A Model-Driven Approach for Developing Responsive Web Apps

João Seixas, André Ribeiro and Alberto Rodrigues da Silva
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal

Keywords: Model-Driven Development, Responsive Web Application, Web Engineering.

Abstract: Nowadays users have multiple devices to access a myriad of web and mobile applications. This has increased the importance of developing such applications in a responsive way, i.e. with the ability to seamlessly display their contents on multiple devices. This paper proposes the XIS-Web technology as a model-driven approach focused in the development of responsive web applications. XIS-Web technology includes two main parts: the XIS-Web modeling language, implemented as a UML profile; and the XIS-Web framework, which is a set of integrated software tools. XIS-Web stands out in four key aspects: supports the modeling of web applications around six viewpoints, which ultimately promotes the separation of concerns that is key to managing complexity; generates user-interface models from extended use-case models, relieving this cumbersome and time consuming 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 allows the creation of platform-independent models without requiring a significant learning curve. This paper also presents an evaluation conducted in a controlled environment with a group of independent users, and briefly introduces simple case studies.

1 INTRODUCTION

The fragmentation of devices capable to accessing the Web has increased the importance of developing responsive applications. 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 (Charland and Leroux, 2011; Heitktter et al., 2012). Several Web Engineering approaches (Kappel et al., 2006; Schwinger et al., 2008; Wakil and Jawawi, 2017) to solve these problems have been proposed, namely by adopting new generation of Web languages like HTML, JavaScript and CSS. Both these approaches address the issue by allowing flexibility in the User Interface (UI), having it scale or even change according to the size and shape of the device in which is being displayed. However, 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 both technical and non-technical stakeholders to participate in the design and development of these apps. Model-driven development (MDD) is an emerging approach to abstract the complexity of developing software based on models that may represent the structure and behavior of such apps (Atkinson and Khüne, 2003; Saraiva and Silva, 2008 and 2010; Liddle, 2011; Silva, 2015; Wakil and Jawawi, 2017).

This paper describes XIS-Web, a MDD approach particularly focused in the development of responsive web applications. XIS-Web is based on previous work, namely it reuses and adapts concepts from XIS (Silva et al. 2007), XIS-Mobile (Ribeiro and Silva, 2014), WebRatio (Ceri et al., 2002) and IFML (OMG, 2015).

XIS (Silva et al. 2007) proposed a MDD approach for designing web and desktop interactive systems at a platform-independent model (PIM) level, using a domain specific language 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. XIS introduced the idea of smart and dummy modeling approaches (Silva et al. 2007). According to 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 UI models through direct

Seixas, J., Ribeiro, A. and Rodrigues da Silva, A.
A Model-Driven Approach for Developing Responsive Web Apps.
DOI: 10.5220/0007678302570264
In *Proceedings of the 14th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2019)*, pages 257-264
ISBN: 978-989-758-375-9

authoring/design. On the other hand, in the dummy approach, the designer needs to manually define all the views, what is cumbersome and time-consuming. The XIS-Mobile approach was defined with the focus on developing cross-platform mobile applications (Ribeiro and Silva, 2014). XIS-Mobile proposed a DSL that reuses the best concepts proposed originally on XIS, namely its multi-view organization and modeling approaches. XIS-Mobile introduced new concepts (e.g. new types of widgets, internet connection, localization and gesture support) to be more appropriate to design mobile applications scenarios. XIS-Mobile is organized in six views: Domain, BusinessEntities, UseCases, InteractionSpace, NavigationSpace and Architectural. While the first four views share the same constructs as in XIS, 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.

