; and (3) QVT-Core. The first provides both a graphical and textual syntax to describe the correspondences between the source and target metamodels. These correspondences are made in a declarative manner and be unidirectional or bidirectional. The second language, QVT-Operational, is represented using a textual concrete syntax that allows for the specification of unidirectional model transformation, but in an imperative manner. Finally, QVT-Core is a declarative language, that is not intended to describe M2M transformations, instead its purpose is to be the target of the compiler of the QVT-Relations language [9].

2.1.6.2 Model-to-Text Transformations

On the other hand, like previously mentioned, a M2T transformation is a model transformation that produces text as output. Typically, the text generated by the M2T transformations is source code (considering the purpose of MDD), and so M2T transformation can also be known by Model-to-Code (M2C) transformations. These transformations adopt one of two approaches, either (1) template-based approaches or (2) visitor-based approaches [24].

Template-based approaches are the most commonly used approaches. These approaches require the definition of code templates, that contain most of the part of the code that is to be generated (static part), and a part with specific annotations (the dynamic part), which in runtime is replaced by data from the source model. Typically, the structure of a template is related to the source code it generates.

Comparatively, visitor-based approaches are a based on the Visitor design pattern [27], and are used to traverse the internal representation of the model and write code to a text stream. The Jamda

framework is an example of an application of this approach. It provides a set of classes to represent UML models, an API for manipulating models and a visitor mechanism [28].

2.1.7 Language Workbench

In the recent past, we have witnessed the development of tools that support both the MDD principles and DSLs. The first tools of this type, that started appearing around the 1980s and 1990s, were the Computer-Aided Software Engineering (CASE) tools. CASE tools were focused on providing developers the tools and methods to express software systems, using GPML representations. CASE tools failed to be widely adopted and had little impact on the software development department, mainly due to: (1) the poor translation of general-purpose language concepts and representations onto the underlying platforms, creating a huge overhead in the code created, making it harder to understand and maintain; (2) the lack of scalability when handling complex systems, considering the tools did not offer support for concurrent engineering, lacked integration of the generated code with other platforms, and the graphical representations were too generic and failed to be applied on several domains [12][29]. Nonetheless, with the evolution of the programming languages, and the lessons learned from the CASE tools, better MDD and DSL supporting tools started to emerge.

Currently, language workbenches are the common solution when developing DSLs, mostly because they have worked around the inflexibility of the CASE tools. We can describe a language workbench as a tool that supports the definition, reuse and composition of DSLs, as well as the creation of customized Integrated Development Environments (IDEs) [30]. In conclusion, language workbenches allow for the creation of DSLs and the custom IDEs that they use. They are also known as meta-CASE tools [31][32].

2.2 Web Application Development

This section describes some concepts important to understand the area of Web Application Development. In Section 2.2.1 we provide definitions for Web Applications with particular emphasis on Responsive Web Applications. Section 2.2.2 describes some of the most popular User Interface Patters for web applications. Section 2.3 surveys technologies related with this research work.

2.2.1 Web Applications

A Web application, or ”web app”, is a software program that runs on a web server. Unlike traditional desktop or native mobile applications, which are launched by their respective operative system, web apps are accessed using the web browser [33].

Typically, web applications are divided in layers. Although they can have as many layers as needed, they usually are divided in three layers: (1) Content (or Data), (2) Application (or Hypertext) and (3) Presentation [34]. Figure 2.5 depicts the layers of a web application generated by XIS-Web. In some cases, like XIS-Web, all three layers reside inside the web browser. This is particularly important if the application is required to run on any device, thus avoiding the installation of dependencies.

Comparing web apps with desktop applications we can enumerate some advantages like: (1) the application only needs to be developed once (the browser supplies the flexibility for running on any platform)- as opposed to desktop applications that need to be developed for each platform; (2) doesn't require installation and has no startup time. They also have some disadvantages when compared to desktop apps like: (1) constraints in term of hardware components (like CPU, memory, file system); (2) much worse offline support - typically desktop apps are capable of running offline (a difficult task for web applications) [35].

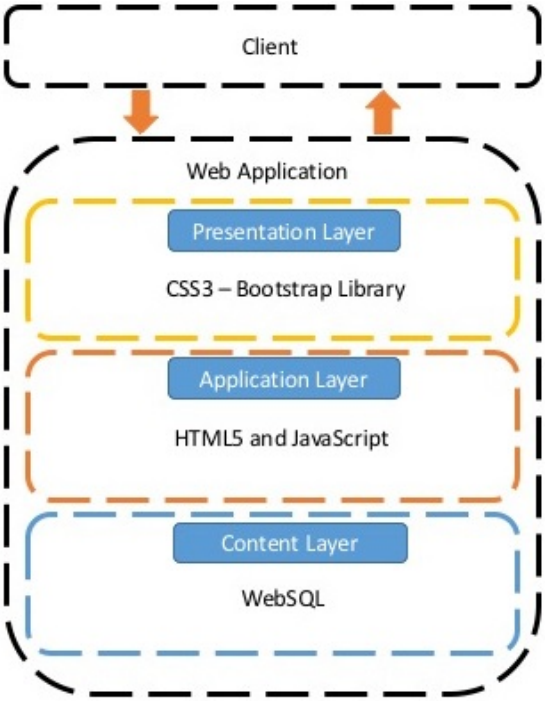


Figure 2.5: Three-layered architecture of applications generated by XIS-Web.

Also, regarding native mobile apps, the following comparisons can also be made: (1) the web app only needs to be developed once to run on any mobile platform; (2) there is no need for an application store to make the application available and (3) web apps can have the same look-and-feel as native mobile apps. On the other hand, native mobile apps can take better advantage of the platform on which they are running, making them more performant, when compared to web apps, and the access to internal components (like hardware or service providers) is more facilitated in native mobile applications than in web apps [36].

2.2.1.1 Responsive Web Design

Responsive Web Design RWD is an approach to web designed aimed at crafting sites that provide an optimal viewing and interaction experience. By using elements that can be rescaled or reshaped according to the device on which they are being displayed, thus allowing for easier reading and navigation with a minimum of resizing, panning and scrolling [37].

In practice, RWD is achieved by (1) using fluid, proportion-based grids that enables page element sizing to be in relative units like percentages, rather than absolute units like pixels or points; (2) flexible images that can be scaled and are also sized in relative units, thus preventing the image from displaying outside their containing element and (3) CSS3 media queries that enable the page to use different CSS style rules, based on the characteristics of the device the site is being displayed on, focusing mostly on the width of the device [38].

In 2014, the number of mobile devices surpassed the number of desktop devices[1]. This makes Responsive Web Design a must have for web applications, to a point where Google has begun to boost the ratings of sites that are mobile friendly (responsive) if the search was made from a mobile device [2].

[1] http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/
[2] https://developers.google.com/webmasters/mobile-sites/

2.2.2 User Interface Patterns

Following the explanation on Responsive Web Design, the User Interface takes a major role in web applications. A user is more likely to use an application if he likes its usability and if he finds it attractive. So, User Interface (UI) design is a crucial task during the development process of a web application.

Given that designing user interfaces is a complicated matter, designers experts have defined UI patterns that can be applied by developers, to achieve the desired usability for their applications [39].

Additionally, patterns in this research work must abide to the Responsive Web Design principles that were mentioned earlier. This section describes some of the patterns for applications that were taken into consideration for this research work.

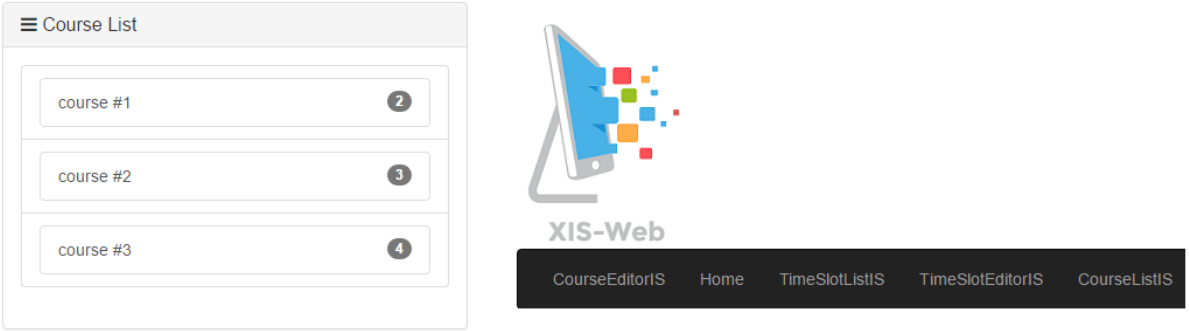


Figure 2.6: Example of a List and a Tab Menu in XIS-Web (from left to right).

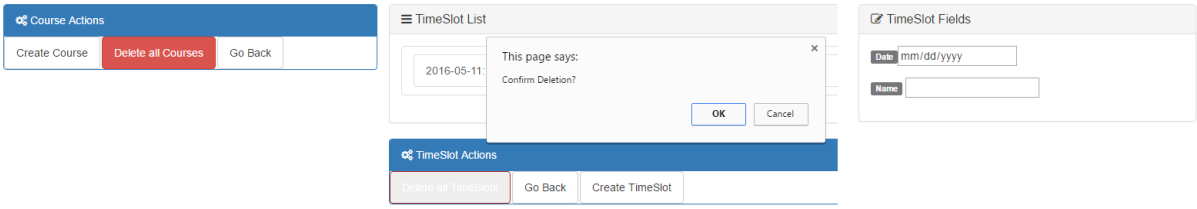


Figure 2.7: Example of an Options Menu, a Dialog and a Form in XIS-Web (from left to right).

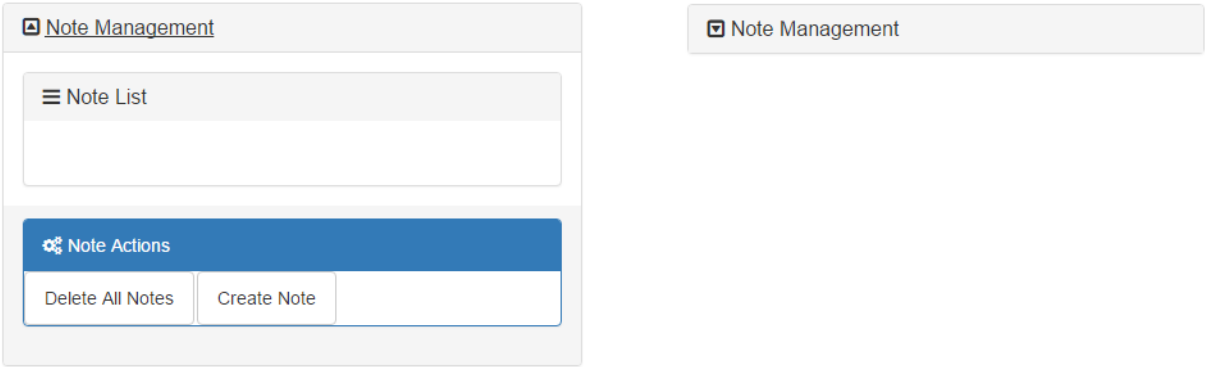


Figure 2.8: Example of a Collapsible Panel open and a Collapsible Panel closed in XIS-Web (from left to right).

List. The List pattern is one of the most common patterns in web applications. It represents navigation options in a vertical list. Typically this is the pattern to be used when the set of options is very large, or when there is the need to enumerate instances of an entity. [40][41] Some variations of this pattern

present a header that helps clarifying the purpose of the list, while some have searching, ordering or filtering of the list. Figure 2.6 shows an example of a List pattern in a XIS-Web interaction space.

Tab Menu. This type of menu consists in a group of navigation options that is either placed side-by-side at the top or bottom of a screen (for non-mobile devices), or vertically on the sides (for mobile devices). Tab Menus are often used for navigation and are most suitable when the number of navigation options is small [40]. Figure 2.6 shows an example of a Tab Menu in XIS-Web.

Options Menu. Typically, actions that are bound to a certain page can be represented via the Options Menu. These actions can either be navigational, triggering the navigation flow from one screen to another, or non-navigational, changing only the current state of the page. Keeping in mind the difference in devices and input methods available, accessing an Options Menu can be done in different ways: by activating an hardware button or by clicking/tapping a menu button widget [40][41]. An example of this pattern in XIS-Web is depicted in Figure 2.7.

Context Menu. Context Menus, like Options Menu provide a set of options, that in this case are bound to a certain element of the screen (its context). In XIS-Web, we use this type of Menu associate with lists, as it is recommended.

Collapsible Panel. A Collapsible Panel is any panel that can be collapsed or restored. This pattern is typically used to reduce ”visual pollution” in web pages, by hiding certain parts of it. In XIS-Web we use this pattern for nested pages (see Figure 2.8) [40].

Form. A form is a container widget for other widgets with the purpose of data entry. In XIS-Web we use forms to edit the attributes of entities from the domain (see Figure 2.7).

Dialog. A dialog (also known as pop-up) is typically a window for user interaction or acknowledgement [40]. In XIS-Web dialogs are used to confirm Delete actions, as shown in Figure 2.7.

Map and Location. This pattern is suitable for when the user wants to search for a given location. It shows a map containing markers with points of interest. By click on each marker it is possible to access the details of the location.

Master-Detail. This pattern is a composite pattern. Typically, it shows multiple instances of the Master entity (for example using the List pattern), and the Detail for that instance (with its attributes) is showed in a separate page [42].

Dashboard A Dashboard is a data visualization UI pattern, that should be used when the user wants to digest data from multiple sources [43]. Figure 2.9 shows an example of a Dashboard.

Gallery A Gallery is a UI pattern focused on the visualization of media content. It is employed when the user needs to browse a collection of images or videos. Figure 2.9 shows an example of a Gallery.

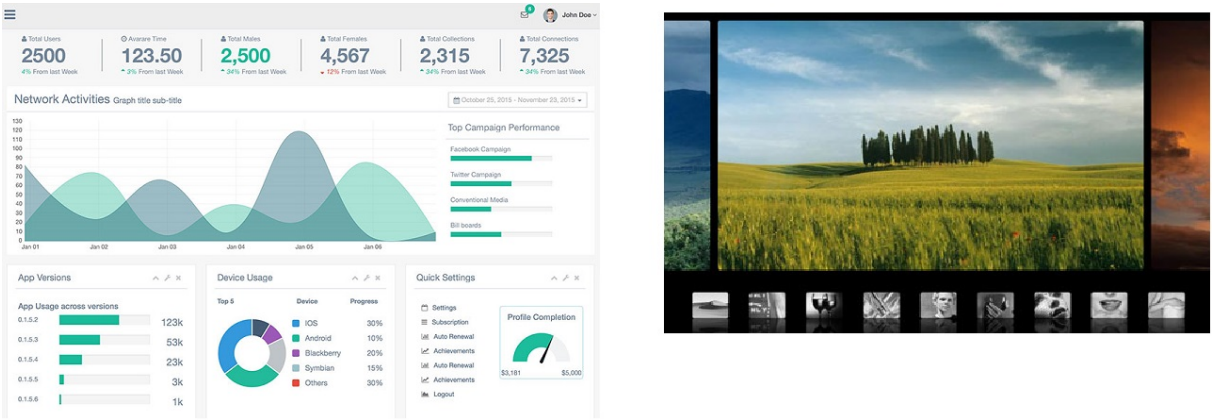


Figure 2.9: Example of applications of the Dashboard and Gallery UI patterns (from left to right).

2.3 Related Work

This section presents a survey of work related to this research work, namely technologies that describe modeling languages with MDD-based frameworks. The following technologies provide a modeling language to build web applications and the supporting framework or tool to automatically generate a web application. Considering the division of the web application, provided in section 2.2.1, where the Content Layer corresponds to the persistency of domain entities in the application, the Hypertext (or Application) layer where the behavior and logic of the system is defined and the Presentation Layer in which the User Interface layout is described and navigation flow defined. These frameworks either provide a built-in mechanism for automatic code generation or are supported by a tool that does it [44].

Hera. This framework, based on the Relationship Management Methodology (RMM), was first introduced in 2000 and proposes models for all three of the web applications layers [45]. To model the Content, it uses plain Resource Description Framework (Schema) (RDF(S)) and its domain models are based on ”concepts”, ”attributes”, ”concept relationships” and ”media types”. Hypertext and Presentation level modeling use much of the RMM’s graphical notation, making them being slices and slice relationships that can either be compositional or navigational. Hera is supported by the Hera Presentation Generator (HPG), which is a proprietary tool available as freeware. This tool guides the user through the design and generation process, and in each step the models can be viewed using a text editor or Microsoft Visio. HPG also performs model validations in order to verify its consistency. From these models it is possible (through Model-to-Text transformations) to create a suitable static presentation in languages like HTML and WML[46][44].

UML-based Web Engineering (UWE). UWE is a language that was first introduced in 1998, and has been evolving since. It proposes three distinct viewpoints: Content View, Navigation Space View and Presentation View. First, the Content View models are based on standard UML Class diagrams. In the Navigation View, State chart diagrams are used to model navigation scenarios and sequence diagrams can be used to depict the application’s flow in the Presentation View. This language is supported by ArgoUWE (which is free). During modeling time, ArgoUWE checks the artifacts for errors and carries out semi-automatic transformations to generate first the navigation model (from the content model) and second, the presentation model from the navigation model. ArgoUWE enforces M2T transformations rule on the models specified using UWE, through the usage of rules written in ATL. The end result of these transformations are Java Beans and JSP [47].

Object-Oriented Hypermedia method (OO-H). The OO-H method first appeared in 2000 [48]. The language is defined as a UML profile and is specifically designed for web application development. Like UWE, standard UML Class diagrams are used to model the Content View. For the Navigation View, Navigation Access Diagrams (NAD) capture navigation paths and services that the users can activate. In the Presentation View, having a NAD as input model, an Abstract Presentation Diagram (APD) can be automatically generated as output, using M2M transformations, that in this cased are called NAD2APD mapping rules. The APD can be interpreted as a sitemap of the web application, because it consists of a set of pages and their links. Finally, the APD can be transformed via M2T transformations in order to generate the web application artifact, in PHP. OO-H has a support framework in VisualWade, a commercial tool, that helps with the modeling in general (edition and validation) and applies the mentioned transformations on the models.

OutSystems Platform[3]**.** OutSystems provides a rapid application development (RAD) framework that enforces an MDD-based approach. This platform is for commercial use (with trial licenses available) and stands out for its very large user base. Typically, the development process using this platform is done using Platform-Independent Models (PIMs) for the Logic, User Interface, Data and Business Process models. Besides generating web applications in HTML5 and CSS3, the OutSystems platform also generates code for the Spring and ASP.NET MVCs. This is also a very known framework for generating native mobile applications.

