A Model-Driven Development Approach for Responsive Web Applications: The XIS-Web Technology

João Seixas
INESC-ID/Instituto Superior Técnico, Universidade de Lisboa
Lisbon, Portugal
joao.eduardo@tecnico.ulisboa.pt

***Abstract*—Nowadays users use multiple devices (e.g., mobile, laptops, watches, automobiles) to access a myriad of Web applications. This situation has increased the importance of developing and maintaining such applications in a responsive way, i.e. with the ability to seamlessly display their contents on multiple devices with different screen of any given size. This paper proposes the XIS-Web technology which is a model-driven approach focused in the development of such "responsive web applications". XIS-Web technology includes two main components: (i) the XIS-Web modeling language (implemented as a UML profile) and (ii) the XIS-Web framework, which is a set of software tools that support its approach. On the other hand, XIS-Web stands out in four key aspects: separates the modeling of a web application in six views, which ultimately promotes the separation of concerns that is key to managing complexity; generates the interaction spaces and the navigation between them, relieving this cumbersome task from the user; employs latest generation web technologies (such as HTML5, JavaScript, CSS) that allow the required flexibility of developing responsive web applications; and, as this language is defined as a UML profile, it allows the creation of platform-independent models without requiring a significant learning curve.**

Keywords—model-driven development; platform-independent model; web applications; domain specific language.

I. INTRODUCTION

Data from 2015 shows that there were about 3300 million Internet users in the world and that the tendency is for this number to keep on rising[1]. Statistics from 2014 show that in the US 90% of the households have at least three devices connected to the Internet and even the device paradigm is shifting, people browse the web not only on their desktop or mobile but also on their TVs, "Wearables" (like watches) or even their appliances. The fragmentation of devices capable to accessing the Web has increased the importance of developing responsive content, that can seamlessly be displayed on a screen of any given size. As an effect, software complexity increased over the years, due to having to design the same application several times, in order to run properly on any device or platform. Several approaches to solve this problem have been proposed, namely Content Management Systems (CMS) like Wordpress[2] or Drupal[3] and the new generation of Web languages like HTML5 and CSS3. Both these approaches address the issue by allowing flexibility in the User Interface, having it scale or even change, according to the size and shape of the device in which is being displayed. Designing responsive web applications, even with the mentioned approaches, still requires technical and programming skills. Therefore, it is useful to define an abstraction layer on the top of these software frameworks, allowing non-technical stakeholders to participate in the design and development of these apps. Model-Driven Development (MDD) is a commonly used approach to abstract the complexity of developing software based on models that represent the structure and behavior of such apps [3][4]. Consequently, MDD approaches enable business and technical specialists to work together and ultimately may result in better communication, higher productivity and a shorter time to market (TTM) [5].

This paper describes XIS-Web, a concrete MDD approach particularly focused in the development of responsive web applications. XIS-Web is based on previous work, namely it reuses and adapts some concepts of the XIS [6] and XIS-Mobile [7][8] approaches.

XIS proposed a MDD approach for designing web and desktop interactive systems at a Platform-Independent Model (PIM) level, using a Domain Specific Language (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, UseCases 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 the 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 UseCases 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

[1] http://www.internetlivestats.com
[2] https://wordpress.org
[3] https://www.drupal.org

with the system. Third, the User-Interfaces view comprises the NavigationSpace and InteractionSpace views. The NavigationSpace view 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. XIS also proposes two modeling approaches: the smart approach and the dummy approach [6]. In the smart approach, the designer only needs to define the Domain, BusinessEntities, Actors and UseCases views, and based on a predefined set of UI patterns, the User-Interfaces views are automatically generated through Model-to-Model (M2M) transformations. Then it is possible to refine these generated UI models through direct authoring or design. On the other hand, in the dummy approach, the designer needs to manually define all the views, what is considered cumbersome and time-consuming.

More recently, the XIS-Mobile approach was defined with the focus on developing cross-platform mobile applications [7][8]. 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 applications scenarios. The XIS-Mobile language is organized in six views: Domain, BusinessEntities, UseCases, InteractionSpace, NavigationSpace and Architectural. While the first four views share the same constructs as in XIS (however 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 Model-to-Text (M2T) transformations.