WebML was a visual notation and an approach for designing Web applications at a PIM level. A WebML's Web app specification is defined by four complementary views (Ceri et al., 2002): Structural, Hypertext, Presentation and Personalization models. First, the Structural view defines the data content of the site, in terms of the relevant entities and relationships. In spite WebML did not propose any language for data modelling, it recommends to use UML class diagrams. Second, Hypertext view describes the site structure, which consists of two sub-models: (i) the Composition view that specifies the pages of the web site, and which content units make up each page; and (ii) the Navigation view that expresses how pages and content units are linked to support their navigation. Third, Presentation view describes the layout and graphic appearance of pages, independently of the output device and of the rendering language, by means of an abstract XML syntax. Fourth, the Personalization view allows defining users and user groups as a form of predefined entities, and the features that these entities can be used for defining profile-driven data and business rules, which may guarantee an effective personalization of the site.

WebML was then extended to cover a wider spectrum of front-end interfaces, thus resulting in the IFML (Interaction Flow Modeling Language), adopted as a standard by the Object Management Group (OMG, 2015). IFML provides a consistent visual notation for the definition of PIM interaction flow models that describe the main aspects of an application front-end, namely: the view part of the application, made of view containers and view components; the objects that embody the state of the application, and the references to business logic actions that can be executed; the binding of view components to data objects and events; the control logic that determines the actions to be executed after an event occurrence; and the distribution of control, data, and business logic at the different tiers of the architecture. There are tools that already support the IFML, such as WebRatio or Sparxs Systems EA.

XIS-Web technology includes both a modelling language and a companion framework tool support. The language is defined as a UML profile and provides the necessary concepts for web application modeling. It is currently built on top of Sparx Systems Enterprise Architect (EA) and Eclipse Modeling Framework (EMF) for the M2M and M2T transformations, respectively.

The paper is organized in seven sections. Section 2 details the key features of the language. Section 3 describes the framework tool support. Section 4 discusses some initial experiments and their respective evaluation. Finally, Section 5 summarizes the key points and raises open and future work.

2 XIS-WEB LANGUAGE

The XIS-Web takes advantage of its heritage by reusing concepts originally found in XIS, XIS-Mobile, WebML and IFML, and using its multi-view organization. As depicted in Fig.1, XIS-Web views maintain the same goal and detail as the viewpoints proposed originally in XIS-Mobile; however, some of their inner constructs are new for adding a different ability to express concepts commonly present in web applications. For instance, there are constructs with different options, like in the Architectural View where a “XisInternalService” can be a webcam or microphone, and a “XisRemoteService” can be a WebAPI or JavaScript service. 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).

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 navigation flow between each interaction space. Below we provide a more detailed explanation of each one of these views.

2.1 NavigationSpace View

The NavigationSpace (NS) View is one of the simplest, yet fundamental views of XIS-Web. It allows 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.

2.2 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. 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 IS through a XisDomainAssociation, this define the domain entities manipulated in the context of that IS.

As depicted in Fig.2, XisWidget can either be: (i) XisCompositeWidget, a container widget that groups other XisWidgets; or (ii) 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.

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.

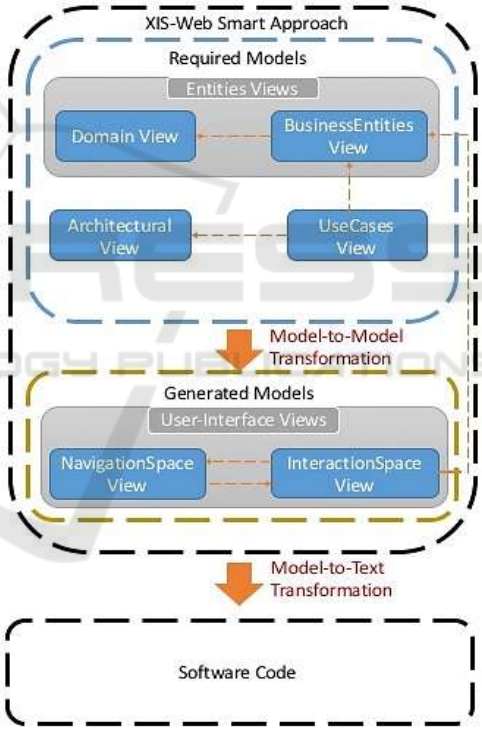

Figure 1: Dependencies between the different XIS-Web views.