Tumult Hype[4]**.** Hype 3.5 is the most recent version of Tumult's Hype Platform. It is a proprietary software (with trial licenses) that takes a different approach at solving the generation of web applications problem (when comparing to the other surveyed frameworks). The main difference of this platform, is that the modeling is focused exclusively on the User Interface. The user populates a screen with the elements that take part in it (widgets), applies the logic using either the Actions, Timeline or Properties views (that can either be animation of the widget, or flow control) and Hype generates the correspondent code. While the other surveyed technologies were intended to generate a "form heavy" web application, this framework is more focused on infographics, presentations and web advertisement.

The discussion of the presented technologies and comparison with XIS-Web will be carried out in Section 6.3.

[3]https://www.outsystems.com/platform
[4]http://tumult.com/hype

Chapter 3

XIS-Web Approach

In this chapter, it is presented an overview of the approach that is proposed by XIS-Web. This technology relies in two major components: the XIS-Web language, a DSL for building responsive web applications that is a UML profile and its companion framework, that supports the MDD of applications that are built using the XIS-Web language. Section 3.2 presents an overview of the package organization of the XIS-Mobile language and Section 3.3 explains the approaches that are available when building an application using the XIS-Web language. Finally, Section 3.4 describes three case study applications that were created during the research work.

3.1 Background

XIS-Web is based on previous work, namely it reuses and adapts some concepts of the XIS and XIS-Mobile approaches [10][1].

XIS proposes a MDD approach for designing web or desktop interacting systems at a PIM level, using a DSL defined as a UML profile, and from these models automatically generate source code.

The XIS UML profile is organized in three main sets of views: Entities, Use-Cases and User-Interfaces. First, the Entity view comprises the Domain and BusinessEntities views. The Domain view represents the relevant classes to the problem domain, their attributes and the relationships between them. While BusinessEntities view defines high-level entities, named business entities, used to aggregate entities of the Domain view or other business entities that can be more easily manipulated in the context of a given use case.

Second, the Use-Cases view contains the Actors and the UseCases views. The Actors view defines the entities that interact with the system under study. The UseCases view specifies the operations that the actors can perform over the business entities, when interacting with the system.

Third, the User-Interfaces view comprises the NavigationSpace and InteractionSpace views. The first represents the navigation flow between the various screens of the system, named interaction spaces, with which the actors interact. In turn, the InteractionSpace view details the content of each interaction space, i.e., the elements of the graphical user interface, and can also specify the access control of the actors to those elements [10].

XIS also proposes two modeling approaches: the smart approach and the dummy approach. Figure 3.1 describes the smart approach (for XIS-Web) and in Section 3.3 both approaches are described.

More recently, the XIS-Mobile approach was defined with the focus on developing cross-platform mobile applications. XIS-Mobile uses a DSL that reuses some of the best concepts proposed on XIS, namely its multi-view organization and modeling approaches. XIS-Mobile introduces new concepts (e.g.,

new types of widgets, internet connection, localization and gesture support) in order to be more appropriate to design mobile application scenarios. The XIS-Mobile language is organized in six views: Domain, BusinessEntities, UseCases, InteractionSpace, NavigationSpace and Architectural. While the first five views share the same goals as in XIS (with some adjustments and different stereotypes), the latter is totally new and represents the interactions between the mobile application and external entities (e.g., web servers or providers). XIS-Mobile is supported by a framework that allows designing and validating the models described in the XIS-Mobile language, generating other models from them (through M2M transformations) and in the end generating native source code for multiple mobile platforms (Android, iOS and Windows Phone), through M2T transformations [1][49].

3.2 Overview

XIS-Web proposes a DSL as a UML profile, with the purpose of designing responsive web applications using an MDD approach. XIS-Web also proposes a companion framework that supports the language by allowing the automatic creation of the application that was designed. Apart from allowing the edition of the models, the framework also gives the possibility to automatically generate the more complex views, using M2M transformations. Once all the views are completed M2T transformations generate responsive web code (HTML5, JavaScript and CSS3 via the Bootstrap library).

Figure 3.1 shows the different views inside XIS-Web, reused from XIS-Mobile. There are four major packages: (1) Entities view, (2) Architectural view, (3) UseCases view and (4) User-Interface views.
The Entities view package is composed by the Domain and BusinessEntities views. The Domain view typically is the entry point of the framework and is the place where the developer defines the problem domain. The BusinessEntities view is dependent on the first one, since it aggregates entities from the Domain view into higher-level entities (called business entities), that are needed to supply context to other diagrams. The Architectural view can be considered a ”Distributed Systems” view, because it's where the interactions of the applications with external entities are described. The UseCases view relies on the BusinessEntities view for context, by associating its use cases to a business entity. It can also depend on the Architectural view if there is a specific use case that uses an external service. Finally, in the User-Interface views, we have the InteractionSpace view and the NavigationSpace view that depend on each other. The InteractionSpace view, in order to have context for the entities that it controls, is associated with a business entity and if there is an external action provided by an external service it also requires association to the entity from the Architectural view. On the other hand, since the NavigationSpace view portrays the navigation flow between interaction spaces, it only depends on the InteractionSpace view. Each of these views is explained in detail in Chapter 4.

The division of XIS-Web in separate packages promotes modularization and separation of concerns by having each view handling a specific problem of developing web applications. Another example of modularization is the aggregation of several domain entities into a business entity and the ability to work with this to provide context to other views. The fact that XIS-Web takes a Use-Case driven approach enforces the user to specify some use cases for the web application that allow XIS-Web to capture web application UI patterns that are key for the M2M transformation step. The M2M transformation reduce the overall complexity of building a web application and the time spent doing it, thus increasing productivity.

3.3 Design Approaches

Considering the design approaches proposed in XIS and later refined by XIS-Mobile (Section 3.1), two design approaches at building web applications can be taken: the dummy approach or the smart ap-

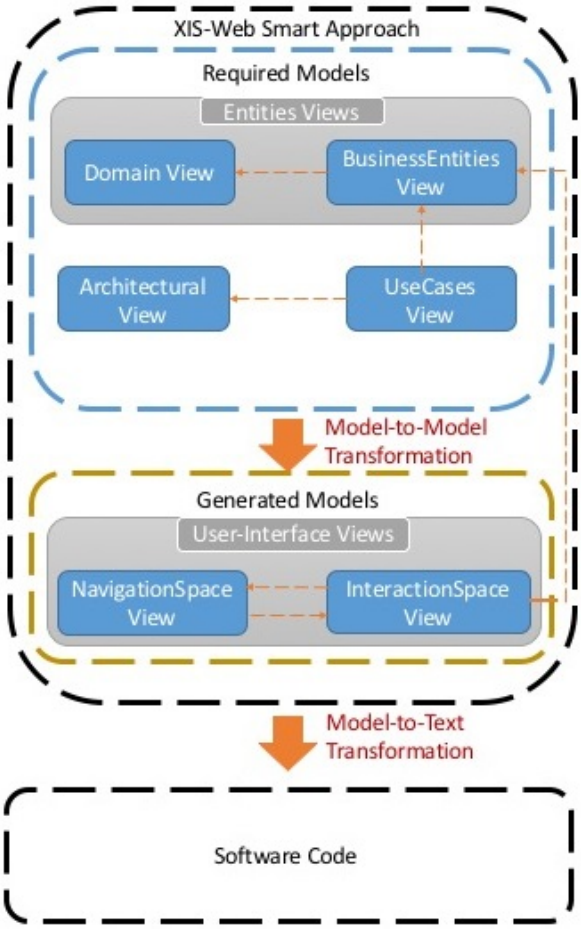


Figure 3.1: Interaction between XIS-Web views (smart approach) (adapted from [1]).

proach. The key difference between these approaches is that the smart one takes advantage of the XIS-Web framework's M2M transformations to automatically create the User-Interface views from the other diagrams (explained in detail in Section 5.2). This is particularly important given that the InteractionSpace view is one of the most complex diagrams, making it very cumbersome and time-consuming.

Adopting the dummy approach, the developer should define all the views mannualy, with the exception of the Architectural view that is only required to model interactions of the application with external entities, thus being optional. As opposed to the smart approach (described in Figure 3.1), where the developer is only required to define a minimum of three views: the Domain, BusinessEntities and UseCases view (due to the optional characteristic of the Architectural view that was explained earlier). From this stage, using the information provided Domain and BusinessEntities view and the patterns represented by each use case (explained in Section 5.2) it is possible to automatically generate the User-Interface views through M2M transformations. Even though the smart approach generates the last two views automatically, the developer can customized them as he wishes.
It is recommended to use the smart approach whenever possible, because it reduces drastically the complexity of modelling a web application, while speeding up the process. After all the views are defined, using either the dumb or smart approach, a software artifact can be generated using M2T transformations (Section 5.3).

3.4 Case Studies

Evaluating the XIS-Web requires assessments to both its usefulness and to its usability. In order to evaluate its usefulness three cases studies were defined.

Case Study A - The TimeSlot Booking App

The ”TimeSlot Booking App” is an application that allows for the management of ”TimeSlots” that are booked by ”Students”. Each ”TimeSlot” has a start date and time, a duration and a name. ”TimeSlots” also contain ”Topics” and ”Topics” can be present on many ”TimeSlots”. A ”TimeSlot” is also associated to a ”Course”. The ”Student” should be able to:

- Manage his current ”TimeSlots”:
  - Create new ”TimeSlots”, view or edit existing ones or remove a ”TimeSlot” from the list. (Creating or editing a ”TimeSlot” requires the definition of all of its attributes).
- Manage his ”Courses”:
  - Create new ”Courses”, view or edit existing ones or remove a ”Course” from the list.
- Share his ”TimeSlots” using an external service.

The information about the domain entities should be persistent.

Case Study B - dDocs App

The ”dDocs” App lets citizens store, manage and share their own personal documents (like ID Card, Driver’s License, Health Insurance, etc.). Every document has a state (expired, expiring, signed and unsigned), and in this system documents originate from a document template. Each document template is created and maintained by the entity that issues it, namely a user with access to the platform with a more administrative role - the curator. The application should be able to communicate with an external service in order to publish and backup documents in a third party repository (e.g., Google Docs). Citizens should have a gallery view of their documents, and a Dashboard with useful information regarding those documents, like: (1) pie chart with document state, (2) bar chart with number of documents by issuer and (3) number of documents for the last three years in a bar chart.
This case study is a simplification of part of a system that is currently being developed for a real world case[a].

[a] http://ddocs.eu

Case Study C - To-Do List App

The To-Do List app is an application that lets the user manage his tasks. Each task contains a title, a description and date of completion. Notes can also be associated to a task and a task belongs to a certain category. Notes can be defined as title and content and Categories have only name. The user should be able to view all his tasks, to create new tasks, view or edit existing tasks and remove a task from the list.

Case study A is used as a running example to explain both the XIS-Web language in Chapter 4 and Chapter 5. This case study was also used as assignment of an evaluation pilot-user test session with people not involved in this research work. More information about this session is given in Section 6.2.

Concerning case studies B and C, these were both implemented manually and then generated via the platform. Case study B is specially important since it is a simplification of part of a real world application, while on the other hand, case study C is a well-known case study within the research group. More details about the evaluation of both case studies B and C is given in Section 6.1.

Chapter 4

XIS-Web Language

The XIS-Web language is a DSL specifically designed to model responsive web applications in a platform independent way. Following the approach taken by the XIS-Mobile, reusing some concepts and redefining others, we divide the language in four main views: (1) Entities View; (2) Architectural View; (3) UseCases View; and (4) User-Interfaces View. In the Entities View we find the Domain and Business Entities view, that have similar concerns. In turn, the User-Interfaces View comprises the NavigationSpace and InteractionSpace views. Each of the aforementioned views is described in detail and used via the case study A - The TimeSlot Booking App - described in Section 3.4. The choice of the case study was made due to its simplicity and practical use, thus contributing for a better understanding and ease of explanation.
In this chapter, Section 4.1 details the Entities View package and its elements, the Domain and BusinessEntities views. Section 4.2 describes the Architectural View, while in Section 4.3 the UseCases View is explained. Finally, Section 4.4 provides a description for the User-Interfaces View, detailing the NavigationSpace view and the InteractionSpace view. The full specification of the XIS-Web language is given online[1].

4.1 Entities View

The two views comprised in the Entities View package are the first ones to be defined as soon as the requirements of the web application that is being developed are clearly specified. In here, the concepts relevant to the domain are defined or identified. This package contains the Domain and BusinessEntities views that are described below.

4.1.1 Domain View

The Domain View allows for the representation in a Class Diagram of the entities that are part of the problem domain. Entities can have one or more attributes and can be connected using associations, aggregations or via inheritances. It is also possible in this view, to define enumerations that can be used as entity attribute types.
To model the entities that take part in the domain, the XIS-Web language allows the following stereotype application in this view: (1) A Class takes the form of a XisEntity; (2) An Attribute is modeled as a XisEntityAttribute; (3) Associations and Aggregations are represented using XisEntityAssociation; (4) Inheritances via XisEntityInheritance; (5) Enumerations are represented as XisEnumeration; and (6)

[1] https://github.com/MDDLingo/xis-web

Enumeration literals or values are XisEnumerationValue. The metamodel of this view is show in Figure 4.1.

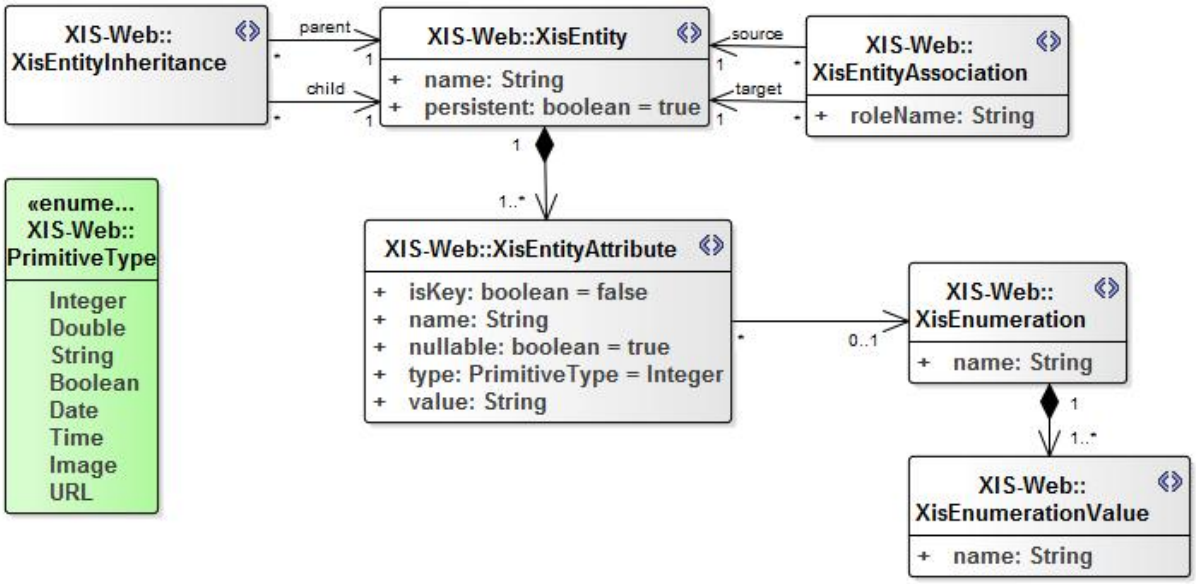


Figure 4.1: Metamodel of the Domain View

In Figure 4.2 we have the TimeSlot Booking App's Domain View. The domain of this case study comprises three XisEntities: TimeSlot, Topic and Course. A TimeSlot contains three XisEntityAttributes, the slot's name, its date and its duration. Moreover, while Topic has attributes for title and content, Course only has a name.

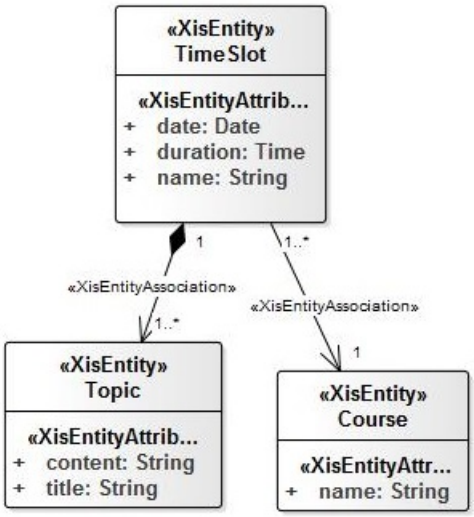


Figure 4.2: Domain View of the TimeSlot Booking App

4.1.2 Business Entities View

The BusinessEntities View represents high-level entities, using a Class Diagram. These are known as business entities and are modeled using the stereotype XisBusinessEntity. Business entities aggregate one or more XisEntities (from the Domain View). The main goal of a business entity is to provide a direct context to use cases and interaction spaces by defining the scope of domain entities that are directly

connected to them and can be manipulated by them. Taking this in consideration, the BusinessEntites View plays a key role during both transformation stages.
We define a business entity by specifying (through an association) a domain entity (from the Domain View) as its master entity. Considering master entity relationships (aggregations and associations) in the Domain View, other domain entities can be aggregated to a business entity, acting as detail and reference entities to the business entity in question.

The stereotypes for this view are: (1) XisBusinessEntity for classes; (2) XisBE-EntityMasterAssociation to describe the master association between the business entity and the corresponding master entity from the Domain View; (3) XisBE-EntityDetailAssociation that as a role similar to the previous item but for a detail entity and (4) XisBE-EntityReferenceAssociation. These associations also contain a tagged value ”filter” that allows the restriction of the domain entity’s attribute that can be used in the context of the XisBusinessEntity. Figure 4.3 shows the metamodel.

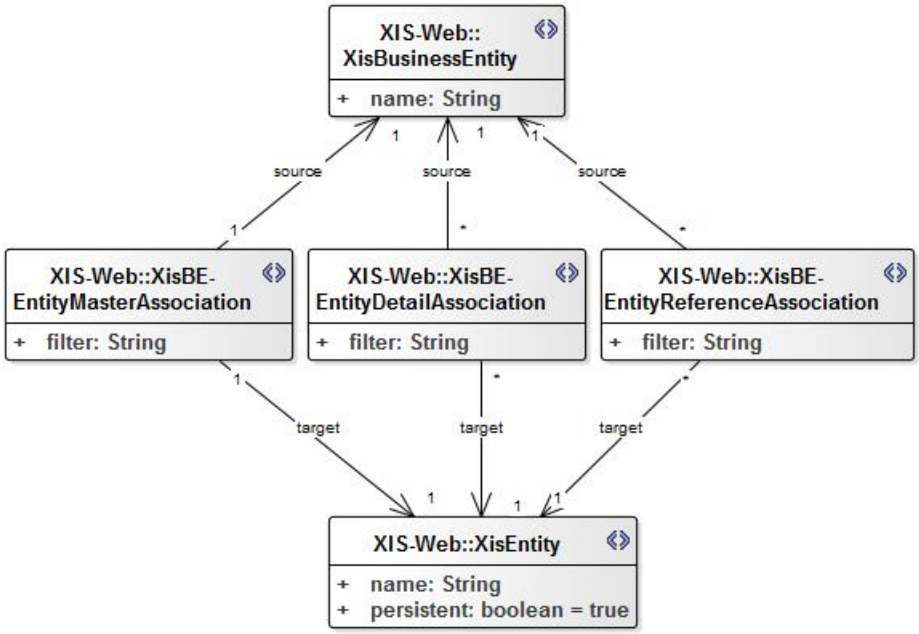


Figure 4.3: Metamodel of the BusinessEntities View

Considering the TimeSlot Booking App we find two business entities, the TimeSlotBE, having TimeSlot as its master entity, Topic as its detail entity (all from the Domain View) and the CourseBE with a master entity association to the Course (from the Domain View). One possible representation is shown in Figure 4.4.