XIS-Web technology includes both a DSL and a companion framework tool support. The DSL is defined as a UML profile [6] and provides the necessary concepts for web application modeling. It is built on top of Sparx Systems Enterprise Architect (EA)[4] and Eclipse Modeling Framework (EMF)5 for the M2M and M2T transformations, respectively. XIS-Web reuses the methodology proposed by XIS and some of its concepts, redefining them in order to better model web applications. Given that XIS-Mobile is a successful implementation of the XIS methodology the supporting technologies are the same in this approach. Considering that both mobile and web apps have overlapping matters, some of XIS-Mobile stereotypes are reused. XIS-Web has a much broader target audience than the one addressed by XIS-Mobile. Not only mobile devices can have access to web applications, but virtually any device that has web browsing capabilities. A key difference between XIS-Web and XIS-Mobile is that it generates a single software artifact that is platform and device independent. Comparing to a mobile native approach, the use of a web-based approach, like the one proposed in this paper, can effectively reduce maintenance costs, given that only one codebase has to be maintained. Moreover, in mobile applications where performance is not a critical issue, web applications have all the advantage [1]: Access to the app is instant, while native applications have to be downloaded from the respective store; User Interface is responsive and if needed can be exactly the same as a native application; Most hardware sensor is accessible via an API; Web applications also have offline support; and Monetizing the web application comes without strings attached to the respective store.

The paper is organized in seven sections. Section II provides a detailed analysis of the key features of the language. Section III describes the framework tool support. Section IV introduces and discusses the related work. Section V presents and discusses evaluation results. Finally, Section VI summarizes the key points and refers some open and near future work.

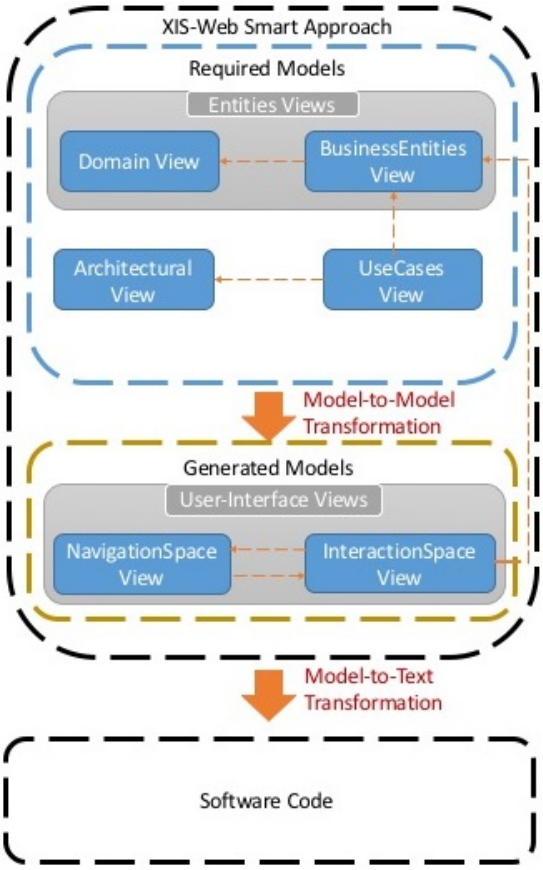

Fig. 1. Dependencies between the different XIS-Web views (adapted from [8]).

[4] http://www.sparxsystems.com.au/products/ea/

[5] https://eclipse.org/modeling/emf/

II. LANGUAGE OVERVIEW

The XIS-Web language takes advantage of its heritage by reusing the methodology and design approaches proposed by XIS, and also by refactoring some concepts present on the XIS-Mobile language and using its multi-view organization. Simultaneously, the XIS-Web language introduces some new concepts of its own to properly model the major aspects of web applications. Although the views maintain the same goal and detail the same viewpoints as proposed in XIS-Mobile, some of their inner concepts (stereotypes and tagged values) are new for adding a different ability to express concepts commonly present in web applications. For instance, there are stereotypes with different options, like in the Architectural View where a "XisInternalService" can be a webcam or microphone, and a "XisRemoteService" can be a JavaScript library. Moreover, the InteractionSpace View is structurally different. While it contains building blocks common to both mobile and web applications such as buttons, labels or textboxes, there are some that only make sense in the web application domain (e.g. IFrames, embedded HTML and some types of input controls) (see Fig.1).