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

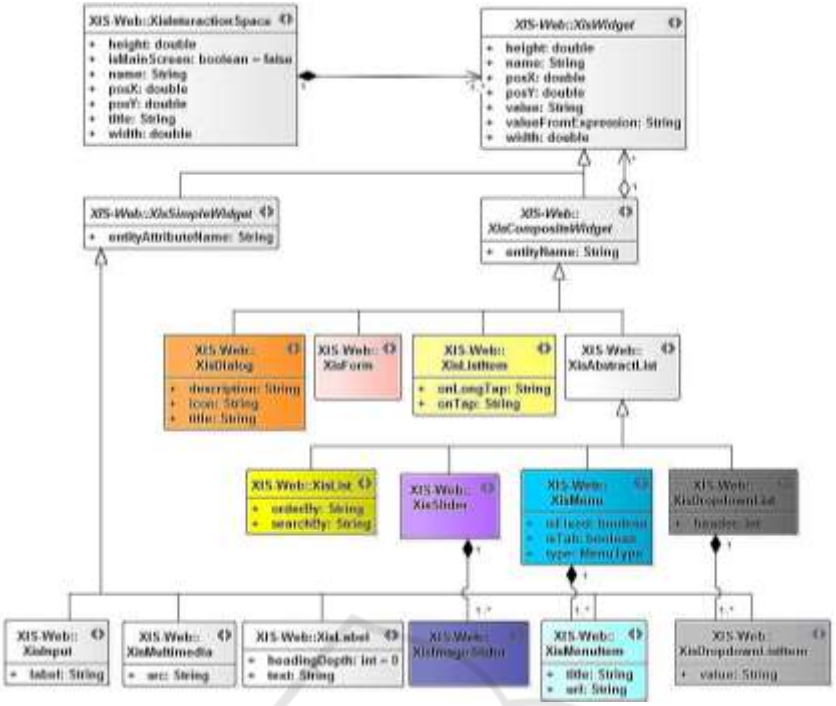

Figure 2: InteractionSpace View metamodel (partial view).

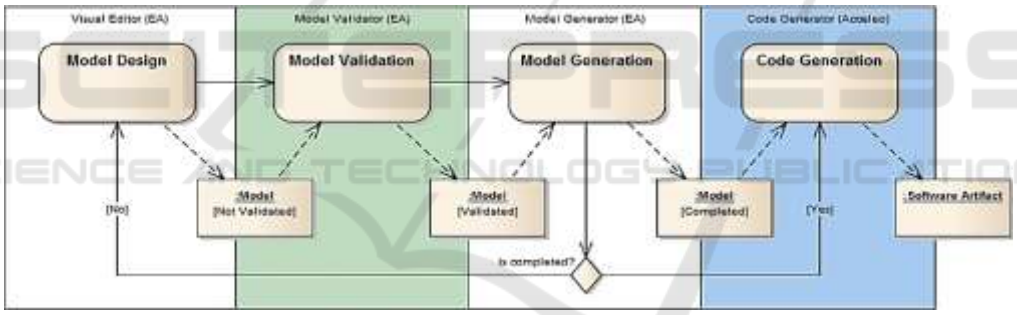

Figure 3: XIS-Web technologies and development process.

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.

2.3 Design Approaches

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

With the "dummy" approach the user shall define all views (the Architectural View is optional, since there may not be interactions with external services) 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 generates interaction spaces applying the well-known "Master-Detail" UI design pattern (Scott and Neil, 2009). Following the guidelines stated by this pattern we shall have per each BE 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.

3 XIS-WEB FRAMEWORK

The major 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 (see Fig. 3). The framework relies in Sparx Enterprise Architect (EA) as main environment, and takes advantages of its Model Driven Generation Technologies in order to validate and apply M2M transformations. It also uses the Acceleo (http://www.eclipse.org/acceleo) plugin present in the Eclipse Modeling Framework to perform the M2T transformations.