4.2 Architectural View

The Architectural View, depicts the interactions of the web applications with external entities, making it also a ”Distributed Systems View”. Like the previous views it is also modeled using a Class Diagram. The web application is represented as a XisWebApp and can be connected to several XisServices, that are interfaces.
The main goal of the XisService is to describe the operations that are performed by its implementation using XisServiceMethods. The XisProvider stereotype is the realization of the XisService. XisServices are divided in two types: (1) XisInternalService if it is provided by an entity inside the device; and (2)

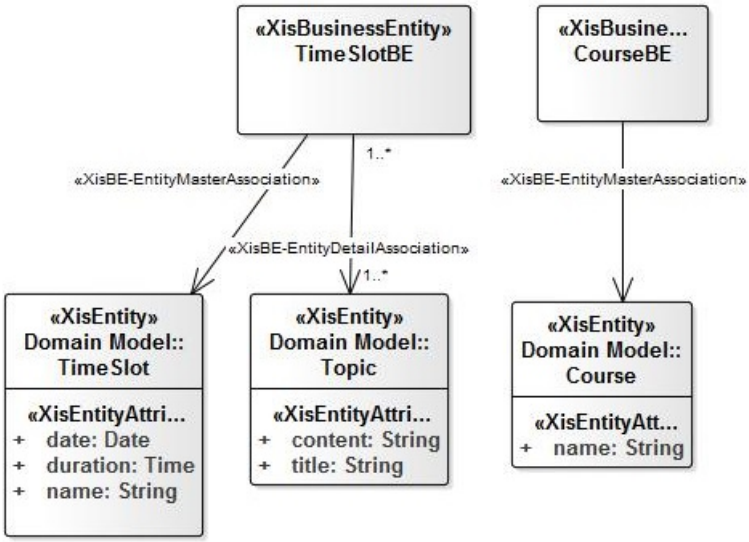


Figure 4.4: BusinessEntities View of the TimeSlot Booking App

XisRemoteService, if it is provided by an external entity. A XisRemoteService can only be realized by a XisServer, that is a direct representation of a web server, or a XisWebClientApp, that represents the web application running on a different device. On the other hand, each XisInternalService can only be realized by a XisInternalProvider. In order to better define its scope, a XisInternalProvider can have one of the following types: (1) Location, that represents the location provider and grants access to the geographical location; (2) Contacts, that allows access to the contacts of the mobile device; (3) Calendar, that enables the interaction with the device's calendar provider; (4) Media that allows the interaction with the media hardware of the device (camera, microphone, media player, etc.); and (5) Custom, that allows for the developer to create its own provider.

The stereotypes applicable to this view are: (1) XisWebApp, (2) XisInternalProvider, (3) XisClientWebApp and (4) XisServer for classes; (5) XisInternalService and (6) XisRemoteService are interfaces; (7) XisServiceMethod for operations; (8) XisWebApp-ServiceAssociation allows for the associations between XisWebApps and XisServices; and (9) XisProvider-ServiceRealization to map the relationships between XisProviders and XisServices.
The correspondent metamodel is shown in Figure 4.5.

Concerning the Architectural View of the TimeSlot Booking App we can see the interaction of the external service with the web application. There is a XisRemoteService that should implement the XisServiceMethod shareTimeSlot() that is provided by the XisServer. An example is shown in Figure 4.6.

4.3 UseCases View

The UseCases View is a representation of the operations that can be performed by an actor when interacting with the web application. These should be described in the context of a provider (from the Architectural View) and/or a business entity (from the BusinessEntities View). Unlike the other views, this one is modeled via a Use Case Diagram. The UseCases View can have two stereotypes: (1) XisEntityUseCase and (2) XisServiceUseCase.

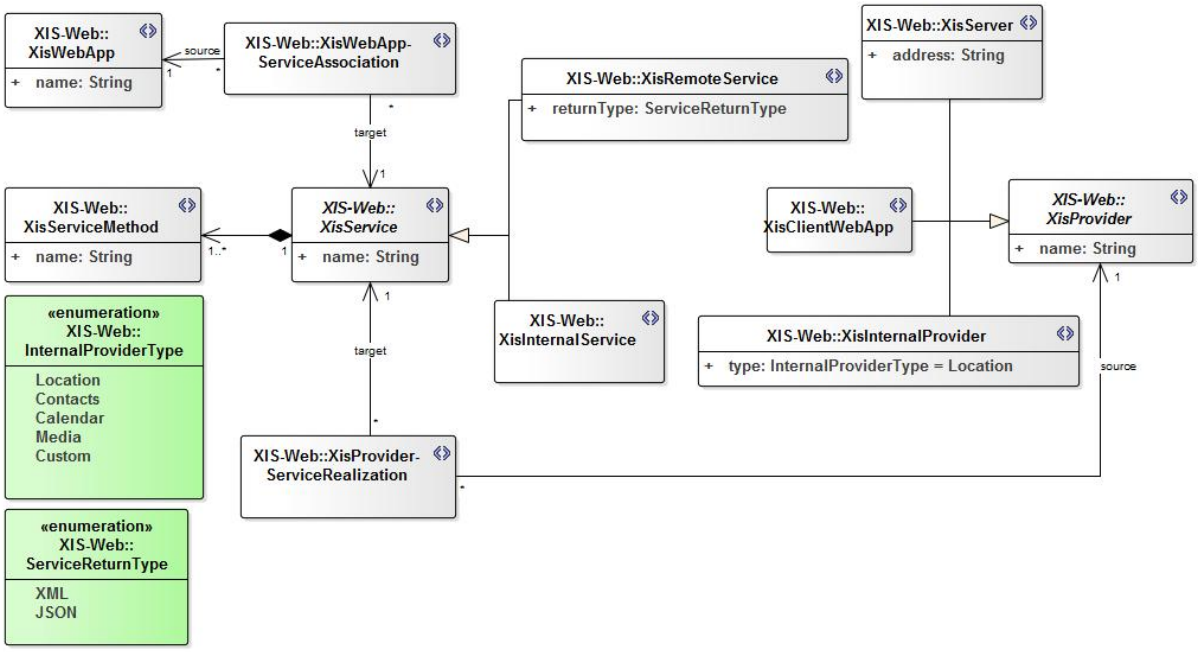


Figure 4.5: Metamodel of the Architectural View

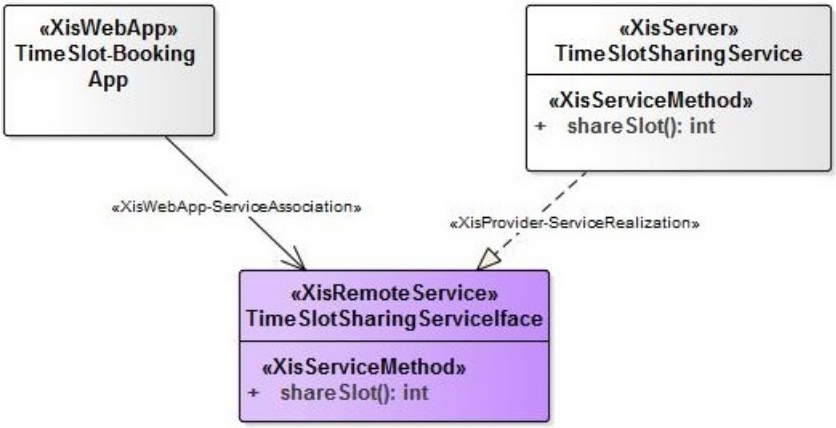


Figure 4.6: Architectural View of the TimeSlot Booking App

A XisServiceUseCase represents an action that is described by a XisService and realized by a XisProvider. Typically it is connected to one of the XisProviders that was described earlier but it can also be connected to a business entity, stating that the operations that it does have an effect on the business entity's domain entities. Typically this type of use case is used as an extension of the XisEntityUseCase.

A XisEntityUseCase representes an action over a business entity and the domain entities aggregated to it. It has a collection of tagged values that enable the user to specify the CRUD (Create, Read, Update, Delete) operations for the master, detail and reference entities. These were the chosen set of operations because they are the most commonly used, as it was observed in [50]. Another important tagged value is the type. A XisEntityUseCase can be: (1) EntityManagement or (2) EntityConfiguration. The type of the XisEntityUseCase has a direct influence on the type of interaction space. If the type is EntityManagement the generated interaction space will allow for the management of a list with multiple instances of the master entity. On the other hand, if the type is EntityConfiguration the generated interaction space will be configured to manage a single entity of the associated entity. A more detailed explanation of the interaction spaces and their types is given in Section 4.4.2.

The XIS-Web 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, to portray the associations between actors and use cases; (5) XisEntityUC-BEAssociation to associate XisEntityUseCases and XisBusinessEntities; (6) XisServiceUC-BEAssociation for associations between XisServiceUseCases and XisBusinessEntities; and (7) XisService-UCProviderAssociation allows the association of XisServiceUseCases with XisProviders. The metamodel of this view is shown in Figure 4.7.

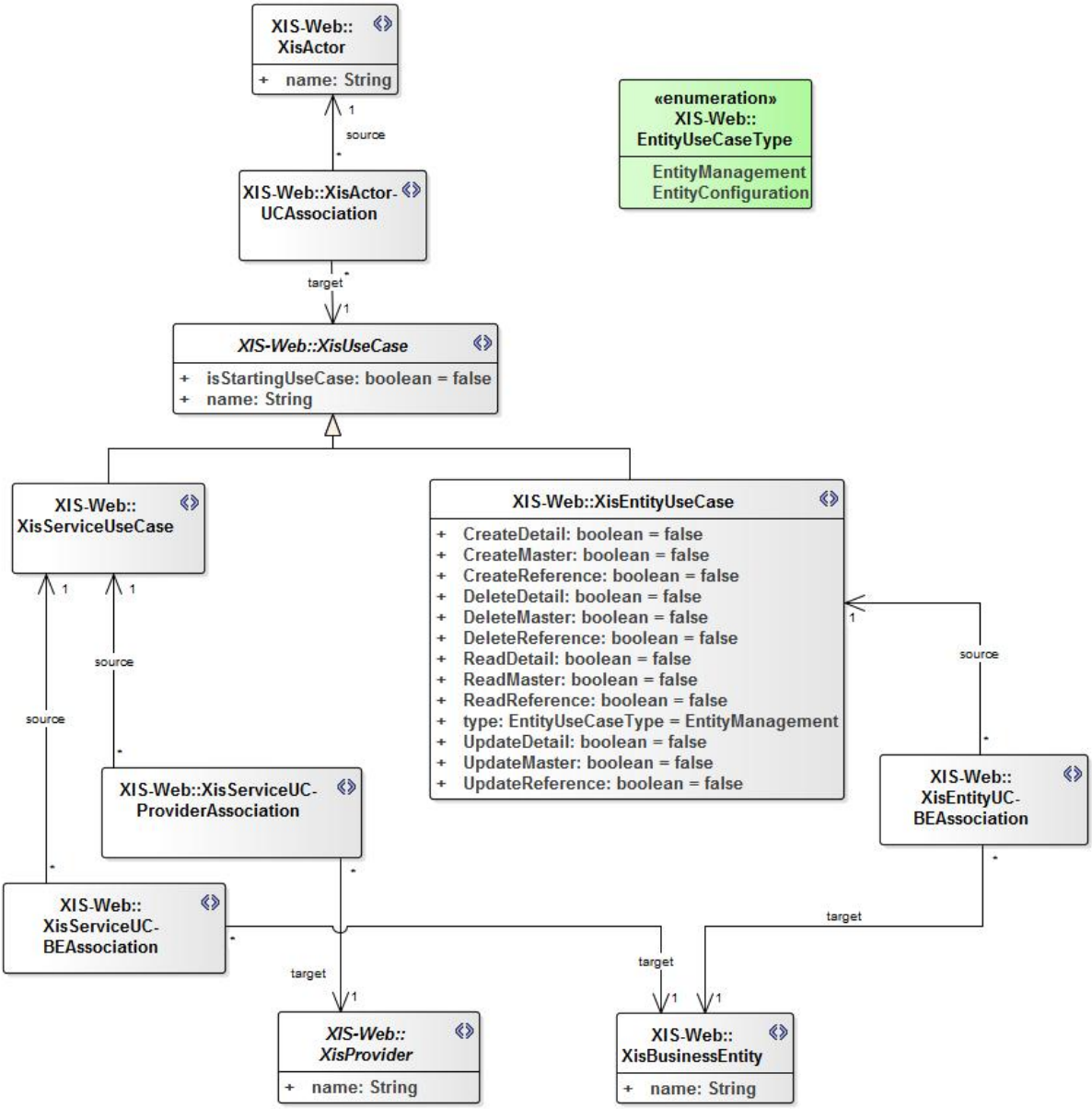


Figure 4.7: Metamodel of the UseCases View

In the UseCases View concerning the TimeSlot Booking App (Figure 4.8), one XisActor was defined, that performs three XisUseCases: (1) ”Manage TimeSlots”, which is a XisEntityUseCase of type ”Entity Management” associated to the TimeSlotBE (from the BusinessEntities View) via a XisEntityUC-BEAssociation; (2) ”Share TimeSlots” which is a XisServiceUseCase that extends the ”Manage TimeSlots” behavior by handling the services provided by the XisServer (from the Architectural View) and (3)

”Manage Courses”, also a XisEntityUseCase, associated to the CourseBE (from the BusinessEntities View) via a XisEntityUC-BEAssociation.

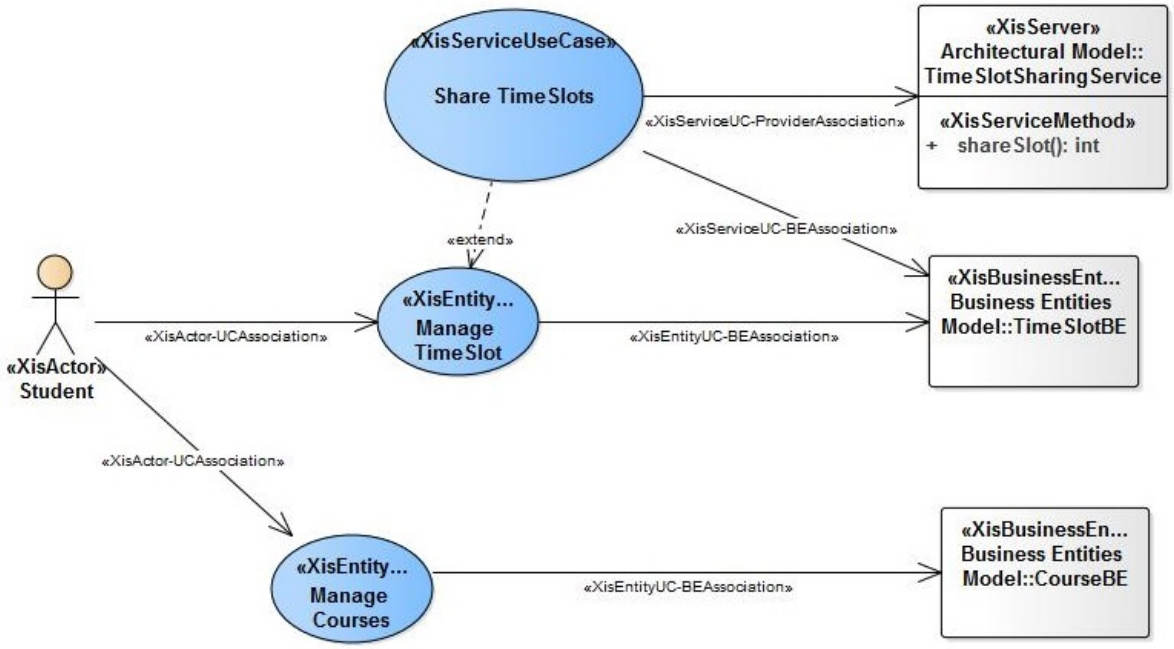


Figure 4.8: UseCases View of the TimeSlot Booking App

4.4 User-Interface View

The User-Interfaces View describes the screens of the web application, known as interaction spaces, and the navigation between them. Like the Entities View it encapsulates two other views: (1) the NavigationSpace View and (2) the InteractionSpace View. The first is where the navigation flow between interaction spaces is depicted, while in the second the actual constituents of the interaction space are represented in detail. A more thorough description is provided below.

4.4.1 NavigationSpace View

In the NavigationSpace View, the navigation flow between the various interaction spaces of the application is described using a Class Diagram. Directed associations with the stereotype XisInteractionSpaceAssociation are the means to represent the navigation. XisInteractionSpaceAssociations have in both ends a XisInteractionSpace (from the InteractionSpace View), making it the only stereotype available to use in this view. It is also responsible to show the action (or event) that triggers the navigation between spaces. The metamodel of this view is shown in Figure 4.9

Considering the TimeSlot Booking App’s NavigationSpace view, there are six XisInteractionSpaces with several XisInteractionSpaceAssociations between them (as represented in Figure 4.10): (1) HomeIS that is the typical entry point of the aplication; (2) TimeSlotListIS, that provides a list of all time slots; (3) TimeSlotEditorIS, that allows for the creation, edition and visualization of the attributes of a time slot (while listing the topics that also belong to that time slot); (4) TopicEditorIS, that allows for the CRUD operations for a topic; (5) CourseListIS that provides a list of all courses and (6) CourseEditorIS that allows for the CRUD operations on courses.

