XIS-CMS: Towards a Model-Driven Approach for Developing Platform-Independent CMS-Specific Modules

Paulo Filipe, André Ribeiro and Alberto Rodrigues da Silva
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Lisbon, Portugal

Keywords: Content Management Systems, Model-Driven Development, Platform-Independent Models, Modeling Languages.

Abstract: Content Management Systems (CMS) are popular web application platforms used in multiple domains. CMS allow non-technical users to manage the content and features of websites with web modules that abstract functionality without requiring particular software programming background. However, without the development of specific web modules, a CMS usually cannot support complex scenarios or specific business needs. In those situations, developers have to build custom modules using the CMS-specific language, which implies that they must master the corresponding programming and other technical skills. This paper proposes a model-driven approach, named XIS-CMS, which aims to increase the productivity and portability of developing these modules in a more abstract and platform-independent way. XIS-CMS approach includes a domain-specific modeling language, defined as a UML profile, and a companion framework defined on top of Sparx Systems Enterprise Architect and Eclipse Modeling Framework technologies. This paper introduces the XIS-CMS approach, its corresponding language and framework, and compares it with related work.

1 INTRODUCTION

Content Management Systems (CMSs) are a popular class of software framework that abstracts the technical features concerning the development and management of web applications. CMS-based applications are defined as the orchestration of multiple web pages with multiple contents presented and managed by particular use of web modules, such as HTML, Image, and List of Links. A web module consists in a collection of code and resource files organized in a library that allows to extend a CMS framework with specific features and can be instantiated in multiple pages of a website or web application (Suh et al., 2002; Boiko, 2001).

A website is mainly composed of static content that needs to be maintained by a developer or webmaster. On the other hand, a web application provides a dynamic experience to the user focusing on his interaction to determine the content that is displayed. A CMS platform supports web applications development and management, because it provides several features to manage the structure, content and presentation of these applications. CMS provide a ready-to-use editor for managing and creating new pages and menus, and for applying user interface layouts and visual templates. They also offer several out-of-the-box modules with basic functionality, such as HTML editor, images, list of links and news or file management (Souer et al., 2008). However, custom modules must be implemented whenever more complex functionality or more specific business requirements are needed. This kind of customer modules requires an additional analysis and development by software developers (Souer and Kupers, 2009).

CMSs show important features and properties such as modularity, extensibility and integration with other services through toolkits of modules. CMSs also promote the separation between the presentation and the content through the application skins, page templates and module views. CMSs offer great adaptability to define and manage the structure of web applications, but do not provide a high-level view of the data that is being used and/or manipulated. Thus, an approach that closes this gap would be an important contribution, by allowing developers and non-technical stakeholders to easily discuss strategic changes or business requirements. This would result in a shorter time-to-market and an

Filipe, P., Ribeiro, A. and Silva, A.
XIS-CMS: Towards a Model-Driven Approach for Developing Platform-Independent CMS-Specific Modules.
DOI: 10.5220/0005745805350543
In *Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2016)*, pages 535-543
ISBN: 978-989-758-168-7

easier maintainability of this class of applications.

With the emergence and evolution of many CMSs frameworks (see for example, the popular CMS Matrix website: http://www.cmsmatrix.org), the adoption of a development approach that can keep the core development portable and easy to apply to new CMS would be also a great advantage. For this reason, such an approach should be as much cross-platform as possible, meaning it should be independent of any CMS framework.

Model-Driven Development (MDD) is an approach that tries to mitigate the above problems in multiple application domains (Schmidt, 2006); (Selic, 2008); (Hutchinson et al., 2014). MDD is a software development paradigm that combines domain-specific languages (DSLs) with transformation engines and generators. DSLs are textual or graphical languages used to specify an application domain, the requirements and functionalities (Deursen, 2000); (da Silva, 2015). DSLs might be described by metamodels that define the domain concepts and associations established between them. Additionally, transformations engines are tools that process a model (defined in a DSL) and generate other models, through Model-to-Model (M2M) transformations, or textual artifacts such as source code, through Model-to-Text (M2T) transformations (Hermans et al., 2009).

This paper proposes the XIS-CMS approach for the development of CMS-specific modules by defining platform-independent domain models with simple or medium complexity, and then to generate the corresponding source code automatically. This approach includes the XIS-CMS language and the XIS-CMS framework that is its companion supporting tool. The XIS-CMS language is a DSL, defined as a UML profile, organized in multiple views that describe several aspects of a set of CMS modules, such as domain model, types of users and permissions. In turn, the XIS-CMS framework supports that language by offering features like design, validation and model transformations.

The outline of this paper is as follows. Section 2 describes the context in which the XIS-CMS approach is based. Section 3 presents the definition of the XIS-CMS language, describing each view and illustrating their use with a very simple case study. Section 4 describes the XIS-CMS framework and the tools used to support the applicability of the XIS-CMS language. Section 5 presents the results of the evaluation of XIS-CMS. Section 6 refers and discusses the related work. Finally, Section 7 presents the conclusion and issues for future work.