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 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 was the technology of choice due to its simplicity and seamless integration with the other technologies. The Hypertext/Application layer is implemented using two technologies that complement each other, HTML5 and JavaScript. Ultimately, for the Presentation layer, we used CSS3 as the main technology, namely applying the Boostrap library which is the most popular framework for developing responsive web applications.

4 EVALUATION

The evaluation of XIS-Web involves the following aspects. First, a set of case studies were developed and evaluated. Second, a pilot-user test session was conducted and preliminary results were obtained.

4.1 Case Studies

Some apps have been developed with the XIS-Web technology; we introduce and report some findings obtained from two of these experiments.

Case Study A: The "TimeSlot Booking App" is an application that allows the management of TimeSlots booked by Students. Each TimeSlot shall have a start date and time, duration and name. TimeSlots also contain Topics that can be associated to other TimeSlots. A TimeSlot is also associated to a Course. A Student shall be able to: manage his TimeSlots; manage his Courses; and share his TimeSlots using an external service. Figure 4 illustrates the top-level view of the XIS-Web model that includes the six view types. Figure 5 shows the Course management's interaction space model and Figure 6 the respective UI form automatically generated by the XIS-Web framework.

Case Study B: The "xDocs App" lets citizens store, manage and share their own personal

documents, (like ID Card, Driver's License, Health Insurance). 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 organization that issues it, namely a user with access to the platform with a more administrative role. xDocs shall be able to communicate with an external service 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 years in a bar chart.

These two apps were developed both according the traditional (manual) approach and the model-driven XIS-Web approach. Table 1 shows the ratio between the lines of code (LOC) generated and the lines of code for the manually implemented versions.

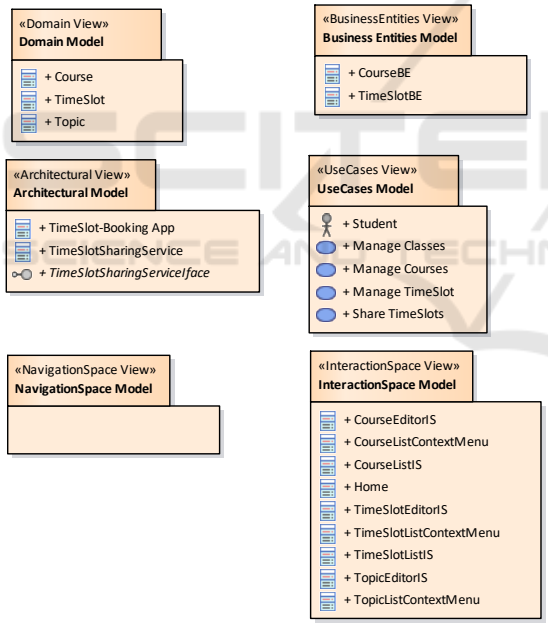

Figure 4: Top-level XIS-Web model (TimeSlot Booking App).

The results obtained were positives because they met one of the goals set for this research: automatically generate more than 70% of an application's source code. Regarding the Case Study A, 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 (xDocs App) the coverage ratio was 69.4%. This happens because xDocs presents more complex logic and UI patterns that are not yet supported in the XIS-Web framework.

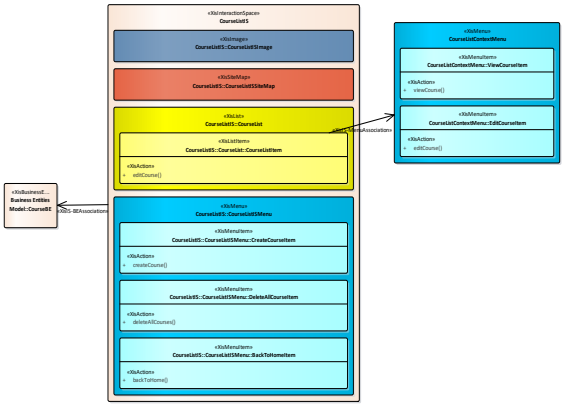

Figure 5: "CourseList" IS model (TimeSlot Booking Appp).