The User-Interfaces View contains the key models when it comes to the definition of the web application's appearance and behavior. It comprises the InteractionSpace and NavigationSpace views. The InteractionSpace describes the structure and layout of each web page or screen, here designated as interaction space, while the NavigationSpace details the hierarchy and flow between each interaction space. Below we provide a more detailed explanation of each one of these views.

A. NavigationSpace View

The NavigationSpace (NS) View is one of the simplest (given the number of stereotypes present), yet most fundamental views of XIS-Web. This view adds an abstraction level to the application, by allowing the user to detail the flow and hierarchy of the different interaction spaces. The stereotypes present in this view are: the XisInteractionSpace, which corresponds to an interaction space of the application, and the XisInteractionSpaceAssociation, which represents the navigation or transition between the interaction spaces. Each XisInteractionSpaceAssociation contain the information of the action name that caused the navigation. There could be several actions that can cause navigation to a given InteractionSpace.

B. InteractionSpace View

The InteractionSpace (IS) View is the view that has the highest number of stereotypes available, hence the most complex one. The majority of the stereotypes for this view are the widgets usually present in the UI of web applications. The IS view represents the UI layout, the events that some widgets can trigger and the gestures used to interact with the application (if the device has a touch screen). The modeling of this view is done via a Composite Class Diagram. The process starts with the creation of a XisInteractionSpace, a class representing the screen, that should contain one or more XisWidgets, classes representing the UI widgets or controls. A Business Entity (BE) is connected to the interaction space through a XisDomainAssociation, this define the domain entities manipulated in the context of this interaction space.

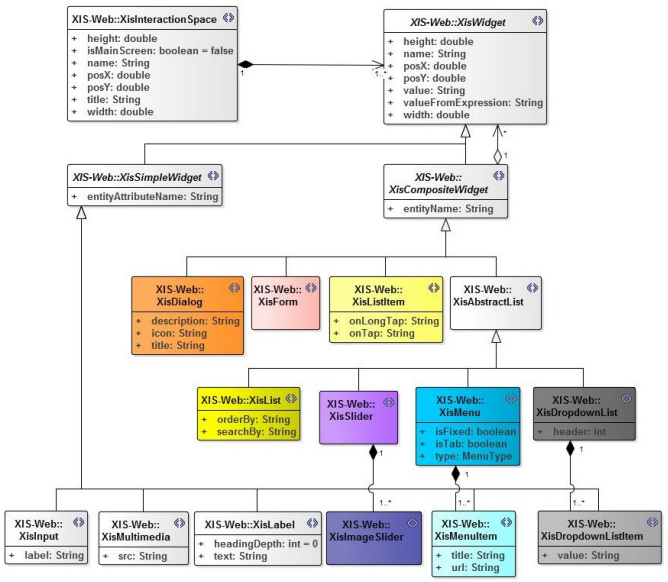

Fig. 2. InteractionSpace View metamodel (partial view).

Each XisWidget can either be: (1) XisCompositeWidget, a container widget that groups other XisWidgets (this stereotype follows the Composite design pattern); or (2) XisSimpleWidget, which represents the set of simple controls, i.e., controls that do not contain other widgets. Every XisWidget must have a value, which is defined using the tagged value "value" that can either be bound to a domain entity's attribute value or to a constant value. In the first case, this tagged value is filled following the expression: <EntityName>.<AttributeName>, where EntityName corresponds to the name of an entity belonging to the business entity associated to the widget's interaction space, and AttributeName to an attribute of that entity. Below, we detail XisSimpleWidget and XisCompositeWidget stereotypes.