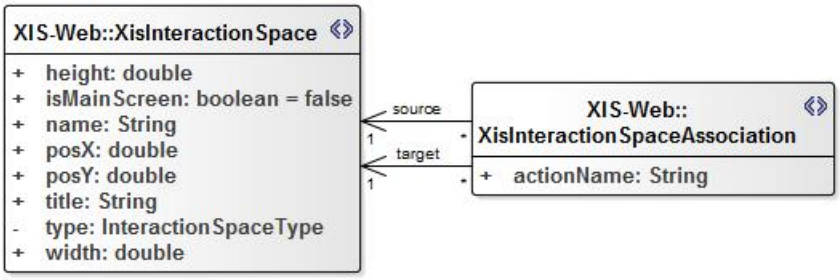


Figure 4.9: Metamodel of the NavigationSpace View

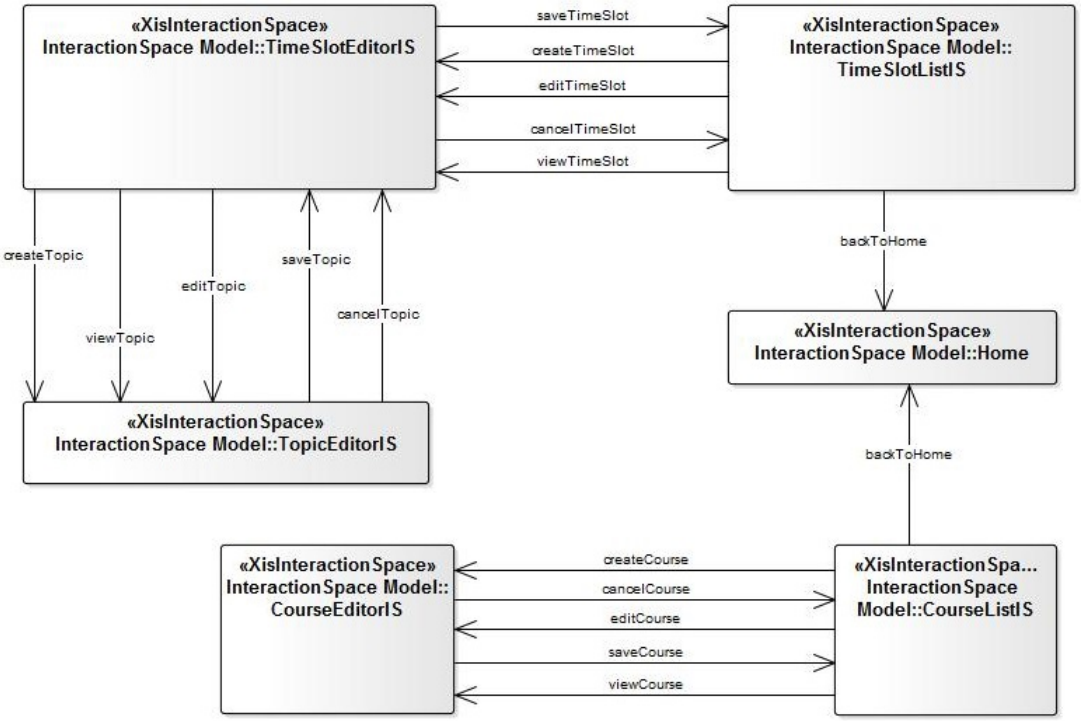


Figure 4.10: NavigationSpace View of the Course Booking App

4.4.2 InteractionSpace View

The InteractionSpace View is the direct representation of a web application's screens or interaction spaces, and is achieved via a Class Diagram. Considering the amount of elements that can take part in a given interaction space, this is the most complex view in XIS-Web. Details like the layout of the UI and events that widgets can trigger are all defined within this view and all this information is fed the M2T transformations, thus allowing for the generation of the UI source code of the application.

There is a specific stereotype of an interaction space that is the XisInteractionSpace. Considering that in this view we want a detailed representation of the screen, XisInteractionSpace becomes the main element of the view. XisInteractionSpaces have type, which can be: (1) HomeInteractionSpace indicating that it is the home screen of the application; (2) MasterEntityList for the screen where a master entity is listed; (3) MasterEntityEditor, (4) DetailEntityEditor, (5) ReferenceEntityEditor where the CRUD operations over a master, detail or entity are possible, respecitvely; (6) ServiceInteractionSpace indicating the screen that makes the connection to a service (internal or external); and (7) CustomInteractionSpace in case there are types of XisInteractionSpace that don't fit in any of the mentioned earlier.

Each XisInteractionSpace is composed of one or more XisWidgets and can be connected to a busi-

ness entity, through a XisIS-BEAssociation, providing the scope of domain entities that can be bound to its XisWidgets.

A XisWidget is the direct representation of a UI widget or control and they can belong to one of two categories: (1) XisCompositeWidget - a container widget that groups other XisWidgets (following the Composite design patter) [51] or (2) XisSimpleWidget - representing the set of simple controls.

Most of XisSimpleWidgets are specializations of either XisInput or XisMultimedia. The XisMultimedia stereotype groups the widgets that are related with media content like: (1) XisAudio, (2) XisImage, (3) XisVideo, (4) XisMap, (5) XisEmbed and (6) XisIFrame. On the other hand, input types that are represented under XisInput can be: (1) XisTextBox, (2) XisCheckBox, (3) XisRadioButton, (4) XisButton, (5) XisLink, (6) XisDatePicker and (7) XisTimePicker. XisSimpleWidgets that do not fit under either of these generalizations are XisSiteMap, XisLabel, XisMenuItem and XisDropdownItem.

XisCompositeWidgets are typically lists. The XisAbstractList is an abstract stereotype present in the InteractionSpace View metamodel (see Figure 4.11)) that allows for an aggregation of the XisCompositeWidgets that can be viewed as lists. These can be: (1) XisList, (2) XisMenu, (3) XisSlider, (4) XisImageSlider and (5) XisDropdown.

XisList represents a the list control, used to display instances of a given item. XisLists can only be composed of XisListItems inside, thus allowing the inclusion of any set of XisSimpleWidgets.

Xis Menus can be of type ”Options Menu” or ”Context Menu”. The first is meant to represent the actions that are available to the page and the second is more oriented torwards actions within certain XisCompositeWidgets. XisMenuItem is the element allowed inside menus that carry inside the correspondent XisAction.

The XisSlider is also an abstract stereotype that means to group multimedia sliders. Currently XIS-Web only supports a XisImageSlider which represents a collection of XisImages.

The last XisAbstractList is the XisDropdown. This widget represents dropdown menus and the stereotype allowed inside is the XisDropdownItem.

Other types of XisCompositeWidget can be (1) XisForm - that corresponds to the user form control, (2) XisDialog - a pop-up screen and (3) XisVisibilityBoundary. A XisVisibilityBoundray is a stereotype that allows for the user to define different views inside the same screen, since it can group XisSimpleWidgets inside and define permissions.

In the TimeSlot Booking App, the TimeSlotListIS and CourseListIS are MasterEntityList interaction spaces. Typically these interaction spaces are composed by a XisImage of the site logo, a XisSiteMap containing the other interaction spaces with navigation, a XisList displaying the instances of the master entity with a context XisMenu associatied and a XisMenu of type ”Options Menu” that aggregates the actions allowed in the page (a possible representation of this interaction space is given in Figure 4.12).

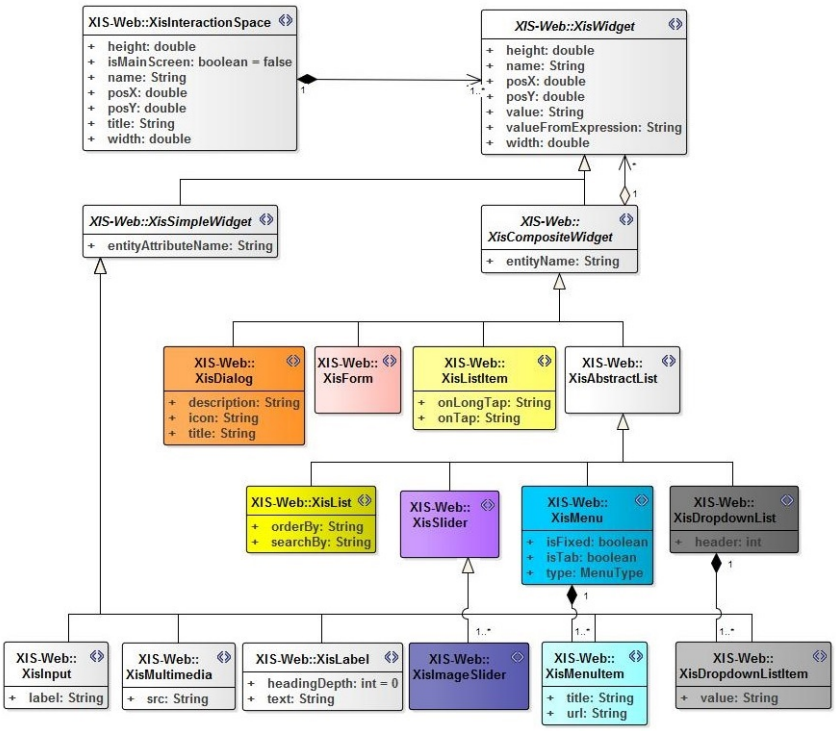


Figure 4.11: Excerpt of the Metamodel of the InteractionSpace View

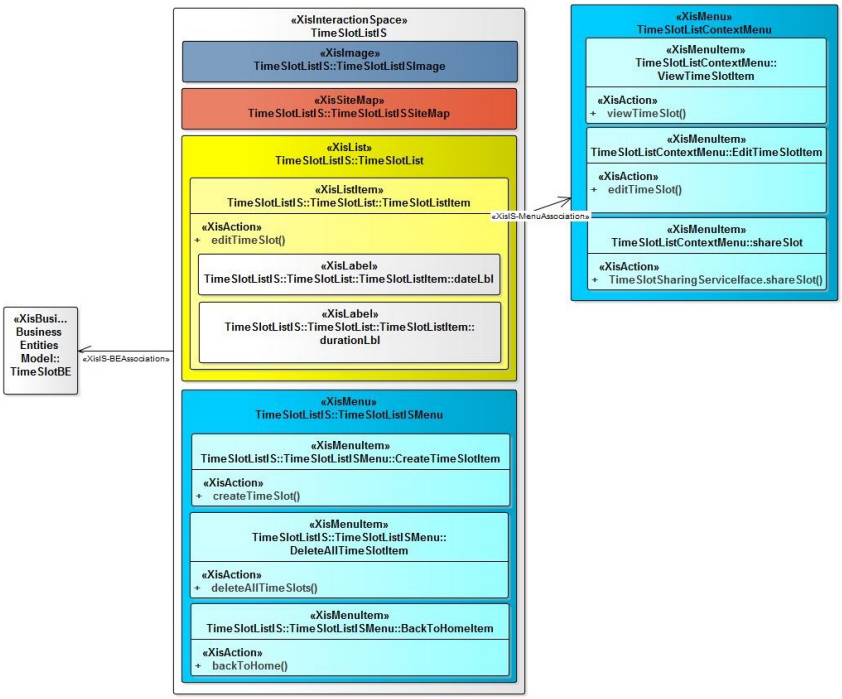


Figure 4.12: The TimeSlotListIS interaction space of the TimeSlot Booking App

Chapter 5

XIS-Web Framework

The XIS-Web technology has a companion framework that acts as an integrated environment, supporting the Model-Driven Development (MDD) of web applications that have been modeled using the XIS-Web language. In this chapter we provide an overview of the XIS-Web framework in Section 5.1, by describing the architecture behind it, the technologies supporting it and the development process that is proposed and by explaining the modifications that took place in EA in order to allow designing models using the XIS-Web language. The explanation of how Model-to-Model (M2M) transformations are accomplished in the XIS-Web framework is given in Section 5.2 and finally in Section 5.3 a description on the Model-to-Text (M2T) is given and also its implementation is explained.

5.1 Overview

One of the key edges of building web applications using the XIS-Web language is the fact that it has an MDD-based framework that supports its automatic creation. The suggested development process of a web application using the framework consists in four steps: (1) modeling the required views (via the Visual Editor), (2) validating those views (using validation rules that are built in the framework), (3) generating the User-Interface Views (assuming ”smart” approach) and finally (4) the generation of the application. By the end of step (3) there is window of opportunity for the designer to tune the application that he is building by customizing the models that were automatically generated by the Model Generator. It is also important to emphasize, that in this suggested development process, only step (1) is manual.

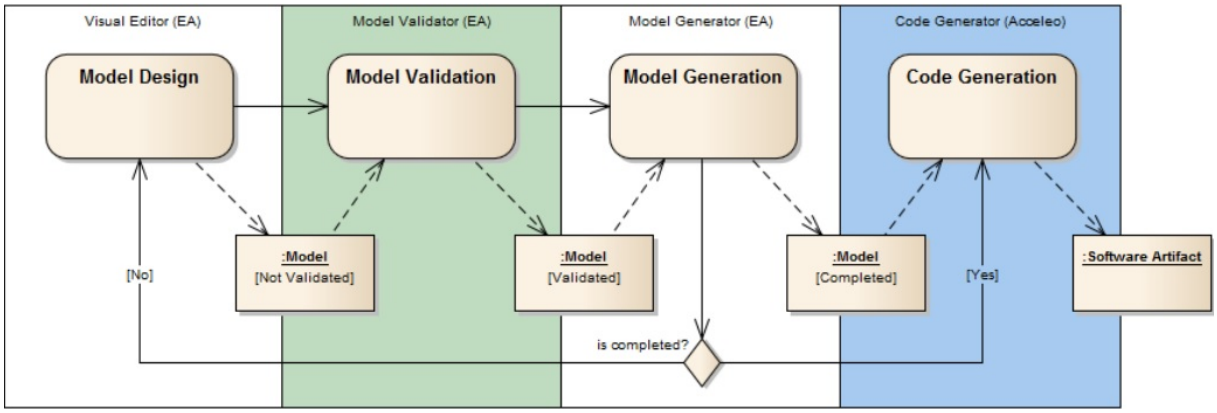


Figure 5.1: XIS-Web Technologies and suggested development process (extracted from [1])

The XIS-Web framework is implemented on top of EA, leveraging it as (1) Visual Editor, (2) Model

Validator and (3) Model Generator via its Model Driven Generation (MDG) Technology function. Alongside EA, the framework also requires the Eclipse Modeling Framework (EMF) for (4) Code Generation. These technologies allows for the parsing and manipulation of models and provides some useful plugins that make the development of web applications with XIS-Web a more seamless process.

Taking into consideration the first step of the suggested development process, the Visual Editor, being built on top of EA via the use of a MDG Technology plugin, allows for the definition of the XIS-Web language as a UML profile that is fully compliant with the OMG specification for UML2. Besides this, EA also allows for the customization of diagram types, toolboxes and model templates (all being put to use by the XIS-Web framework).

Delving into the second step of the suggested development process, the Model Validator works as a plugin using EA's Model Validation API. This validation plays a crucial role in the development process by mitigating the errors that can be made by the designers, reducing the time spent on building the application, improving the quality of the generated models and consequently the quality of the generated code. The current standard for model validation is OCL [52], and the reason that it is not being used in the XIS-Web framework lies with several limitations regarding stereotype validation. Despite this solution not being the standard language, it allows for the definition of constraints, custom error messages, the assignment of severity levels and the immediate navigation to the element that broke the rule and caused the message.

The third step, Model Generator, is implemented as a plugin using EA's Automation Interface. This technology allows the accessing of the repository that is created by EA while models are built within it, that contains the information of the model's diagrams and their elements and it also enables the creation of new diagrams and elements. The Model Generator, taking the information from the diagrams built in step one (the UseCases, Domain, BusinessEntities and Architectural view) typically creates multiple InteractionSpace view diagrams and a NavigationSpace view diagram. The M2M transformations are explained in detail in Section 5.2.

Between the third and fourth step, there is an unmentioned step that consists in the translation of the models that are exported from EA that have to be read by the EMF. This step is required due to compatibility issues between the XMI format generated by EA and the XMI format that is used in EMF and Acceleo. The translation of the models is achieved using a Java program that proceeds to replace the EA's XMI namespaces by the ones supported by EMF, it also proceeds to the conversion of tagged values belonging to elements, allowing them to be transported to EMF.

Finally, we have the Code Generator (fourth step). This step is built using Acceleo, a template-based code generation framework that is included in EMF as a plugin. It implements the MOF Model to Text Language (MTL [53]) standard and allows for the definition of code templates for any kind of model compatible with EMF (thus the XMI conversion between EA and EMF). We can simplify the code templates as being part simple text (area that is not evaluated by Acceleo and is written "as is" in the generated artifact) and part dynamic that consist in Acceleo annotations that are replaced by actual values during generation time.

Enterprise Architect's Model Driven Generation (MDG) Technologies play a very important role in three steps of the suggested development process. It allowed for the tailoring of the development environment in a suitable way for the XIS-Web language, namely by allowing for the definition of custom diagrams with custom toolboxes and custom project templates.

The XIS-Web language profile that was developed in EA, not only defines the stereotypes that are meant to be used in each view, but it also provides a definition for the customization that EA allows. For each aspect that is to be customized, the language profile presents a sub-profile containing the information necessary for that task. The custom diagram profile provides specific metaclasses for each type of diagram and allows associating custom toolboxes to the corresponding diagrams (e.g., toolboxes

that have UseCases stereotypes should be shown when working in a UseCases view). The toolboxes for each diagram also have their own custom diagram profiles. Since currently XIS-Web has six views, there are six sub-profiles for toolboxes, corresponding to the different views.

When creating a new model using the XIS-Web framework, the automatic creation of a project template is provided. This project template creates the package structure with all the views and respecting diagrams (linked to the respective toolboxes) of a XIS-Web project.

5.2 Model-to-Model Transformations

In order to provide support for the ”smart approach” that was earlier described, the XIS-Web framework provides a Model-to-Model (M2M) transformation mechanism. It leverages the information given in the Domain, BusinessEntities, Architectural (optional) and UseCases views to perform the transformations required to generated the User-Interface views. The views from which the transformation mechanism is fed do not suffer any alterations, instead two new views are created: the NavigationSpace view and the InteractionSpace view. Considering that the InteractionSpace view is one of the most complex views of the framework, the automatic generation of it reduces drastically the amount of work required to model a web application using the XIS-Web language. The M2M transformations are performed inside EA (the Visual Editor) and can be triggered via the option ”Generate Models” in the XIS-Web plugin as shown in Figure 5.2.