Figure 6:"CourseList" form (TimeSlot Booking App).

4.2 User Session Evaluation

The second part of the evaluation involved a pilot-user test session with the goal to evaluate the usability of XIS-Web approach. This test session focused on three aspects: (i) the Language, namely if it is a good fit for the domain of web applications and its learning curve; (ii) the Framework, namely the Visual Editor, the Model Validator, Model Generator and Code Generator; and (iii) the General Approach.

The participants were people not directly involved in the research and their main goal was the detection of potential bugs and UI limitations. The group had 12 subjects with ages ranging from 23 to 30, with at least a Bachelor in Computer Science or Software Engineering degree. Half of the participants had experience with web application development and 4 participants had professional

experience. The participants received a 15-minute presentation explaining the fundamental concepts of the XIS-Web language and its framework. They were 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 40 minutes. In the end, the participants were asked to fill a questionnaire to rate the XIS-Web language, framework and the general approach.

Table 2 shows the results obtained from this session, which are very encouraging because all the reviewed aspects got very positive scores. From the Language dimension (4.1 in 5) we concluded that the language was not that easy to learn, and it is something we intend to change in future work, by refactoring and reducing the number of concepts available. From the Framework dimension (4.53 in 5) participants considered that the Model Editor is the strongest point of the framework. Overall participants considered that XIS-Web approach (4.33 in 5) brings significant productivity gains when comparing to the other frameworks, but the associated constraints regarding the development environment (i.e., the dependency from the EA tool) may lead them not to use it in their projects.

Table 1: Ratio between generated code and manually implemented code, per language and total.

Languages	TimeSlot Booking App			xDocs App		
	Man.	Gen.	Ratio	Man.	Gen.	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%

Legend: Man(ual); Gen(erated)

Table 2: General evaluation (values in a scale of 1-5).

XIS-Web	Language	Framework	General Approach
	4.1	4.53	4.33

In spite of the number of participants involved in this experiment (12), it is sufficient to take meaningful conclusions, namely considering that experts in usability claim that a small group of 5-testers is enough to reveal over 80% of the usability problems (Nielsen and Landauer, 1993). Moreover, given that our questionnaire focuses on usability aspects, 12 is a reasonable number for an exploratory assessment to identify the major limitations and challenges.

5 CONCLUSION

This paper presents XIS-Web as a model-driven approach that allows the development of responsive web apps. The XIS-Web language is 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 and UI specific concepts like entities and business entities, but also widgets or providers. The views make the language are: Domain, BusinessEntities, Architectural, UseCases, InteractionSpace and NavigationSpace views. In addition, XIS-Web is supporting two design approaches as defined originally in XIS: the dummy approach and the smart approach.

The XIS-Web includes a framework that allows to define and generate responsive web apps. This framework is based on Sparx Systems EA tool, namely making use of its MDG Technologies and Eclipse Modeling Framework, namely through the use of the Acceleo plugins. The framework takes the specification created by the developer, using the XIS-Web language and through the application of M2M and then M2T transformations, generates the source code for the application. This framework comprehends four components: (i) the Visual Editor that allows the definition of the views; (ii) the Model Validator that performs validations on the specification created using pre-defined rules; (iii) the Model Generator that allows the generation of the UI Views; and (iv)the Code Generator that generates the application's source code.

XIS-Web was evaluated to assess its usefulness and usability to the purpose of modelling responsive web applications. The evaluation process was conducted in two complementary aspects: case study comparison (manual implementation vs. automatic generation), and a pilot-user test session assessment.

First, the results from the comparison between the manual implementation and the automatic generation of the case studies 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 (xDocs App), the ratio of generation was of 70%. These results were considering very positive particularly as a proof of concept.