XisSimpleWidget. Most of the XisSimpleWidgets are specializations of two abstract stereotypes: XisInput and XisMultimedia. XisInput is an abstraction of the input types that are normally available in web applications. In the XIS-Web language we consider the following XisInput specializations: XisTextBox, XisCheckBox, XisRadioButton, XisButton, XisLink, XisDatePicker and XisTimePicker. There is also the XisInputCustom stereotype that allows the user to model another type of input which is not defined in the language. Regarding the XisMultimedia, it is specialized in the stereotypes: XisAudio, XisImage, XisVideo, XisMap, XisEmbed and XisIFrame. The remaining types of XisSimpleWidgets are the XisLabel and the XisMenuItem.

XisCompositeWidget. We divided the composite widgets in two groups: the ones that are lists and the others. The XisAbstractList is an abstract stereotype that aggregates the lists. It specializes in: XisList, XisMenu, XisSlider and XisDropdown. A XisList can be seen as the ordinary list of items. To represent that, we have defined that a XisList can only contain XisListItems, and since these are also composite widgets they can have any simple widget inside. This allows the user to have any custom type of list.

Each screen has its own menu, providing the actions that the user can perform on it. This is accomplished via the XisMenu stereotype that is composed of XisMenuItem. XisSlider is different from the previous ones in the sense that is also an abstract stereotype. It is a generalization of the XisImageSlider that aggregates XisImages. The last XisAbstractList is the XisDropdown, which is much like a XisMenu with the difference that it can only contain one or more XisLabels.

Other types of composite widgets are the XisForm, XisDialog and XisVisibilityBoundary. A XisForm represents the form element that is present in web pages. A form typically is made of labels followed by an input field, making a XisForm having XisLabels followed by XisInputs inside. The XisDialog stereotype corresponds to the alert dialog in a web page. A XisVisibilityBoundary is a stereotype that allows the user to define different views inside the same screen.

C. Design Approaches

Like XIS, XIS-Web proposes two modeling approaches that leverage model transformations, the "dummy" and "smart" approaches.

Using the "dummy" approach, the user should define all views (the Architectural View is optional, since there may not be interactions with external entities) including the NS and IS views. This approach is desirable if the user wants highly customized interaction spaces and to have full control of the model design. When the model is finished, the user can trigger the generation of source code, through a M2T transformations.

Taking the models defined by the user and applying M2M transformations, the "smart" approach automatically generates the User-Interface views. Currently XIS-Web is generating interaction spaces applying the well-known "Master-Detail" UI design pattern [11]. Following the guidelines provided by this pattern we have: each BE has two interaction spaces, the MasterIS, where all of its instances are displayed in a list with bulk actions (CRUD operations) and the DetailIS, where the different attributes of the BE are viewed inside a form element. The configuration of M2M transformations and the application of this UI design pattern is done in the UseCases View. A XisUseCase abstracts the pattern, via its tagged values that define the CRUD operations that will be available in the generated interaction spaces. Interaction spaces have a common infrastructure, consisting of site logo, site map, followed by the content of the page and finishing with a footer. If it is a MasterIS the menu containing the action is located before the content of the page, in the case of a DetailIS the actions are located bellow the content. In the DetailIS, the input fields present in the form, are directly derived from the attributes present in the Master entity.

TABLE I. XISUSECASE TAGGED VALUES

XisUseCase tagged values	
Name	***Possible Values***
isStartingUseCase	True, False
Type	EntityManagement, EntityConfiguration
CRUD Master	True, False
CRUD Detail	True, False
CRUD Reference	True, False

III. FRAMEWORK DESCRIPTION

Having a DSL to describe web applications is useful, but the great advantage of using the XIS-Web language is the fact that it has a framework that supports an MDD-based approach to ease the development process. The framework relies in Sparx Systems Enterprise Architect (EA) as main environment, and takes advantages of its Model Driven Generation (MDG) Technologies in order to validate and apply M2M transformations. It also uses the Acceleo[6] plugin present in the Eclipse Modeling Framework (EMF) to perform the M2T transformations. To provide a better understanding of the M2M and M2T transformations applied by the framework, a more detail explanation is given bellow.