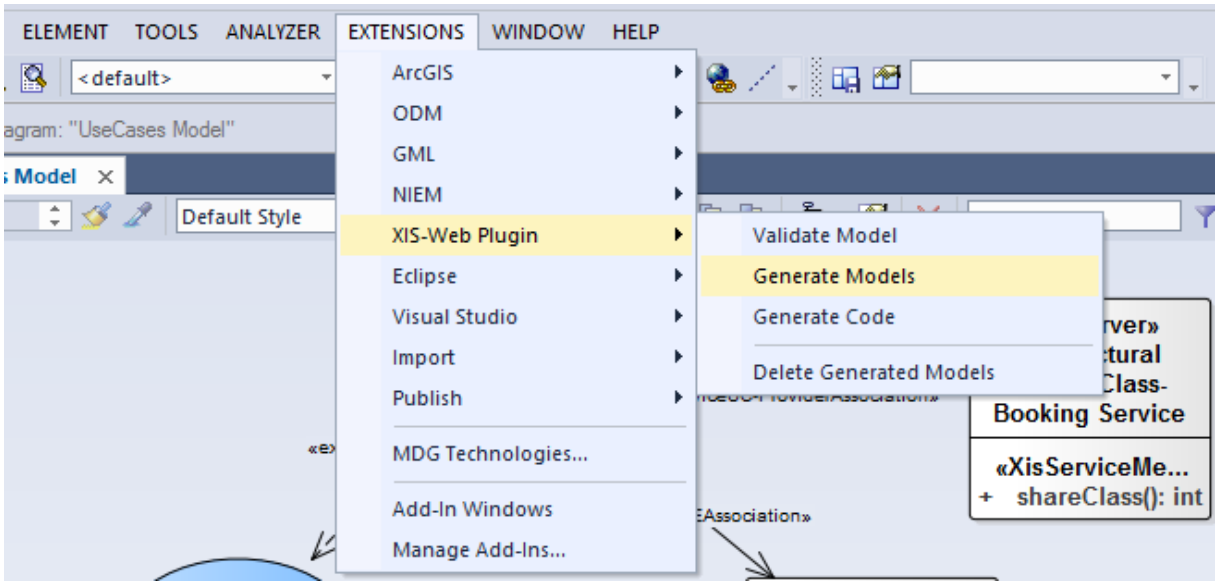


Figure 5.2: Option for Generating Models in the XIS-Web framework.

Considering the views that are used to supply the transformations, the one with most impact in the generated models (and the actual application) is the UseCases view. This view provides the abstractions (i.e., the stereotypes, tagged values and associations) that configure the process. On the other hand, the Architectural view is considered optional because the web application doesn’t require external connections or services. The application of the UI patterns described in Section 2.2.2 is a direct consequence of the tagged values that are defined in a XisUseCase.

XisEntityUseCase

An action made by an Actor that takes effect over a domain entity (in this case a XisEntity) is represented with a XisEntityUseCase. Therefore, this stereotype requires a connection to a business entity (a XisBusinessEntity) that in turn is responsible for the binding and organization of the domain entities. The organization of these domain entities is achieved by the aggregation of several domain entities that have similar concerns under the same business entity, with different roles. A XisEntityUseCase only manipulates directly a domain entity that has a master association to the business entity. A business entity can only have one domain entity associated with the role Master.

The set of tagged values contained in the XisEntityUseCase allow for the representation of the CRUD (Create, Read, Update, Delete) and Search operations for the master, detail and reference entities (this is depicted in Figure 5.3). Besides the definition of these operations, a XisEntityUseCase type also has a direct relation to the interaction space type that will be generated. A XisEntityUseCase of type ”EntityManagement” will originate interaction spaces that allow for the management of multiple instances of the master entity, the management of a single Master entity chosen from that list, and the same for the detail and reference entities, if they have a ”many-to-many” relationship with the master entity.

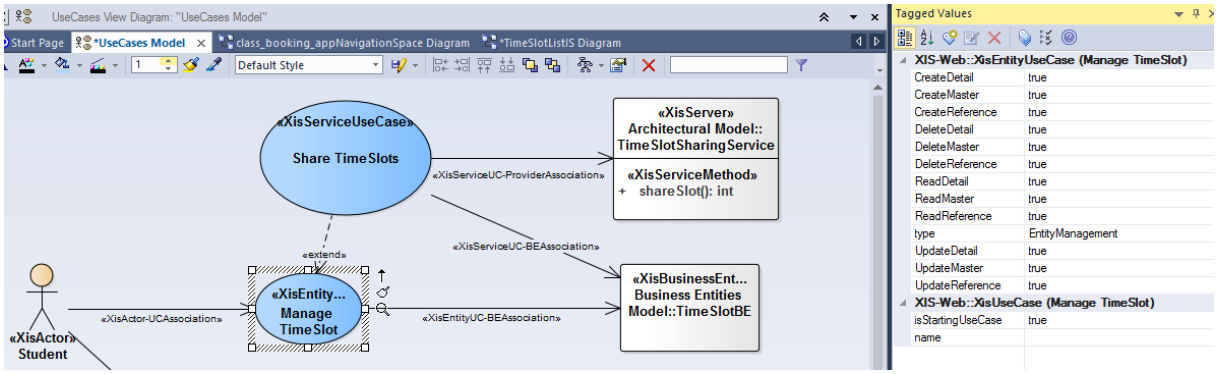


Figure 5.3: UseCases view of the TimeSlot Booking App depicting the tagged values for a XisEntityUseCase.

Using the TimeSlot Booking App as a running example, namely its views and in particular its UseCases diagram that was shown in Figure 5.3 we have the generation of three screens (XisInteractionSpaces), that are represented in Figure 5.4. Due to space and visibility restrictions, the associations between interaction spaces and business entities have been omitted.

The first depicted interaction space is a XisInteractionSpace of type MasterEntityList, and thus it allows for the management of the Master entity ”TimeSlot”. The second one still refers to the same master entity, allowing the edition of a single instance (MasterEntityEditor). Considering the information given in the Domain view of this case study, a ”TimeSlot” can have one or more ”Topics” which causes the inclusion of the management of the instances of detail entity in the same interaction space of the edition of a single instance of the master entity. The approach taken to minimize the complexity of the interaction space, while allowing it to be fully responsive to any device is to add a collapsible panel (a XisCollapsible), containing the part of the page that regards the detail entity, after the form that allows for the edition of the fields of the master entity. This reduces the visual cluttering that would be otherwise created just by having these elements appearing together, while keeping the application responsive to any device. The same pattern would be applicable if the master entity had a similar connection to its reference entity. The third screen shows the interaction spaces that allows for the edition of the ”Topic” entity and it is similar to the second one, minus the part of the collapsible panel.

On the other hand, a XisEntityUseCase of type ”EntityConfiguration” results in the creation of only the second and third interaction space, relinquishing the first space and the option for managing multiple instances of the same entity.

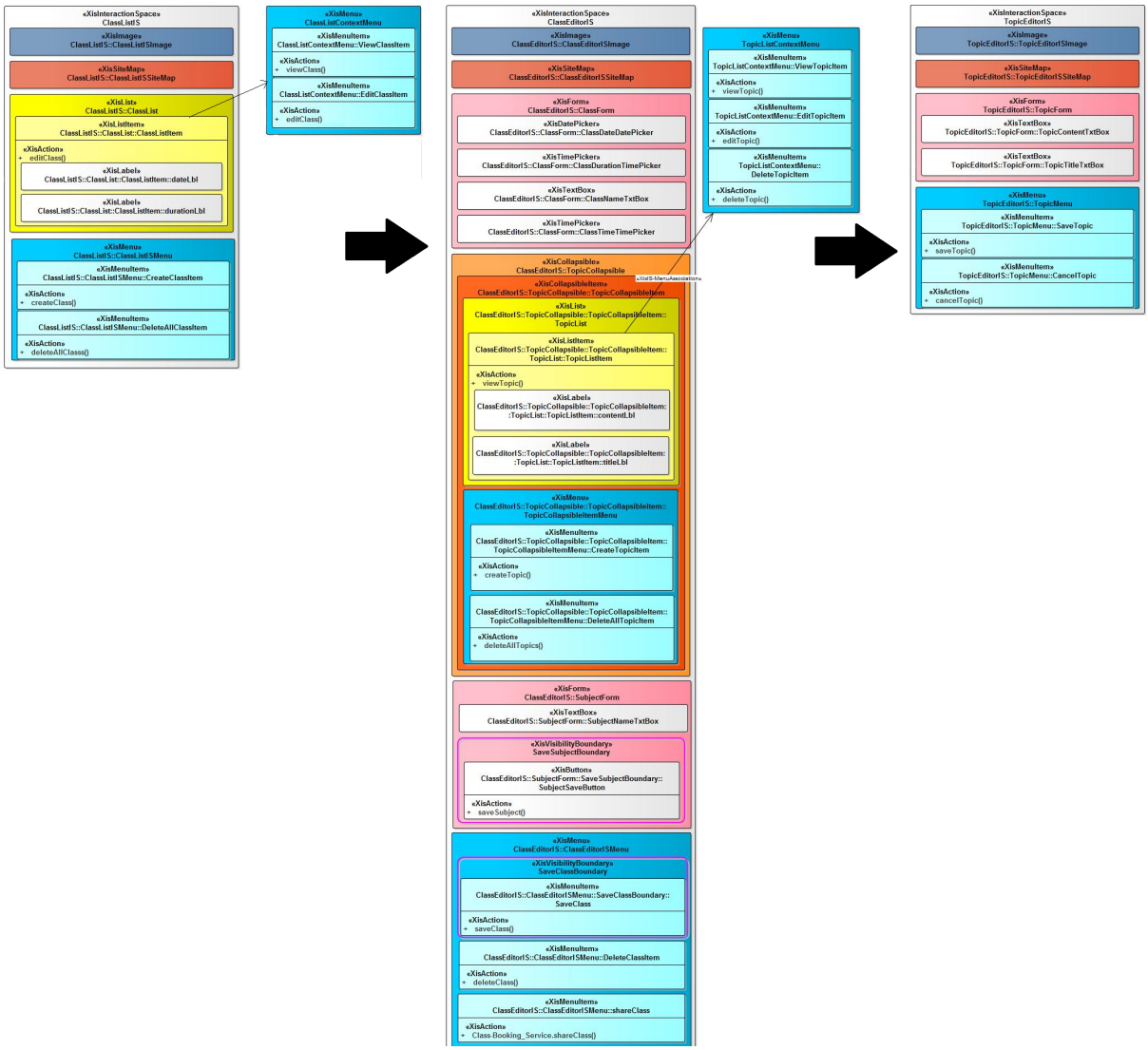


Figure 5.4: Example of generated interaction spaces for a XisEntityUseCase of type ”EntityManagement”.

XisServiceUseCase

A XisServiceUseCase is used to represent an action that uses the operations (XisServiceMethods) realized by a provider (XisProvider from the Architectural view). Because of this, a XisServiceUseCase is required to be associated with a provider, while also being able to be connected to a business entity. This double connection states that operations realized by the associated provider have direct influence on the associated business entity (that can be an aggregate of domain entities). One applicable pattern, regarding XisServiceUseCases is to have them as ”standalone” UseCases (see Figure 5.5 for an example with the TimeSlot Booking App). This ultimately originates a XisInteractionSpace of type ServiceInteractionSpace that allows for the management of a single master entity and the application of the operation realized by the provider on that same entity, demonstrated in Figure 5.6.

XisEntityUseCase extention with XisServiceUseCase

To add flexibility to the language, and to provide more options during development and generation

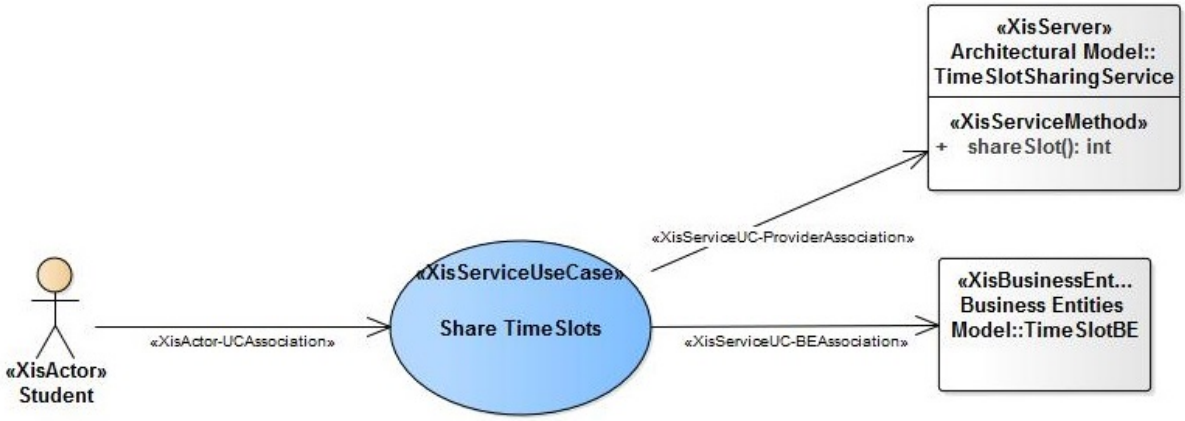


Figure 5.5: Example of a UseCases view with a XisServiceUseCase.

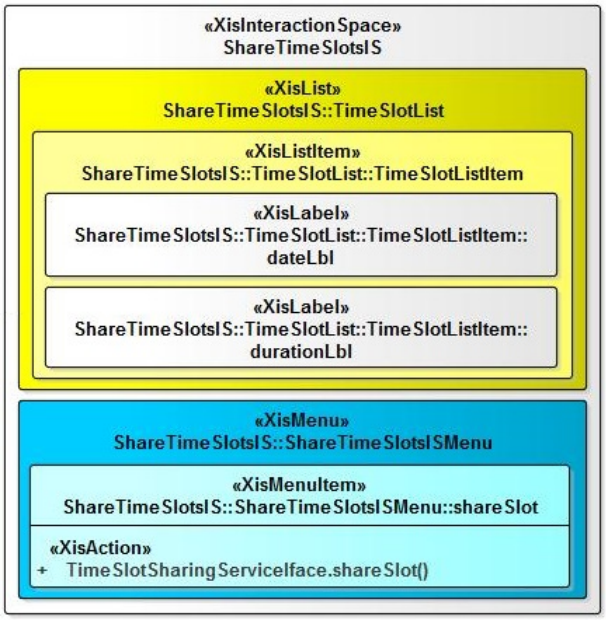


Figure 5.6: Example of a XisInteractionSpace generated from a XisServiceUseCase.

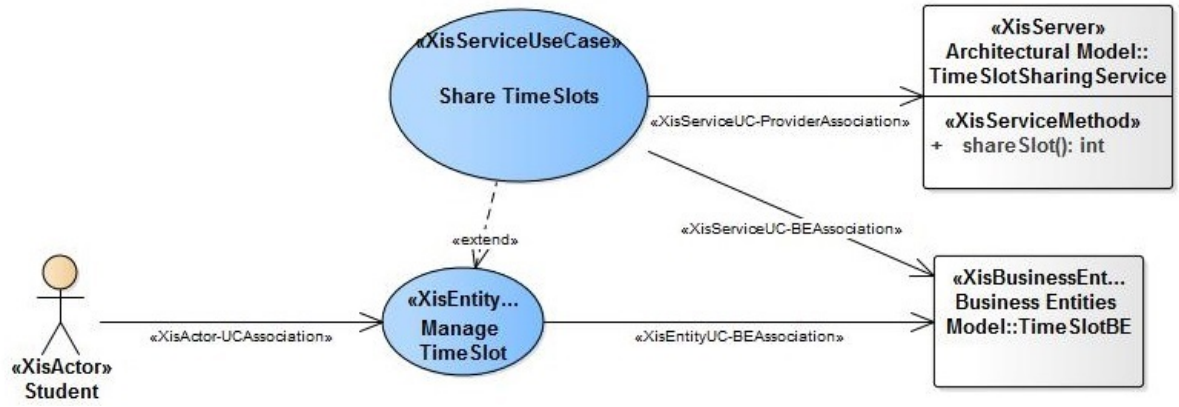


Figure 5.7: Example of a UseCases view with a XisServiceUseCase extending a XisEntityUseCase.

time, the XIS-Web also allows for the extension of a XisEntityUse case with one or more XisServiceUse-Cases. As consequence, interaction spaces will be generated maintaining the model described for

XisEntityUseCases, with the added feature of new options (XisMenuItems) to the options menu (XisMenu) that was generated in the context of the XisEntityUseCase. Figure 5.7 illustrates an example of an application of this pattern, concerning the TimeSlot Booking App, followed by Figure 5.8 that shows an interaction space generated from the previous diagram where the new action is highlighted.

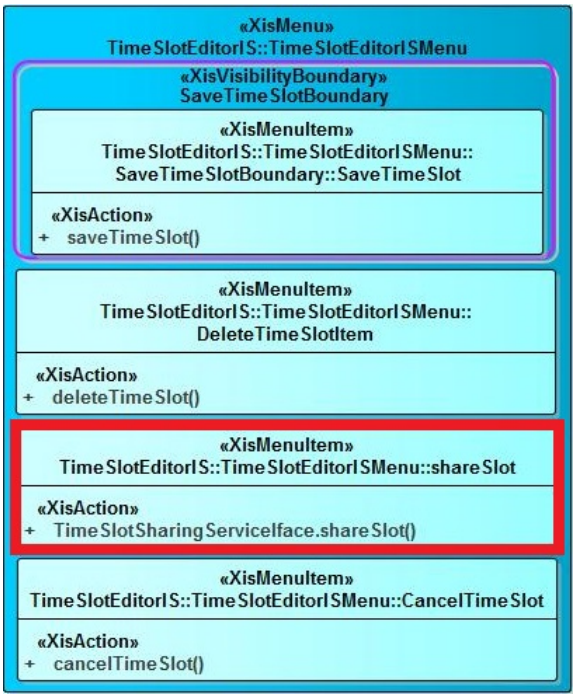


Figure 5.8: Part of the TimeSlotListIS XisInteractionSpace displaying the XisAction performed by the XisServer.

Multiple Independent XisUseCases

In cases where there are multiple XisUseCases that are not related to each other (not extending or including), yet a different pattern is applied. The designer is obligated to chose which of the XisUseCases should originate a home interaction space (a XisInteracitonSpace of type HomeInteractionSpace). From this the List navigation pattern (described in Section 2.2.2) is applied for the remaining XisUseCases. Support for more navigation patterns is planned to be added to the framework, namely through the usage of the Springboard Menu and Navigation Tree (with page hierarchy).

5.3 Model-to-Text Transformations

The M2T transformations step in XIS-Web is implemented using Acceleo, an Eclipse plugin. As mentioned before, it is a template-based code generator that allows for the parsing of models compatible with EMF and the writing of software artifacts that use the information of those same models. It was created by Obeo, a company from Toulouse that is specialized in providing software solutions to create and transform complex systems. It was first released for Eclipse in 2006 and later, in 2009 it was included in Eclipse's M2T project.

Acceleo templates revolve around two major constructs: queries and templates. Queries are used for retrieving information from the model that is being fed to Acceleo, and so, they do not generate text. These are specified with the [query .../] tags that allow for the use of OCL inside. Templates are a set

of statements (that can either be inherent to Acceleo or built afterwards) that are used to generated any kind of text. They are delimited by the [template .../] tags.

For more complicated features, Acceleo also provides a java service wrapper that allows the circunvention of templates and queries. The java service wrapper is also able to navigate the model, performing computation that would otherwise be impossible with just queries and templates. After this computation is complete, Acceleo provides a service invoker library that allows for the java services to be invoked as a traditional query. Figure 5.9 shows an example of a code template for Acceleo that in this case, is producing HTML5 code assuming there is a class named "TimeSlot" in the model that contains four attributes (date, time, duration and name).