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 XIS-Web aspects: the language, the framework and the general approach. The results obtained from this experiment were also very positive. These results showed preliminary evidences of XIS-Web's usefulness, and usability to modelling responsive web applications.

For future work we identify some aspects to improve. First, the case studies presented in this research can only exercise and evaluate certain parts of the framework; thus, to better evaluate the proposed approach, one needs to conduct other case studies, varying in complexity and subject. Second, currently XIS-Web is applying the Master-Detail UI pattern; to diversify the type of applications that can be generated by XIS-Web it would be important to add the generation of other UI patterns like Breadcrumbs, Galleries or Dashboards (Crumlish and Malone, 2009; Scott and Neil, 2009). Third, considering that IFML is the OMG standard for UI modelling, we would consider integrate its constructs in XIS-Web, particularly those related with the Interaction and Navigation views.

ACKNOWLEDGEMENTS

This work was partially supported by national funds under FCT projects UID/CEC/50021/2019 and 02/SAICT/2017/29360.

REFERENCES

Atkinson, C., Khüne, T., 2003. Model-driven development: a metamodeling foundation, In IEEE Software, IEEE, 20(5), 36-41.

Ceri, S. et al., 2002. Designing Data-Intensive Web Applications. Morgan Kaufmann Publishers.

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

Crumlish, C., Malone, E., 2009. Designing social interfaces: Principles, patterns, and practices for improving the user experience. O'Reilly Media, Inc.

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

Garrigos, I., Gomez, J., Cachero, C., 2003. Modelling adaptive web applications. Proc. of the IADIS International Conference WWW/Internet, pp. 813-6.

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

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

Kappel, G., Prll, B., Reich, S., Retschitzegger, W., 2006.Web engineering. John Wiley & Sons.

Koch, N., Kraus, A., 2002. The expressive power of uml-based web engineering. In Proc. of the 2nd International Workshop on Web-oriented Software Technology (IWWOST 2002), pp. 21-32.

Liddle, S. W., 2011. Model-driven software development. In Handbook of Conceptual Modeling, pp. 17-54. Springer.

Nielsen, J., Landauer, T.K., 1993. 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.

OMG, 2015. Interaction Flow Modeling Language, version 1.0, http://www.omg.org/spec/IFML/.

Pastor, O., Fons, J., Pelechano, V., & Abrahão, S., 2006. Conceptual modelling of web applications: the OOWS approach. In Web Engineering (pp.277-302). Springer.

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

Ribeiro, A., Silva, A.R.,2014. XIS-Mobile: A DSL for Mobile Applications",in Proceeding of ACM SAC, ACM.

Saraiva, J., Silva, A.R., 2008. Evaluation of MDE Tools from a Metamodeling Perspective. Journal of Database Management, 19(4): 50-75.

Saraiva, J., Silva, A.R., 2010. A Reference Model for the Analysis and Comparison of MDE Approaches for Web-Application Development, Journal of Software Engineering and Applications, 3(5): 419-425.

Rossi, G., Urbieta, M., Distante, D., Rivero, J. M., & Firmenich, S. (2016). 25 Years of Model-Driven Web Engineering. What we achieved, What is missing. CLEI Electronic Journal, 19(3), 5-57.

Schwinger, W., et al., 2008. A survey on web modeling approaches for ubiquitous web applications. IJWIS, 4(3):234-305.

Schwinger, W., Retschitzegger, W., Schauerhuber, A., Kappel, G., Wimmer, M., Pröll, B., & Garrigos, I., 2008. A survey on web modeling approaches for ubiquitous web applications. International Journal of Web Information Systems, 4(3), 234-305.

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

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

Silva, A.R., 2015. Model-Driven Engineering: a Survey Supported by a Unified Conceptual Model. Computer Languages, Systems & Structures, Elsevier, 43(C),139–155.

Wakil, K., & Jawawi, D. N., 2017. Comparison between Web Engineering Methods to Develop Multi Web Applications. Journal of Software, 12(10), 783-794.