Model-to-Model Transformations (M2M). To correctly use the XIS-Web stereotypes in each diagram (in EA), the Visual Editor was implemented through an MDG Technology plugin. This is fully compliant with the OMG specification for UML2, and so it has a very good support for UML profiles. It allows for the creation of toolboxes, diagrams, project templates and patterns customized for the profile being developed. Provided that users can make mistakes, a model validation was implemented to make sure that the models are suitable for further generation. This avoids the burden of having to correct mistakes further down the development process chain and improves the overall quality of the models and the code itself. This validation is done using the Model Validation API provided by EA, which allows the definition of custom error messages, levels of severity to each rule and the immediate navigation to the element in fault. Once the model is validated, the User-Interface views can be generated. The M2M transformation is implemented using EA's Automation Interface, which offers an API for retrieving and managing data (e.g. elements, relationships) contained in the EA repository.

Model-to-Text Transformations (M2T). The M2T generator is based on Acceleo. It is a template-based code generator framework that implements the MOF MTL (Model to Text Language) and it is compatible with any kind of EMF model. Typically, the code templates have a static part (regular text) and a dynamic part that changes in function of the model (Acceleo annotations). Currently in XIS-Web, the code generation of a web application is structured in three parts: Content, Hypertext/Application and Presentation. For the Content layer, WebSQL[7] was the technology of choice due to its simplicity and seamless integration with the other technologies. The Hypertext/Application layer is implemented using two

[6] http://www.eclipse.org/acceleo

[7] https://www.w3.org/TR/webdatabase/

technologies that complement each other, HTML5 and JavaScript. Ultimately, for the Presentation layer, we used CSS3[8] as the main technology, namely applying the Boostrap[9] library which is the most popular framework for developing responsive web applications.

The development process of an application using the XIS-Web framework consists in four steps: (1) modeling the required views, using the model editor in EA, (2) validating the views by running a set of rules, (3) generating the User-Interfaces View through M2M transformations by leveraging EA's MDG Technologies (assuming "smart" approach) and (4) generating the source code.

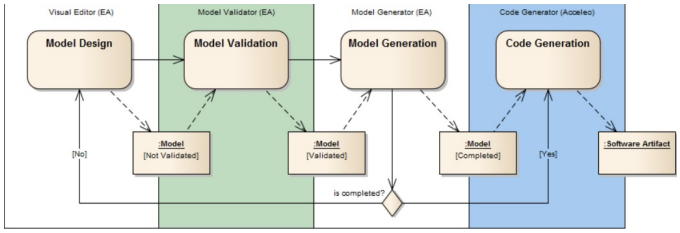

Fig. 3. XIS-Web technologies and development process (extracted from [8]).

IV. Related Work

In this section, we present a survey of the most known technologies that describe modeling languages with MDD frameworks. [14].

Hera. Based on Relationship Management Methodology (RMM) [13] it was first introduced in 2000 and proposes models for Content, Hypertext/Application and Presentation viewpoints. To model the Content, it uses the Resource Description Framework (Schema) - RDF(S). Hypertext and Presentation level modeling are very much like RMM's graphical notation, being mainly based by slices and slice relationships that can either be compositional or navigational [12][14]. Hera is supported by the Hera Presentation Generator (HPG) and this 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 can also verify the model's consistency. From these models it is possible to create a suitable static presentation in languages like HTML, WML.

UML-based Web Engineering (UWE). UWE is a framework that has been continuously being developed since 1998 [15][14]. This framework proposes three viewpoints: Content View, NavigationSpace View and Presentation View. 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. ArgoUWE is the free support tool for UWE. During modeling, it 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. UWE uses M2T transformation rules in ATL to translate its Content and Presentation models into Java Beans and JSPs.