Acceleo Code Template:

```
[template public writeCode(c : Class)]
[comment @main /]
[file (c.name.toUpperFirst().concat('.html'), false, 'UTF-8')]
     <div id="[c.name.toUpperFirst()/]" class="container">
     [for p : Property | c.attribute separator('\n')]
           <span class="lbl lbl-default">[p.name]</span>
     [/for]
     </div>
[/file]
[/template]
```

Sample Generated Output:

```
<div id="Class" class="container">
  <span class="lbl lbl-default">date</span>
  <span class="lbl lbl-default">time</span>
  <span class="lbl lbl-default">duration</span>
  <span class="lbl lbl-default">name</span>
</div>
```

Figure 5.9: Example of an Acceleo Code Template followed by a possible Generated Output.

Being Acceleo an Eclipse plugin, it provides a template editor that give some assistance to when developing the templates, a profiler tool for analyzing the quality of the generated code and a debugging framework that works similarly to the Eclipse's own debugging framework, by allowing the placement of breakpoints in the code, the navigation between instructions and the validation of expressions.

The project containing the code templates defined for the XIS-Web framework was developed considering Acceleo's best practices regarding naming conventions, project and module organizations. Figure 5.10 shows the organization of the Acceleo project of the XIS-Web code generator. There are four main packages and their description is as follows: (1) common - contains the utility modules (i.e., the queries that are commonly used by templates, and the templates that serve as backup for the application. A very recurring query in this package is the confirmation of a XIS-Web stereotype contained in a UML element.), (2) files - a packaged that contains the modules that generate files, (3) main - the entry points of the generator, and finally (4) services - the package containing the modules that make use of Java service wrappers and the corresponding Java files.

Currently, the XIS-Web framework supports the generation of web applications that rely on HTML5, JavaScript, WebSQL and CSS3. The projects that are generated by the framework are responsive web applications that can run on any device and are ready to be used by the end-users and do not require any compilation or edition.

The technologies chosen for the XIS-Web framework were the same as for the XIS-Mobile framework, considering that XIS-Mobile was a sucessful implementation of the XIS methodology and that a

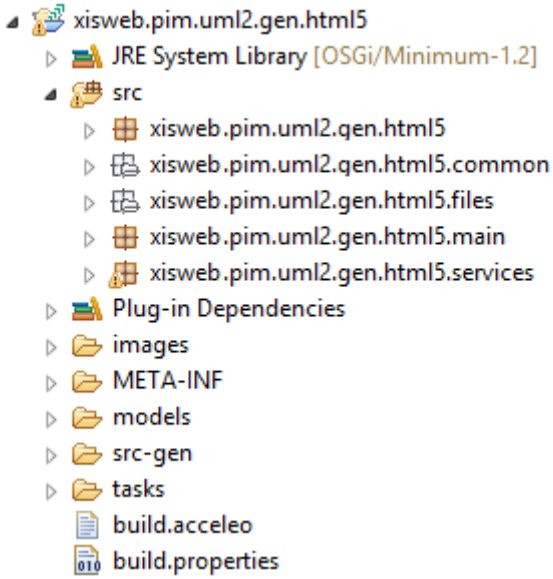


Figure 5.10: Acceleo project organization for HTML5.

survey regarding the most suitable techonolgies for these subjects was made on that work [1][49].

Chapter 6

Evaluation

XIS-Web was evaluated in three different ways. First, as described in Section 6.1, the evaluation of XIS-Web's M2T capabilities is done via the comparison between generated case studies and manual implementation of the same case studies - the dDocs App and the TimeSlotBooking App. Second, Section 6.2 presents the results of the pilot-user test session in which a set of 12 independent users tested XIS-Web focusing on three aspects: (1) the Language (namely if it is a good fit for the domain and its learning curve), (2) the Framework (namely the Visual Editor, Model Validator, Model Generator and Code Generator) and (3) the General Approach. Then, Section 6.3 presents a comparison between XIS-Web and the related work. Finally, Section 6.4 describes the main conclusions obtained from this evaluation.

6.1 Code Generation vs Manual Implementation of Case Studies

This section presents and reviews the results obtained from the implementation of the Case Study A (TimeSlot Booking App) and the Case Study B (dDocs App), presented in Section 3.4, comparing the manual implementation with the code that was generated. The metric used to perform this evaluation was the ratio between the lines of code (LOC) generated and the lines of code for the manually implemented versions of the applications. The measurements were performed using the open source project CLOC - Count Lines of Code[1].

Languages	TimeSlot Booking App			dDocs App		
	Manual	Generated	Ratio	Manual	Generated	Ratio
JavaScript	259	226	87,3%	367	319	86,9%
HTML	416	320	76,9%	797	464	58,2%
Total	675	546	80,8%	1127	783	69,4%

Table 6.1: Ratio between generated code and manually implemented code, per language and total.

After careful analysis of Table 6.1, we can draw some conclusions. Regarding the Case Study A (TimeSlot Booking App), the generation ratio was of 80.8% for the whole application, namely 87.3% for the JavaScript and 76.9% for the HTML part. While these results are quite positive, meeting one of the proposed goals (G4), the main issue found for this case study in particular (when comparing to the manual implementation), was the non-generation of the "shareTimeSlot" method that is executed by an external service. In cases of external logic, XIS-Web generates a simple stub for the method.

[1] http://cloc.sourceforge.net

This happens because the language does not to capture the intents and logic that are executed by a WebService (a XisServer).

Regarding Case Study B, the results are not as good as for Case Study A, showing a total ratio of 69.4%. This happens because the dDocs app presents more complex logic and UI patterns that are not yet implemented in the XIS-Web Framework. For example, the generated code lacked the Gallery view of the documents, the Dashboards containing relevant data from documents and the communication with a third party repository to backup the documents.

Figure 6.1 shows some screens that result from the generation of the source code for Case Study A - TimeSlot Booking App.

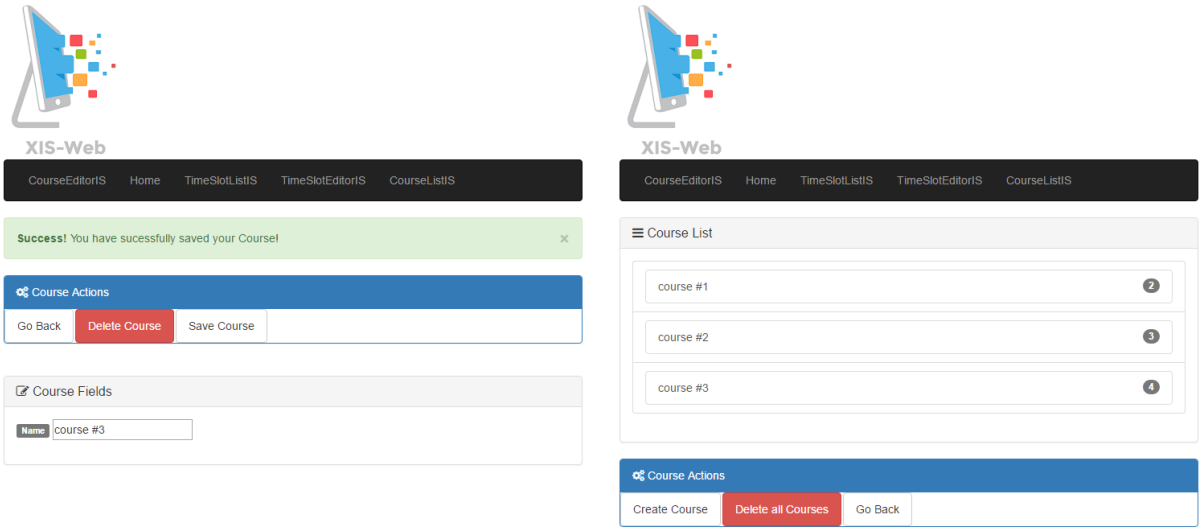


Figure 6.1: Result of generating CourseEditorIS and CourseListIS (from left to right).

Overall, the results obtained were positive, allowing the achievement of one of the Goals (G4) and enabling a proof of concept. The percentage of code generated for Case Study A was greater than the one generated for Case Study B (by about 10%), which allow us to conclude that the first case study was simpler than the second one and provided a better match with the XIS-Web language.

6.2 User Session Evaluation

A pilot-user test session was conducted in order to better evaluate the XIS-Web technology. The participants were people not directly involved in the research work and its main goal was to detect potential bugs and user limitations. The group had 12 elements, with ages ranging from 23 to 30 and with at least a Bachelor of Science degree. Three of the participants had no previous experience with UML, half of the participants had also no experience with web application development and 4 participants had professional experience in Information Technology. The user session was conducted under the following conditions:

- Session took place in a controlled environment (laboratory);
- Participants should follow the tasks described in the script (Appendix B);
- The tasks were performed without previous use and learning;
- The user was required to do the test using a computer running Windows with Java and Sparx Systems Enterprise Architect (version 7.5 or above) installed;

- While the users performed the tasks, their behavior and performance could be logged;
- Users were free to think out loud and share ideas.

The participants received a 10 minute presentation explaining the fundamental concepts of the XIS-Web language and its framework. This was followed by a demonstration of the developed of a simple case study using the XIS-Web framework that demonstrated how to create a XIS-Web application, starting from modeling the views until the code was generated. The participants were asked to follow a script (Appendix A) that described The TimeSlot Booking App described in Section 3.4 and should create this application using XIS-Web. The participants should validate their models, generate the User-Interface views (the smart approach) and launch the code generation. The average time for the 12 participants was of 38 minutes (since they started modeling until they had the code generated). In the end the participants were asked to fill a questionnaire to rate the XIS-Web language, the framework and the approach taken. The analysis of the questionnaire's results is described below.

6.2.1 Analysis

The analysis of this experiment was split in the following aspects: language, framework and general approach, which is a common evaluation method used in the XIS-* family. Section 6.2.1.1 presents and discusses the evaluation made regarding the language, Section 6.2.1.2 analyses the evaluation made regarding the framework and finally, Section 6.2.1.3 describes and discusses the evaluation made regarding the general approach.

6.2.1.1 Language Analysis

The XIS-Web language part had five questions:

- **L1.** How suitable is the size (number of concepts) of the language?
- **L2.** How easy to use is the notation (defined as a UML Profile)?
- **L3.** How easy is to learn the language without the UI concepts?
- **L4.** How easy to learn is the language without the UI concepts?
- **L5.** How suitable is the language for the Web Apps development domain?

Table 6.2 shows the average score for each question concerning the XIS-Web Language. Considering that the scale provided was from 1 to 5, we can observe that all questions had a positive score (greater than 3), which denotes some success. From the answers provided by the participants, regarding the first question, we can conclude almost all participants found the language to be of a suitable size (giving it a score of 4 or 5), while two participants found it to be medium (score of 3).

From the Language questions, the one that got the least favorable result (of 3.92 - which is still a very good mark), was the one referring to the difficulty of learning the language without the UI concepts. From this result and in comparison to the scores of the other questions, we can conclude that the participants found that the language, despite having more stereotypes, makes more sense when integrated with the User-Interface Views.

On the other hand, the question with best average score (of 4.25) was the one refering to the usage of the notation defined (UML Profile). From this result we can observe that the participants found that XIS-Web, being a DSL created as a UML Profile, is easy to use, due to the concepts that derive from the UML. Participants also found this language very suitable for developing web appilcations with.

Overall, considering the results obtained regarding the Language part, we assume that the XIS-Web language contains several concepts and initially can be a bit difficult to learn and understand all of them. This effect is somewhat mitigated from the fact that XIS-Web is a UML Profile. So, taking into consideration the size, notation and concepts, we believe that this language is adequate for the web application development domain.

XIS-Web Language					
Average	Language.1	Language.2	Language.3	Language.4	Language.5
	4	4.25	3.92	4.17	4.17

Table 6.2: Average score (in scale of 1-5) by question for the XIS-Web Language.

6.2.1.2 Framework Analysis

The XIS-Web framework part included five questions:

- **F1.** How do you rate the usability of EA with the XIS-Web plugin?
- **F2.** How do you rate the usability of the Model Editor (Stereotypes, Toolboxes, Project Template)?
- **F3.** How do you rate the usability of the Model Validator?
- **F4.** How do you rate the simplicity of the Model-to-Model transformation process?
- **F5.** How do you rate the simplicity of the Model-to-Text transformation process?

Table 6.3 shows the average score for each question concerning the XIS-Web Framework. Comparing the results obtained in the Framework questions to the ones obtained in the Language questions, we can see that the framework questions got an even better score, with a minimum of 4.17, on a scale of 1 to 5. Particularly, Questions F4 and F5 got very good results, being close to the maximum value possible. With this, we can conclude that the participants found both the Model-to-Model and Model-to-Text transformations, mechanisms that were simple to use and that brought benefits to the development process.

Considering the overall results of these questions, we can observe that all parts of the XIS-Web framework show that they possess good usability and usefulness. Objectively, Question F2 was the one with the lowest value, which can lead us to conclude that the both the Model Editor still has room for improvements. From participant feedback, we conclude that this usability issue is related with the lack of flexibility from EA, namely when deleting instances.

XIS-Web Framework					
Average	Framework.1	Framework.2	Framework.3	Framework.4	Framework.5
	4.5	4.17	4.42	4.75	4.83

Table 6.3: Average score (in scale of 1-5) by question for the XIS-Web Framework.

6.2.1.3 General Approach Analysis

The XIS-Web General Approach aspect included two questions:

- **A1.** How do you rate the productivity with XIS-Web when comparing to the traditional software development process?

- **A2.** Would you use such a tool on your own Web App projects?

Table 6.3 presents the average score for each question concerning the XIS-Web General Approach. Both question got positive results, but the participants found XIS-Web a tool that increases productivity when comparing to the traditional approach, rather than using it on projects of their own. This can be related to the constraints that involve the use of XIS-Web, namely the Operative System (must be Windows) and the dependency of both Enterprise Architect and Java.

XIS-Web Framework		
Average	Approach.1	Approach.2
	4.5	4.17

Table 6.4: Average score (in scale of 1-5) by question for the XIS-Web General Approach.

Overall, as Table 6.5 demonstrates, the results were very encouraging because all the reviewed aspects got positive scores. From the Language questions we concluded that the language was not that easy to learn, and its something we pretend to change in future work, by refactoring it and reducing the number of concepts available. From the Framework questions we learned that the Model Editor was the weakest point of the framework. The participants found that XIS-Web brings significant productivity gains when comparing to the other frameworks, but its constraints regarding the development environment may lead them not to use it in their projects.

We conclude that the number of participants in this session (12) is sufficient to take meaningful conclusions, considering that experts in usability claim that a group of 5 testers is enough to reveal over 80% of the usability problems [54]. Moreover, given that our questionnaire focuses on the usability of the language, framework and its approach, 12 is a reasonable number for an exploratory assessment, in order to identify challenges in usability. The complete results of the questionnaire are provided in Appendix B.

XIS-Web			
Average	Language	Framework	General Approach
	4.1	4.53	4.33

Table 6.5: Average score (in a scale of 1-5) for each of the surveyed XIS-Web aspects.

6.3 Comparison with Frameworks from Related Work

Tables 6.6 and 6.7 summarize the technologies presented in Section 2.3 and provide comparison with XIS-Web. The platforms were evaluated based on the framework defined originally in [3]: (1) number of views, (2) abstraction Level of each view, (3) perspective of each view, (4) modeling language, (5) types of transformations, (6) transformation language, (7) metamodeling language, (8) target platform (for code generation) and (9) tool support.

XIS-Web, when compared with the other surveyed technologies, stands out in four key aspects:

- Separates the modeling of a web application in six views, which ultimately promotes a ”separation of concerns” principle that is key to managing complexity in software development;
- Generates the interaction spaces and navigation flow between them, relieving this cumbersome task from the user, allowing a quicker TTM and increasing productivity;

- Employs the latest generation web technologies that allow the required flexibility when developing responsive web applications;

- The definition of the language as a UML profile, which is a standard in modeling and allowing the creation of PIMs throughout the modeling process, thus lowering the learning curve.

View organization and Abstraction Level			
Name	View	Abstraction Level	Perspective
XIS-Web	Domain	PIM	Static
	BusinessEntities	PIM	Static
	Architectural	PIM	Static
	UseCases	PIM	Dynamic
	NavigationSpace	PIM	Static
	InteractionSpace	PIM	Static
Hera	Conceptual	PIM	Static
	Navigation Space	PIM	Dynamic
	Presentation	PIM	Static
UWE	Conceptual	PIM	Static
	Navigation Space	PIM	Dynamic
	Presentation	PIM	Static
OO-H	Content	PIM	Static
	Navigation	PIM	Dynamic
	Presentation	PIM	Static
OutSystems Platform	Logic	PIM	Dynamic
	User Interface	PIM	Static
	Data Model	PIM	Static
	Business Processes	PIM	Dynamic
Tumult Hype	User Interface	PSM	Static
	Actions	PSM	Dynamic
	Timeline	PSM	Dynamic
	Properties	PSM	Static

Table 6.6: Summary of View organization and Abstraction level of the platforms surveyed in Section 2.3

Technologies	Models		Transformations		Metamodeling Languages	Target Platform(s)	Tool Support
	Abstraction Level	Modeling Language	Types	Transformation Language			
XIS-Web	PIM	XIS-Web (UML Profile)	M2M, M2T	C#, Acceleo, MTL	UML	HTML5 + JS	XIS-Web Framework: EA, EMF, Acceleo
Hera	PIM	HERA	M2T	XSL	RDF(S), RMM	HTML, WML	Hera Presentation Generator (HPG), Microsoft Visio
UWE	PIM	UWE (UML Profile)	M2M, M2T	QVT, XML	UML	Java (Servlet/JSP)	ArgoUWE, MagicUWE
OO-H	PIM	OO-H (UML Profile), PRML	M2M, M2T	OCL	UML	PHP	Visual Wade
OutSystems	PIM	OutSystems Markup Language	M2T	ND	Proprietary	HTML5 + CSS3, ASP.NET, Spring, Native Mobile Platforms	OutSystems Platform
TumultHype	PSM	ND	M2T	ND	ND	HTML5 + CSS3	Hype 3.5

Legend: ND (Not Defined/Not Relevant; Model Levels: PIM (Platform-Independent Model); PSM (Platform-Specific Model); Transformation Types: M2M (Model-to-Model), M2T (Model-to-Text)

Table 6.7: Comparison between surveyed technologies in Section 2.3

6.4 Summary

In this section we present the results of the evaluation that was performed to XIS-Web. First, there was a comparison in terms of LOC ratio between the manual implementation and the automatic generation of the case studies A and B described in Section 3.4. The overall results of these measurements were quite positive, with close to 81% of the Case Study A generated automatically and close to 70% (69.4%) of the Case Study B generated automatically. Some reasons for these values are that in both cases, some of the application's logic was carried out by an external entity. Currently logic carried out by external entities is not captured by the XIS-Web language, and so it is not possible to generate the code that corresponds to that logic. Other reason for the discrepancy in ratio, this one applicable to Case Study B, was the fact that the framework is not yet prepared to generate either the Gallery or the Dashboard UI pattern.