Object-Oriented Hypermedia method (OO-H). The OO-H method first appeared in 2000, being the most recent publications from 2005 and 2007 [14][16]. The language is defined as a UML Profile and it is designed for web application development. Like UWE, standard UML class diagrams are used to model the Content View. In the Navigation View, Navigation Access Diagrams (NAD) capture navigation paths and services that the users can activate (similar to XIS-Web's NavigationSpace View). In the Presentation View, having a NAD as a base model, an Abstract Presentation Diagram (APD) can be generated by using NAD2APD (which can be abstractly described as M2M transformations) mapping rules. This APD can be interpreted as a sitemap of the web application because it consists of a set of pages and their links. This APD can then be transformed via M2T transformations in order to generate the web application artifact (in PHP). OO-H relies on VisualWade, a commercial tool, to help with the modeling in general (edition and validation) and to apply the mentioned transformations.

OutSystems Platform[10]. OutSystems provides a rapid application development (RAD) Framework that enforces the Model-Driven Development approach. The platform is for commercial use (with trial licenses available) and it stands out for its very large user base. The development of an application is done using the PIMs for the Logic, User Interface, Data Model and Business Process. Besides generating web applications in HTML5 and CSS3, it also does it with Spring or ASP.NET. This is also a very known framework for generating native mobile applications.

Tumult Hype[11]. Hype 3.5 is the newest version of Tumult's Hype Platform. It is a proprietary software (with a free trial available) that takes a different approach at solving the same problem (rapid web application development), when in comparison to the other frameworks. The main difference of this platform, is that the only the modeling is focused on the user interface. The user populates a screen with the elements that take part in it, applies the logic using either the Actions, Timeline or Properties views (that can either be animation or flow control) and the platform generates the correspondent code. While the other surveyed methods or frameworks were more intended to generate a "form heavy" web application, Hype3.5 is more focused on infographics, presentations and web advertisement.

V. Evaluation

XIS-Web was evaluated in three different ways. First, the case studies for this evaluation are described in detail in Section A. Then, as described in Section B, the evaluation of XIS-Web's M2T capabilities is done via the comparison between the manual implementation and automatic generation of the code for two case studies – the dDocs App and the TimeSlotBookingApp. These case study applications will also be described in that section. Second, Section C 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 of web applications and its learning

[8] https://developer.mozilla.org/en-US/docs/Web/CSS/CSS3
[9] http://getbootstrap.com
[10] https://www.outsystems.com/platform
[11] http://tumult.com/hype

curve), (2) the Framework (namely the Visual Editor, the Model Validator, Model Generator and Code Generator) and (3) the General Approach. Finally, Section D presents a comparison between XIS-Web and the technologies from the related work.

A. Case Studies Description

Case Study A – The TimeSlotBooking 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.
- 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 – The 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 view 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. (see http://ddocs.eu).

B. Code Generation vs Manual Implementation of Case Studies

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[12].

TABLE II. RATIO BETWEEN GENERATED CODE AND MANUALLY IMPLEMENTED CODE, PER LANGUAGE AND TOTAL

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 %	979	464	58.2 %
Total	675	546	80.8 %	1127	783	69.4 %

The results obtained were considered positives because they met one of the goals set for this dissertation (G4) that was: Automatically generate more than 70% of an application's source code. Regarding the Case Study A (Time Slot Booking App), the average ratio of 80.8% was due to 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 because the language does not capture the intents and logic that are executed by a WebService (a XisServer). For Case Study B (dDocs App), the results are not as good as for Case Study A, with an average 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 f the documents, the Dashboards containing relevant data from documents and the communication with a third party repository to backup the documents.

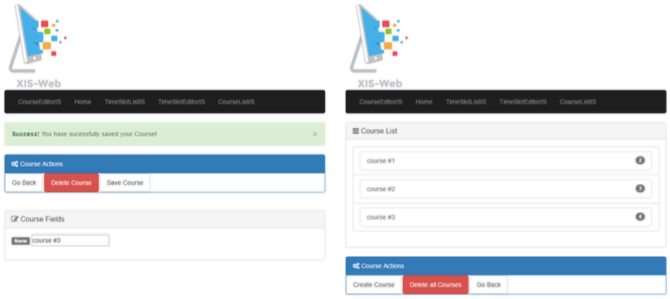

Fig. 4. Result of generating two interaction spaces from the TimeSlot Booking App.

C. User Session Evaluation

The participants of this session were people not directly involved in the research work and its main goal was the detection of potential bugs and user limitations. The group had 12 elements, with ages ranging from 23 to 30, with at least a Bachelor of Science degree. Three of the participants had no previous experience with UML, half of the participants had also

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

no experience with web application development and 4 participants had professional experience in Information Technology. The participants received a 10-minute presentation explaining the fundamental concepts of the XIS-Web language and its framework. They were also asked to follow a script that described the TimeSlot Booking App and were asked to design this application using XIS-Web. The average time for the 12 participants was of 38 minutes. In the end, the participants were asked to fill a questionnaire to rate the XIS-Web language, framework and the general approach. The results and analysis of the questionnaire's results is described below.

TABLE III. AVERAGE SCORE (IN A SCALE OF 1-5) FOR EACH OF THE SURVEYED XIS-WEB ASPECTS

XIS-Web			
Average	Language	Framework	General Approach
	4.1	4.53	4.33

Overall, as Table III 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 it is something we pretend to change in future work, by refactoring 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 the associated 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 [19]. 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.

D. Comparison with Frameworks from the Related Work

In these tables we present a summary of the surveyed frameworks. XIS-Web, when comparing to the other surveyed frameworks, stands out in four key aspects: (1) Separates the modeling of a web application in six views, which ultimately promotes a separation of concerns that is key to managing complexity in software development, (2) Generates the interaction spaces and the navigation between them, relieving this cumbersome task from the user, allowing a quicker TTM, (3) It employs latest generation web technologies that allow the required flexibility of developing a responsive web application and (4) The language is described as a UML profile, which is a standard in modeling that allows the creation of PIM thus lowering its learning curve.

TABLE IV. SUMMARY OF VIEW ORGANIZATION AND ABSTRACTION LEVEL OF THE SURVEYED PLATFORMS.

Modeling Languages			
Name	**Viewpoint**	**Abstraction**	**Perspective**
XIS-Web	Domain View	PIM	Static
	BusinessEntities View	PIM	Static
	Architectural View	PIM	Static
	UseCases View	PIM	Dynamic
	NavigationSpace View	PIM	Static
	InteractionSpace View	PIM	Static
Hera	Conceptual View	PIM	Static
	Application View	PIM	Dynamic
	Presentation View	PIM	Static
UWE	Conceptual View	PIM	Static
	Navigation Space View	PIM	Dynamic
	Presentation View	PIM	Static
OO-H	Content View	PIM	Static
	Navigation View	PIM	Dynamic
	Presentation View	PIM	Static
Outsystems	Logic View	PIM	Dynamic
	User Interface View	PIM	Static
	Data Model View	PIM	Static
	Business Processes View	PIM	Dynamic
Tumult Hype	User Interface View	PSM	Static
	Actions View	PSM	Dynamic
	Timeline View	PSM	Dynamic
	Properties View	PSM	Static

TABLE V. COMPARISON BETWEEN SURVEYED APPROACHES.

Model-Driven Approaches	Models		Transformations		Metamodeling languages	Target Platform(s)	Tool Support
	Levels	Language	Types	Languages			
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	VisualWade
OutSystems	PIM	OutSystems Markup Language	M2T	ND	Proprietary	HTML5 + CSS3, ASP.NET, Spring, Native Mobile Languages	OutSystems Platform
Tumult Hype	PSM	ND	M2T	ND	ND	HTML5 + CSS3	Hype 3.5

Legend: ND (Not defined/ Not Relevant); Models Levels: PIM (Platform Independent Model); PSM (Platform Specific Model); Transformations Types: M2M (Model-to-Model), M2T (Model-to-Text).