The second evaluation task was the assessment of XIS-Web by participants that were not involved in this research work. This was achieved by conducting a pilot-user test session where the participants followed a script that allowed them to create a web application using the XIS-Web framework, starting from scratch. The participants were then asked to fill a questionnaire that assessed three aspects of XIS-Web, namely the Language, the Framework and the General Approach. The results obtained from this evaluation were very encouraging in all three aspects. The participants found that there was room from improvement regarding the difficulty of learning the language and the usability of the Model Editor. Regarding the first issue we have plans to refactor the language, reducing the amount of concepts required to work with and for the second issue we pretend to provide better help tools to improve the creation of models using the Visual Editor for XIS-Web.

The last evaluation task was the comparison of XIS-Web with the technologies surveyed in Section 2.3. The results obtained were also encouraging, having XIS-Web being even or pulling ahead in some key aspects. These aspects were: the division of the language in six views that promoted a "separation of concerns" principle, the usage of M2M transformation for automatically creating the more complex views, the usage of latest generation web technologies that allow for the generated web application to have responsiveness regaridng the device on which is being run and finally, the fact that the language is a UML profile, making it easier on the users to learn it.

The evaluation results obtained support the thesis proposed in this dissertation, that states that a MDD-based approach can reduce the software development complexity and that HTML5 is a valid choice when building responsive web applications. Also, these results provided preliminary evidence of XIS-Web's usefulness, usability and feasibility.

Chapter 7

Conclusion

The recent developments in technology, have caused that in households and in our daily routine, Internet access is made using multiple types of devices. Nowadays a smartphone is used for many other activities beyond phone calls and a watch possesses much more capabilities than just telling time. This has changed the software development paradigm, to one in which the developers must have concerns regarding the device on which its application will be used. Fortunately, steps have been taken to ease both the complexity of software development and the fragmentation of such devices. The usage of web technologies (like the ones used in XIS-Web: HTML5, JavaScript and CSS3), Content Management Systems (CMS), cross-platform development tools or MDD-based approaches are examples of solutions focused on solving these problems. MDD approaches (like the one presented in this dissertation), mitigate these problems by putting models in the center of the development process. Everything else, like source code, or documentation can be generated automatically from those models through model transformations.

In this research work we presented XIS-Web (both the language and the framework) as an MDD-based approach that allows the development of responsive web applications and contributes to some problems of software development like complexity and device fragmentation. The XIS-Web language is a DSL defined as a UML Profile, specifically built to create responsive web applications in a platform-independent way. This language has a multi-view organization that enforces the ”separation of concerns” principle and makes use of domain specific concepts like widgets or providers. The views make the language are: Domain, BusinessEntities, Architectural, UseCases, InteractionSpace and NavigationSpace views. XIS-Web is supporting two design approaches: the dummy approach and the smart approach as defined originally in XIS [10].

The XIS-Web approach has a companion framework that provides support to define and generate the responsive web applications. This framework is based on Sparx Systems Enterprise Architect, namely making use of its MDG Technologies and Eclipse Modeling Framework, namely through the use of the Acceleo plugin. The framework takes the specification created by the developer, using the XIS-Web language and through the application of Model-to-Model and then Model-to-Text transformations, it generates the source code for the application. The framework can be divided into four components: (1) the Visual Editor, that allows for the definition of the views, (2) the Model Validator that performs validations on the specification created using pre-defined rules, (3) the Model Generator that allows for the generation of the User-Interface Views and (4) the Code Generator that generates the application's source code. Ultimately the framework grants benefits in productivity, by using a single PIM specification of the system that can be deploy into multiple devices (responsive web application) and by avoiding the implementation of boilerplate code and the reduction of errors (by using code produced from a code

template).

XIS-Web has been developed following the Action Research Methodology and has been evaluated in order to assess its usefulness, its usability, feasibility and adequacy to the purpose of modelling responsive web applications. The evaluation process was conducted in three complementary aspects: case study comparison (manual implementation vs. automatic generation), pilot-user test session assessment and comparison with the related work.

First, a comparison between the manual implementation and the automatic generation of the case studies A and B was made. Results from this task showed that for Case Study A - TimeSlot Booking App, XIS-Web managed to generate up to 80% of the application's source code, while for Case Study B - dDocs App, the ratio of generation was of 70%. These results were considering very positive specially as a proof of concept. The reason why XIS-Web didn't succeed in generated more percentage of the case studies was mainly because of two reasons: first, the language is not yet prepared to handle logic perform by external entities, generating only a stub in that place and second, the framework doesn't have implemented the UI pattern for Dashboards or for Galleries (that affected the dDocs App).

Second, the pilot-user test session focused on the assessment of XIS-Web by participants that were not directly involved in this research work. Participants were asked to follow a script that taught them how to design and develop an application using the XIS-Web framework and in the end they filled a questionnaire. The questions in the questionnaire focused on three aspects of XIS-Web: the Language, the Framework and the General Approach.

The results obtained regarding the evaluated aspects are summarized in Table 6.5. Overall, the results gathered were very positive. The language was evaluated with 4.11, the framework was evaluated with 4.53 and the general approach got a score of 4.33 (on average, using a scale of 1-5). These results showed preliminary evidences of XIS-Web's usefulness, usability, feasibility and adequacy to modelling responsive web applications.

All of the goals established in Section 1.3 were achieved during the development of this research work.

Below, we describe the main contributions of this research work, in Section 7.1 and in Section 7.2 we present some ideas for future work.

7.1 Main Contributions

We conclude that XIS-Web provides a suitable solution for developing responsive web applications. It is so because it combines an MDD-based approach with latest generation web technologies that allow for a quicker and less error prone development of responsive web applications. By putting to use concepts that are specific to the web application domain, and giving them a graphical representation, XIS-Web not only reduces the complexity inherent to software development by introducing a new layer of abstraction but it also makes possible for non-developers like business experts or functional analysts to participate in the process. With the automatic generation of most part of a web application XIS-Web reduces the burden on developers avoiding the implementation of boilerplate code, thus reducing the costs and TTM for the application and ultimately increasing productivity [8].

This research work also provides the following contributions aligned with the research goals:

- **G1.** Design and implement a DSL that allows the development of responsive web applications.
- **G2.** Implement the tools that support such language:
  - A visual editor that allows the authoring of the application's models using a UML profile;
  - A model validator enforces the language's constraints on the user created models;

  - A model generator that automatically generates the models related with the User Interface;
  - A code generator that automatically generates the application's source code, from the existing models.

- **G3.** Evaluate the quality of the proposed system in the following ways:
  - The usability and usefulness of the language and framework through a pilot-user test session;
  - The feasibility of the system through case study applications;
  - The system's added value through comparison with related work.

7.2 Future Work

This section presents some ideas for future work and research directions that can be followed in the future regarding this research. During the development of XIS-Web some of these propositions came up, but were not considered either due to time constraints or due to their complexity. None of these future work proposals undermines the achievements of this work. They are briefly explained below:

- **Exercise the language with more complex case studies and build applications in business scenarios:** In order to better evaluate a language, one needs plenty of case studies, varying in complexity and in subject. The case studies presented in this research can only exercise and evaluate certain parts of the language and framework, thus creating the need of more case study applications, and if possible more complex ones. In order to provide more credibility to XIS-Web it would be interesting to further evaluate it using applications that are being developed in business scenarios.

- **Support incremental model and code generation:** If the user already generated code, and noted something that he wishes to change, he has to make the changes he wants to in the models and the proceed to generate new UI models and code. This is particularly cumbersome for applications of considerable size, and it would be best if XIS-Web could support incremental model and code generation to prevent this situation from happening.

- **Add new UI patterns:** Currently XIS-Web is applying the Master Detail UI pattern described in Section 2.2.2. In order to diversify the type of applications that can be generated by XIS-Web, it would be important to add generation of new UI patterns such as Dashboards, Galleries and Breadcrumbs

- **Support page hierarchy:** Currently in XIS-Web there is no notion of page hierarchy. It would be interesting to make changes to the language and to the framework in order to cater for the existence of child pages.

- **Have the generated applications be collaborative:** While WebSQL is a good solution to have the generated applications working on any device, it introduces the problem of not being collaborative applications. It would be important to introduce a new data layer that would allow users to work collaboratively (if possible maintaining the premise of the application being able to work on any device.)

- **Have a common XIS-* language/framework:** Considering that the mobile application's domain is somewhat similar to the web application's domain, both XIS-Mobile and XIS-Web could be refactored in order to create common language that could be used to specify both types of applications.

The user could then chose if he wanted it create/deploy a web application, mobile application or both. Also, the WebSQL allows the exporting of its data in JSON format, which could be used as input for XIS-Analytics, enabling Dashboards on web applications created by XIS-Web.

Bibliography

[1] A. Ribeiro, "Development of mobile applications using a model-driven software development approach," masters thesis, Instituto Superior Técnico, Lisbon, Portugal, 2014.

[2] R. L. Baskerville, "Investigating information systems with action research," *Communications of the AIS*, vol. 2, no. 3es, p. 4, 1999.

[3] A. R. da Silva, "Model-driven engineering: a survey supported by the unified conceptual model," *Computer Languages, Systems & Structures*, vol. 43, pp. 139–155, 2015.

[4] "Omg. object management group - meta object facility (mof) core specification, v2.4.2.." `http://www.omg.org/mof/`. Accessed: 2016-04-20.

[5] R. J. Mitchell, *Managing complexity in software engineering*. No. 17, IET, 1990.

[6] F. Fondement and R. Silaghi, "Defining model driven engineering processes," in *Third International Workshop in Software Model Engineering (WiSME), held at the 7th International Conference on the Unified Modeling Language (UML)*, Citeseer, 2004.

[7] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[8] S. W. Liddle, "Model-driven software development," in *Handbook of Conceptual Modeling*, pp. 17–54, Springer, 2011.

[9] M. Brambilla, J. Cabot, and M. Wimmer, "Model-driven software engineering in practice," *Synthesis Lectures on Software Engineering*, vol. 1, no. 1, pp. 1–182, 2012.

[10] A. R. Da Silva, J. Saraiva, R. Silva, and C. Martins, "Xis-uml profile for extreme modeling interactive systems," in *Model-Based Methodologies for Pervasive and Embedded Software, 2007. MOMPES'07. Fourth International Workshop on*, pp. 55–66, IEEE, 2007.

[11] D. Steinberg, F. Budinsky, E. Merks, and M. Paternostro, *EMF: eclipse modeling framework*. Pearson Education, 2008.

[12] S. J. Mellor, T. Clark, and T. Futagami, "Model-driven development: guest editors' introduction.," *IEEE software*, vol. 20, no. 5, pp. 14–18, 2003.

[13] T. Kuhn, R. Gotzhein, and C. Webel, "Model-driven development with sdl–process, tools, and experiences," in *Model Driven Engineering Languages and Systems*, pp. 83–97, Springer, 2006.

[14] A. G. Kleppe, J. B. Warmer, and W. Bast, *MDA explained: the model driven architecture: practice and promise*. Addison-Wesley Professional, 2003.

[15] R. Soley *et al.*, “Model driven architecture,” *OMG white paper*, vol. 308, no. 308, p. 5, 2000.

[16] T. Kühne, “Matters of (meta-) modeling,” *Software & Systems Modeling*, vol. 5, no. 4, pp. 369–385, 2006.

[17] D. Harel and B. Rumpe, “Modeling languages: Syntax, semantics and all that stu,” 2000.

[18] M. Mernik, J. Heering, and A. M. Sloane, “When and how to develop domain-specific languages,” *ACM computing surveys (CSUR)*, vol. 37, no. 4, pp. 316–344, 2005.

[19] M. Völter, T. Stahl, J. Bettin, A. Haase, and S. Helsen, *Model-driven software development: technology, engineering, management*. John Wiley & Sons, 2013.

[20] “Omg. united modeling language infrastructure specification, version 2.4.1.” `http://www.uml.org/`. Accessed: 2016-04-20.

[21] A. Van Deursen, P. Klint, and J. Visser, “Domain-specific languages: An annotated bibliography.,” *Sigplan Notices*, vol. 35, no. 6, pp. 26–36, 2000.

[22] “Omg: Object management group - mda (model driven architecture) guide version 1.0.1; 2001.” `http://www.omg.org/mda/`. Accessed: 2016-04-28.

[23] S. C. Johnson, *Yacc: Yet another compiler-compiler*, vol. 32. Bell Laboratories Murray Hill, NJ, 1975.

[24] K. Czarnecki and S. Helsen, “Classification of model transformation approaches,” in *Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture*, vol. 45, pp. 1–17, USA, 2003.

[25] F. Jouault, F. Allilaire, J. Bézivin, and I. Kurtev, “Atl: A model transformation tool,” *Science of computer programming*, vol. 72, no. 1, pp. 31–39, 2008.

[26] “Meta object facility (mof) 2.0 query/view/transformation(qvt) specification, version 1.1. omg.” `http://www.omg.org/spec/QVT/`. Accessed: 2016-05-02.

[27] P. Wolfgang, *Design patterns for object-oriented software development*. Reading, Mass.: Addison-Wesley, 1994.

[28] S. Jorges, “Construction and evolution of code generators,” 2013.

[29] A. F. Case, “Computer-aided software engineering (case): technology for improving software development productivity,” *ACM SIGMIS Database*, vol. 17, no. 1, pp. 35–43, 1985.

[30] M. Fowler, “Language workbenches: The killer-app for domain specific languages,” 2005.

[31] G. Costagliola, V. Deufemia, F. Ferrucci, and C. Gravino, “Constructing meta-case workbenches by exploiting visual language generators,” *Software Engineering, IEEE Transactions on*, vol. 32, no. 3, pp. 156–175, 2006.

[32] S. Erdweg, T. van der Storm, M. Völter, M. Boersma, R. Bosman, W. R. Cook, A. Gerritsen, A. Hulshout, S. Kelly, A. Loh, *et al.*, “The state of the art in language workbenches,” in *Software language engineering*, pp. 197–217, Springer, 2013.

[33] A. Ginige and S. Murugesan, “Guest editors’ introduction: The essence of web engineering—managing the diversity and complexity of web application development,” *IEEE MultiMedia*, no. 2, pp. 22–25, 2001.

[34] A. Leff and J. T. Rayfield, "Web-application development using the model/view/controller design pattern," in *Enterprise Distributed Object Computing Conference, 2001. EDOC'01. Proceedings. Fifth IEEE International*, pp. 118–127, IEEE, 2001.

[35] T. O'reilly, "What is web 2.0: Design patterns and business models for the next generation of software," *Communications & strategies*, no. 1, p. 17, 2007.

[36] A. Charland and B. Leroux, "Mobile application development: web vs. native," *Communications of the ACM*, vol. 54, no. 5, pp. 49–53, 2011.

[37] B. S. Gardner, "Responsive web design: Enriching the user experience," *Sigma Journal: Inside the Digital Ecosystem*, vol. 11, no. 1, pp. 13–19, 2011.

[38] B. Frain, *Responsive web design with HTML5 and CSS3*. Packt Publishing Ltd, 2012.

[39] B. Shneiderman, *Designing the user interface*. Pearson Education India, 2003.

[40] J. Tidwell, *Designing interfaces*. " O'Reilly Media, Inc.", 2010.

[41] A. Dix, *Human-computer interaction*. Springer, 2009.

[42] J. Fons, V. Pelechano, M. Albert, and O. Pastor, "Development of web applications from web enhanced conceptual schemas," in *Conceptual Modeling-ER 2003*, pp. 232–245, Springer, 2003.

[43] E. R. Tufte and P. Graves-Morris, *The visual display of quantitative information*, vol. 2. Graphics press Cheshire, CT, 1983.

[44] W. Schwinger, W. Retschitzegger, A. Schauerhuber, G. Kappel, M. Wimmer, B. Pröll, C. Cachero Castro, S. Casteleyn, O. De Troyer, P. Fraternali, *et al.*, "A survey on web modeling approaches for ubiquitous web applications," *International Journal of Web Information Systems*, vol. 4, no. 3, pp. 234–305, 2008.

[45] T. Isakowitz, E. A. Stohr, and P. Balasubramanian, "Rmm: a methodology for structured hypermedia design," *Communications of the ACM*, vol. 38, no. 8, pp. 34–44, 1995.

[46] F. Frasincar, G.-J. Houben, and P. Barna, "Hpg: the hera presentation generator," *J. Web Eng.*, vol. 5, no. 2, pp. 175–200, 2006.

[47] N. Koch and A. Kraus, "The expressive power of uml-based web engineering," in *Second International Workshop on Web-oriented Software Technology (IWWOST02)*, vol. 16, CYTED, 2002.

[48] I. Garrigós Fernández, J. Gómez Ortega, C. Cachero Castro, *et al.*, "Modelling adaptive web applications," 2003.

[49] A. Ribeiro, A. R. da Silva, *et al.*, "Evaluation of xis-mobile, a domain specific language for mobile application development," *Journal of Software Engineering and Applications*, vol. 7, no. 11, p. 906, 2014.

[50] M. Langlands, "Inside the oval: Use-case content patterns," tech. rep., Technical report, Planet Project, 2010.

[51] J. Vlissides, R. Helm, R. Johnson, and E. Gamma, "Design patterns: Elements of reusable object-oriented software," *Reading: Addison-Wesley*, vol. 49, no. 120, p. 11, 1995.

[52] "Omg. object constraint language (ocl), v2.4; 2014." `http://www.omg.org/spec/OCL`. Accessed: 2016-04-28.

[53] "Omg. mof model to text transformation language (mofm2t), v1.0." http://www.omg.org/spec/MOFM2T/1.0/. Accessed: 2016-04-20.

[54] J. Nielsen and T. K. Landauer, "A mathematical model of the finding of usability problems," in *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*, pp. 206–213, ACM, 1993.

Appendix A

Pilot-User Test Session Guide

User Session Guide

Pilot-User Test Session – User Guide, v1.0, 2016/May

The objective of this pilot-user test session is to perform an evaluation of XIS-Web (both language and framework) by users that are not familiarized with it, in order to detect bugs or user limitations. The evaluation method involves a very simple case study application: "The TimeSlot Booking App". The creation of the application using the XIS-Web technology by the users, will be used to assess the language usability and to study further improvements to both the language and the framework. The case study application description is as follows:

Case Study - The TimeSlot Booking App

"The TimeSlot Booking App" is an application that allows for the management of "TimeSlot" that are booked by "Students". Each "TimeSlot" has a start date and time, a duration and a name. A "TimeSlot" also contains at least one "Topic" and is associated to a "Course". The "Student" should be able to:

- Manage his current "TimeSlots";
    - Create new "TimeSlots", view or edit existing ones or remove a "TimeSlot" from the list. (Creating or editing a "TimeSlot" requires the definition of all of the "TimeSlots" attributes.
- Manage his "Courses";
    - Create new "Courses", view or edit existing ones or remove a "Course" from the list.
- Share his "TimeSlots" using an external service;

The information about Time slots should be persistent and there should be the possibility for a user to share his time slots with a remote server.