VI. CONCLUSION

The XIS-Web Technology, presented in this paper, aims to be a solution when it comes to frameworks that generate responsive web applications. It gives the user the concepts required to model a web application, while promoting a "separation of concerns" principle by being divided in six views, which is a very important feature when handling software complexity. With the use of M2M transformations it ultimately meets some principles proposed by MDD like increase in productivity and reduction of errors. Considering the M2T transformations that are achieved with the aid of the supporting tools (EA and Acceleo), XIS-Web can reduce maintenance overhead and increase TTM by generating one software specification that due to being responsive can run properly on virtually any device.

Here are some ideas for future work:

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 subject. The case studies presented in this research can only exercise and evaluate certain parts of the framework.

Addition of new UI patterns. Currently XIS-Web is applying the Master Detail UI pattern. In order to diversify the type of applications that can be generated by XIS-Web it would be important to add the generation of new UI patterns like Breadcrumbs, Galleries or Dashboards.

Have collaborative applications. WebSQL allows applications to run on any device, but limits the collaboration of multiple users. It would be interesting to develop applications using a different data layer that could possibly provide both features.

Have a common XIS-* language/framework. Considering that the mobile and web application domain are quite similar, both XIS-Web and XIS-Mobile could be refactored in order to create a common language that could be used to specify both types of applications. Also, the inclusion of XIS-Analytics in this project would allow for the creation of the Dashboard pattern in web/mobile applications.

REFERENCES

[1] A. Charland, B. Leroux, "Mobile application development: web vs. Native", in Communications of the ACM, 54(5), pp. 49-53, 2011.

[2] H. Heitktter, S. Hanschke, T. A. Majchrzak, " Evaluating cross-platform development approaches for mobile applications", Web information systems and technologies, pp. 120-138, Springer, 2012.

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

[4] C. Atkinson, T. Khüne, "Model-driven development: a metamodeling foundation", pp. 36-41. Software, IEEE, 20(5), 2003.

[5] B. Selic, "The pragmatics of model-driven development", IEEE Software, 20(5), 19, 2003.

[6] A. Silva, J. Saraiva, R. Silva, C. Martins, "XIS-UML Profile for eXtreme Modeling Interactive Systems". MOMPES, IEEE, 2007.

[7] A. Ribeiro, A. R. da Silva, "Evaluation of XIS-Mobile, a Domain Specific Language for Mobile Application Development", Journal of Software Engineering and Applications, 7(11), Scientific Research Publishing, 2014.

[8] A. Ribeiro, A. da Silva, "XIS-Mobile: A DSL for Mobile Applications", in Proceeding of SAC, ACM, 2014.

[9] Object Management Group: Object Contraint Language (OCL) Specification. Accessed on March 2015. http://www.omg.org/spec/OCL

[10] Object Management Group: MOF Model To Text Transformation Language (MOFM2T). Accessed on March 2015. http://www.omg.org/spec/MOFM2T/1.0

[11] B. Scott, T. Neil, "Designing web interfaces: Principles and patterns for rich interactions". O'Reilly Media, Inc., 2009.

[12] F. Frasincar, G-J. Houben, P. Barna, "HPG: the Hera presentation generator" in Journal of web Engineering, 5(2), pp. 175-200, 2006.

[13] T. Isakowitz, E. A. Stohr, P. Balasubramanian, "RMM: a methodology for structured hypermedia design", in Communications of the ACM, 38(8), pp. 34-44, 1995.

[14] W. Schwinger et al, “A survey on web modeling approaches for ubiquitous web applications”. IJWIS, 4(3):234305, 2008.
[15] N. Koch, A. Kraus, “The expressive power of uml-based web engineering”, Proc. of the 2nd International Workshop on Web-oriented Software Technology (IWWOST 2002), pp. 21-32, 2002.
[16] I. Garrigos, J. Gomez, C. Cachero, “Modelling adaptive web applications”, Proc. of the IADIS International Conference WWW/Internet, pp. 813-6, 2003.
[17] C. Crumlish, E. Malone, “Designing social interfaces: Principles, patterns, and practices for improving the user experience”. O’Reilly Media, Inc., 2009.
[18] G. Kappel, B. Prll, S. Reich, W. Retschitzegger, “ Web engineering”. John Wiley & Sons, 2006.
[19] J. Nielsen, 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.