Test Conditions:

- 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 Java Runtime Environment (recommended version 7) and Sparx Enterprise Architect (version 7.5, 10 or above);**[1]
- 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);

[1] Users that are from IST can have access to free and full licensed versions of Enterprise Architect, 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.

- The session will last 50 minutes (at most).
- The user must fill a survey in the end – available at: http://goo.gl/forms/GDkiXaqASV

Instructions:

1. Download the XIS-Web EA Plugin installer from here: https://goo.gl/VwR5Vi
2. Install the XIS-Web EA Plugin. This plugin provides an extension for EA, that contains the information about the XIS-Web profile (language), its diagrams, toolboxes and project template. Functions provided by the XIS-Web framework (like validation and generations) are also provided by this plugin.
3. Open EA and create a new Project.
4. Upon the prompting of the Model Wizard window, under the "Others" option, select the "XIS-Web Framework" technology, then check "XIS-Web Framework model" option and confirm your choice. On the right side view, it should appear a Package Diagram with 6 packages (one for each view of XIS-Web).
5. In order to delete the stub diagrams created by EA, make use of the XIS-Web plugin. Go to the "Extension" menu option, "XIS-Web Plugin" option and select the "Delete Generated Models" option.
6. Begin by modeling the Domain view:
    a. Create a XisEntity named "TimeSlot" and give it the following XisEntityAttributes:
        i. "date" with type "Date";
        ii. "duration" with type "Time";
        iii. "name" with type "String";
    b. Create a XisEntity named "Topic" and give it the following XisEntityAttributes:
        i. "content" with type "String";
        ii. "title" with type "String";
    c. Create a XisEntity named "Course" and give it the following XisEntityAttribute:
        i. "name" with type "String";
    d. Connect the "TimeSlot" entity to the "Topic" entity using a XisEntityAssociation:
        i. Double click on the XisEntityAssociation, and in the "Role(s)" menu, set the SOURCE "Multiplicity" to "1..*" and the TARGET "Multiplicity" to "1".
    e. Connect the "TimeSlot" entity to the "Course" entity using a XisEntityAssociation:
        i. Double click on the XisEntityAssociation, and in the "Role(s)" menu, set the SOURCE "Multiplicity" to "1..*" and the TARGET "Multiplicity" to "1".
7. Continue by modeling the BusinessEntities view:
    a. Create the business entity "TimeSlotBE".
    b. Copy the "TimeSlot", "Topic" and "Course" entities from the Domain view and paste them as "Link".
    c. Make the following connections between the "TimeSlotBE" and the entities from the Domain view:

        i. Connect the “TimeSlotBE” to the “TimeSlot” entity using a “XisBE-EntityMasterAssociation”;
        ii. Connect it to the “Topic” entity using a “XisBE-EntityDetailAssociation”;
    d. Create the business entity “CourseBE”.
    e. Connect the “CourseBE” to the “Course” entity using a “XisBE-EntityMasterDetailAssociation”;
8. Model the Architectural view:
    a. Create a XisWebApp named “TimeSlotBookingApp”.
    b. Create a XisRemoteService named “TimeSlotSharingService” and add a XisService Method named “shareTimeSlots”; This method should return “void” because it will originate a method stub for the developer to implement.
        i. Note: The XisRemoteService is an Interface;
        ii. You will be prompted to accept the realization of the method, and should respond affirmatively.
    c. Create a “XisServer” named “TimeSlotServer”.
    d. Connect the “TimeSlotBookingApp” to the “TimeSlotSharingService” using a XisWebApp-ServiceAssociation.
    e. Connect the “TimeSlotServer” to the “TimeSlotSharingService” using a XisProvider-ServiceRealization.
9. Model the UseCases view:
    a. Create a XisActor named “Student”.
    b. Create a XisEntityUseCase named “Manage TimeSlots”; Double-click on it and select the “XIS-Web” menu (to the right) to see its tagged values:
        i. Confirm that the tagged value “type” is set to “Entity Management”.
        ii. Set all the Boolean tagged values to true (CRUD for Master, Detail and Reference).
        iii. Also set the “isStartingUseCase” tagged value to true.
    c. Create a XisServiceUseCase named “Share TimeSlots”.
    d. Connect the “Student” to the “Manage TimeSlots” use case using a XisActor-UCAssociation.
    e. Connect the “Share TimeSlots” use case to the “Manage TimeSlots” use case as an “Extension”.
    f. Copy the “TimeSlotBE” from the BusinessEntities view and paste it as “Link”.
    g. Connect the “Manage TimeSlots” use case to the “TimeSlotBE” using a XisEntityUC-BEAssociation.
    h. Connect the “Share TimeSlots” use case to the “TimeSlotBE” using a XisServiceUC-BEAssociation.
    i. Copy the “TimeSlotServer” from the Architectural view and paste it as “Link”.
    j. Associate the “Share TimeSlots” use case to the “TimeSlot Server” server with a XisServiceUC-ProviderAssociation.
    k. Create a XisEntityUseCase named “Manage Courses”; Double-click on it and select the “XIS-Web” menu (to the right) to see its tagged values:
        i. Confirm that the tagged value “type” is set to “Entity Management”.
        ii. Set the Boolean tagged values to true for the CRUD operations involving the master entity (Create, Read, Delete and Update the Master Entity).
    l. Copy the “CourseBE” from the BusinessEntities view and paste it as a “Link”
    m. Connect the “Manage Courses” use case to the “CourseBE” business entity;
10. Validate the Model:

    a. Click on the “Extension” menu option, select “XIS-Web Plugin” and click on the “Validate Model” option.
11. (If the model validation ended with no errors) Apply the Model-to-Model generation (the “smart approach”) to automatically produce the NavigationSpace view and InteractionSpace view diagrams. Otherwise fix the errors and try again.
    a. To do so, click on the “Extension” menu option, select “XIS-Web Plugin” and click on the “Generate Models” option.
    b. In the popup, select “List Menu” and click “Generate!”.
12. Check the contents of the InteractionSpace and NavigationSpace. If you want, you can perform some changes to it (optional).
13. Re-Validate the Models by repeating step 10.
14. (If the model validation ended with no errors) Apply the Model-to-Code Generation, to automatically generate the code for the web application. Otherwise fix the errors and try again.
    a. To do so, click on the “Extension” menu option, select “XIS-Web Plugin” and click on the “Generate Code” option. Choose the path on which you would like the code to be generated to.
    b. Click the “Generate!” button.
15. Check if the target generation folder contains the source code of your application.
16. Congratulations! You can now use the application you have developed with XIS-Web!
17. To use your application, go to the folder that you specified as generation target, and inside the “src” try opening one of the generated web pages. (Recommended starting point is the home screen).

Extra info available at: https://github.com/theedward/xisweb

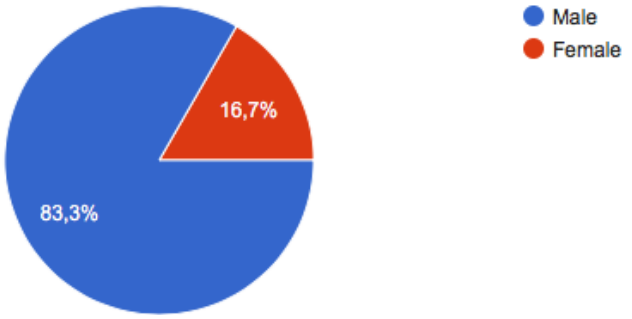
Appendix B

Results of the User Test Session's Questionnaire

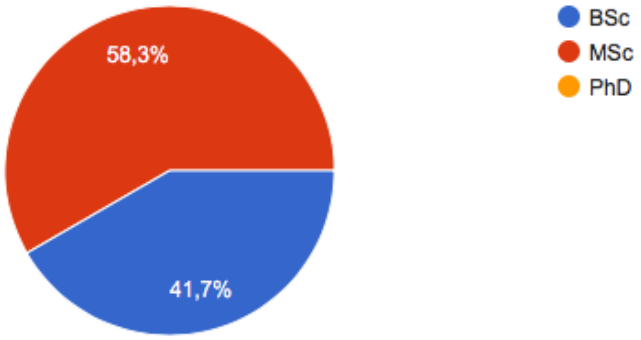
Age (12 respostas)

23
23
23
25
25
25
24
24
27
27
22
30

Gender (12 respostas)



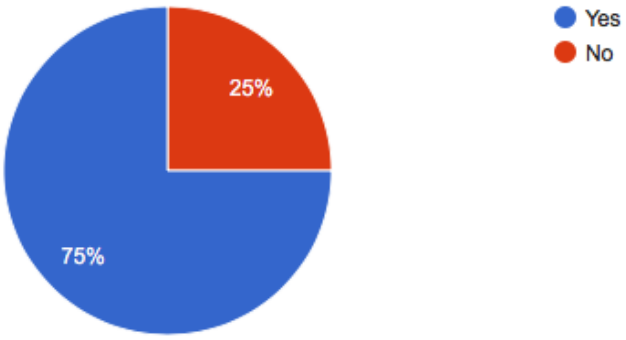
Degree (12 respostas)



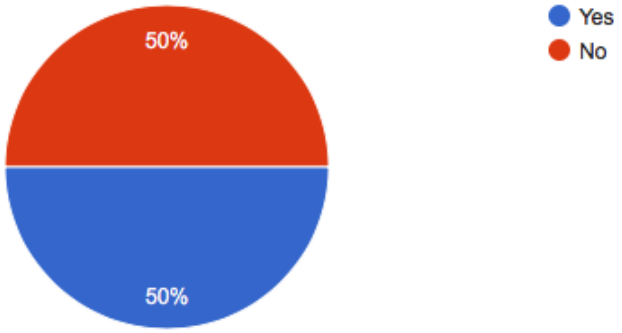
Number of Years of Professional Experience (12 respostas)

0
0
0
0
0
0
0
0
1
1
2
10

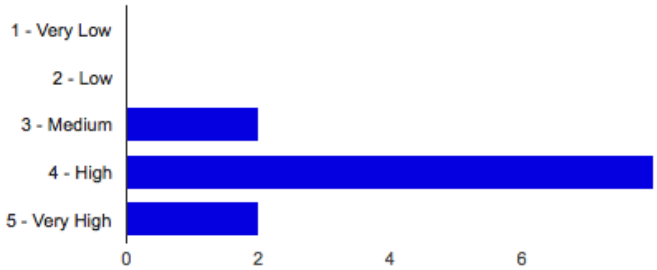
Knowledge and Previous Experience with UML (12 respostas)



Knowledge and Previous Experience with Web App Development (12 respostas)

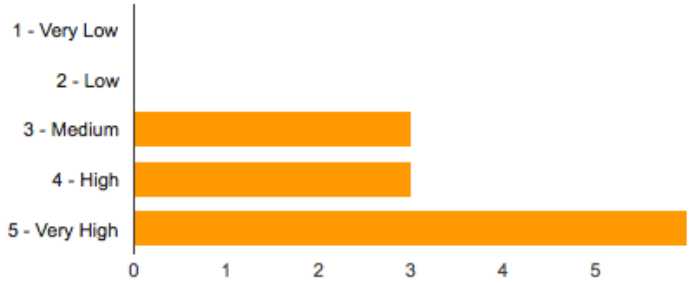


How suitable is the size (number of stereotypes) of the Language? [XIS-Web Language (UML Profile)]



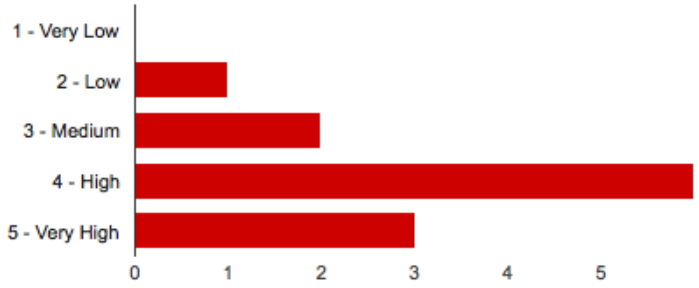
1 - Very Low	**0**	0%
2 - Low	**0**	0%
3 - Medium	**2**	16.7%
4 - High	**8**	66.7%
5 - Very High	**2**	16.7%

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



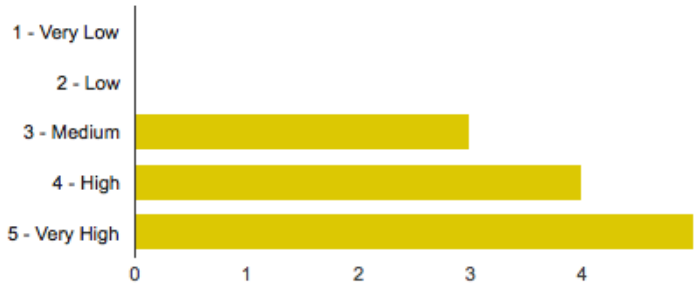
1 - Very Low	**0**	0%
2 - Low	**0**	0%
3 - Medium	**3**	25%
4 - High	**3**	25%
5 - Very High	**6**	50%

How easy to learn is the language without the UI concepts? [XIS-Web Language (UML Profile)]



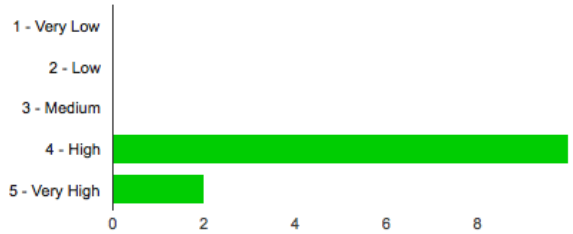
1 - Very Low	0	0%
2 - Low	1	8.3%
3 - Medium	2	16.7%
4 - High	6	50%
5 - Very High	3	25%

How easy to learn is the language with the UI concepts? [XIS-Web Language (UML Profile)]



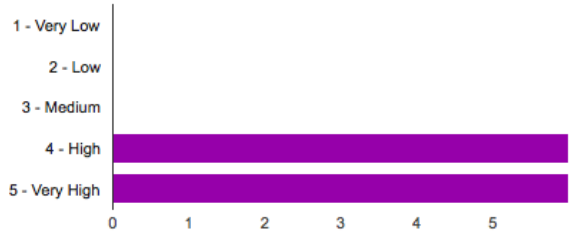
1 - Very Low	0	0%
2 - Low	0	0%
3 - Medium	3	25%
4 - High	4	33.3%
5 - Very High	5	41.7%

How suitable is the language for the Web Apps development domain? [XIS-Web Language (UML Profile)]



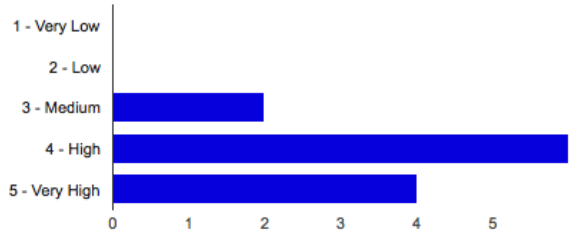
1 - Very Low	0	0%
2 - Low	0	0%
3 - Medium	0	0%
4 - High	10	83.3%
5 - Very High	2	16.7%

How do you rate the usability of the XIS-Web plugin in EA? [XIS-Web Framework (EA Development Environment)]



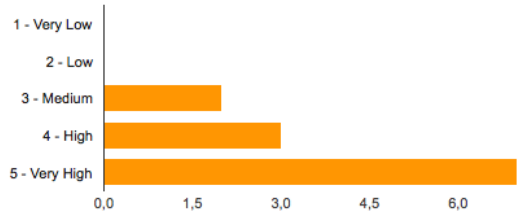
1 - Very Low	0	0%
2 - Low	0	0%
3 - Medium	0	0%
4 - High	6	50%
5 - Very High	6	50%

How do you rate the usability of the Visuall Editor? [XIS-Web Framework (EA Development Environment)]



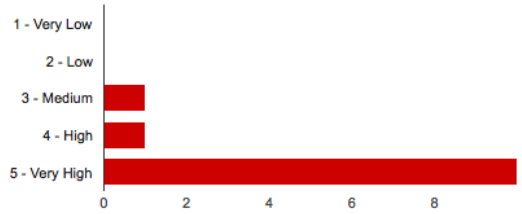
1 - Very Low	0	0%
2 - Low	0	0%
3 - Medium	2	16.7%
4 - High	6	50%
5 - Very High	4	33.3%

How do you rate the usefulness of the Model Validator? [XIS-Web Framework (EA Development Environment)]



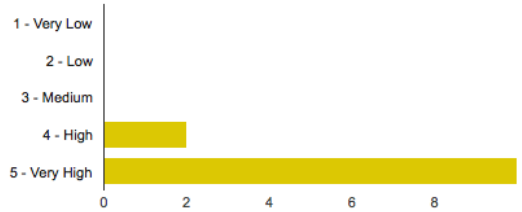
1 - Very Low	0	0%
2 - Low	0	0%
3 - Medium	2	16.7%
4 - High	3	25%
5 - Very High	7	58.3%

How do you rate the simplicity of the Model-to-Model transformation process? [XIS-Web Framework (EA Development Environment)]



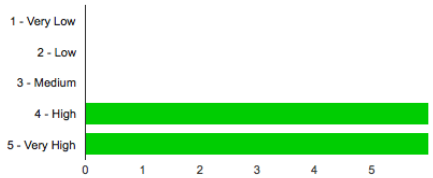
1 - Very Low	0	0%
2 - Low	0	0%
3 - Medium	1	8.3%
4 - High	1	8.3%
5 - Very High	10	83.3%

How do you rate the simplicity of the Model-to-Text transformation process? [XIS-Web Framework (EA Development Environment)]



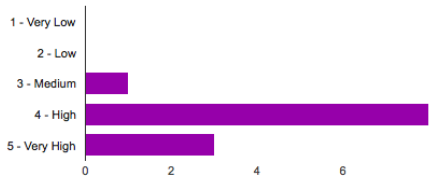
1 - Very Low	0	0%
2 - Low	0	0%
3 - Medium	0	0%
4 - High	2	16.7%
5 - Very High	10	83.3%

How do you rate the productivity with XIS-Web comparing to the traditional software development process? [XIS-Web Framework (EA Development Environment)]



1 - Very Low	0	0%
2 - Low	0	0%
3 - Medium	0	0%
4 - High	6	50%
5 - Very High	6	50%

How likely would you use such a tool on your own Web App projects? [XIS-Web Framework (EA Development Environment)]



1 - Very Low	0	0%
2 - Low	0	0%
3 - Medium	1	8.3%
4 - High	8	66.7%
5 - Very High	3	25%