2 BACKGROUND

XIS-CMS derives from previous work, namely from the XIS and XIS-Mobile UML profiles. XIS is focused on the design of interactive systems at a Platform-Independent Model (PIM) level following a MDD approach. XIS (Silva et al., 2007) comprises three major sets of views: Entities, Use-Cases and User-Interfaces. First, the Entity view is composed of the Domain and BusinessEntities views. The Domain view represents the relevant classes to the problem domain, their attributes and the relationships among them. In turn, the BusinessEntities view defines higher-level entities, known as business entities, that 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 NavigationSpace view defines the navigation flow between the several screens of the system (designated by interaction spaces) with which the user interacts. In turn, the InteractionSpace view represents the elements of the graphical interface contained in each interaction space and can also specify the access control of the actors to these elements.

XIS also proposes two modeling approaches: the smart approach and the dummy approach (Silva et al., 2007). When using the smart approach, the designer only needs to define the Domain, BusinessEntities, Actors and UseCases views. Then, the User-Interfaces views are automatically generated through M2M transformations (based on the models defined in the Domain and UseCases views). After that, 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 define manually the entire Domain, Actors, NavigationSpace and InteractionSpace views. XIS was originally defined to develop desktop or web interactive applications and to generate code for software frameworks, such as Windows Forms.NET and Microsoft ASP.NET, while XIS-CMS is specifically focused on CMS modules applications, particularly in CMS module development.

More recently, XIS-Mobile, an extension of XIS, was defined with the focus on developing cross-platform mobile applications (Ribeiro and da Silva, 2014a); (Ribeiro and da Silva, 2014b). XIS-Mobile is a DSL that reuses some of the best concepts proposed on XIS, namely its multi-view organization and design approaches. Additionally, it 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. Thus, the XIS-Mobile language is organized in six views: Domain, BusinessEntities, UseCases, InteractionSpace, NavigationSpace and Architectural. While the first four views share the same goals as in XIS (however with different stereotypes and adjustments), 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 models described in the XIS-Mobile language, generating other models from them (through M2M transformations) and ultimately generating native source code for multiple mobile platforms (Android, iOS and Windows Phone), through M2T transformations (Ribeiro & da Silva, 2014a).

3 XIS-CMS LANGUAGE

There are few languages that attempt to design the whole structure and management of a web application (i.e., including menus, pages, modules and page deployment), such as CMS-ML (Saraiva, 2013) and JooMDD (http://icampus.thm.de/joomdd). Unlike these languages, the XIS-CMS language is focused on the design and development of toolkits of CMS modules, with the respective features for data and modules management. The XIS-CMS language adopts some concepts defined in the XIS language, but introduces new views and an appropriate multi-view organization. As depicted in Figure 1, XIS-CMS models are organized in a Toolkit view that includes three major views: Entities, Modules and Roles views. In turn, the Entities view contains two sub-views: Domain and BusinessEntities views. The Modules view contains two sub-views: UseCases and User-Interfaces views. At last, the User-Interfaces view comprises the NavigationSpace and the InteractionSpace views.

XIS-CMS proposes two modeling approaches: the smart approach and the dummy approach. While in the dummy approach all views must be manually defined, in the smart approach, the NavigationSpace and the InteractionSpace views are automatically generated, through M2M transformations, from the Domain, BusinessEntities, Roles and UseCases views.

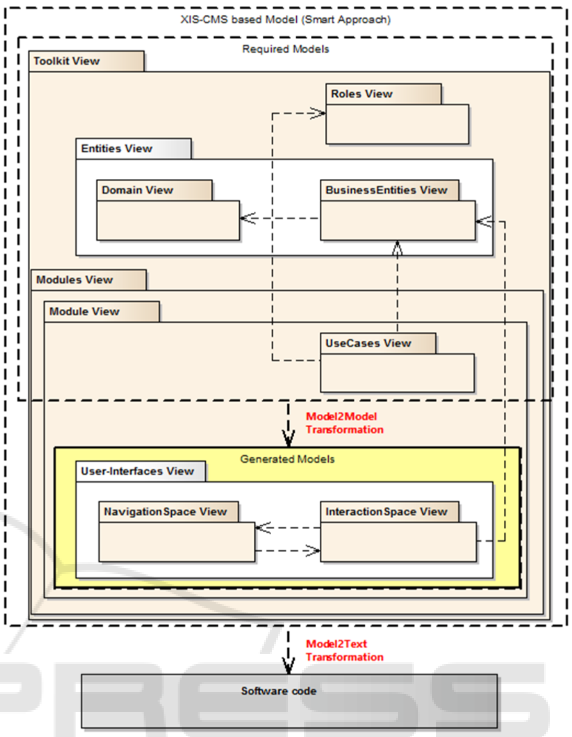


Figure 1: The multi-view organization of XIS-CMS.

For the sake of better understand the XIS-CMS language, we introduce a simple case study:

«The **MySuppliers** web application allows managing the products and the suppliers of a generic store. A supplier is defined by his personal information (name, registration date, address and city) and can supply several products until a configured maximum value (defined by the module setting). Each product has a name, width and height. The products are managed by a Product Manager, while the suppliers are managed by a Supplier Manager. The Supplier Manager can also view, but not edit, the products. For this purpose, it is necessary to define a toolkit with two modules: one for the product management and another for the supplier management. This toolkit can be deployed on multiple CMS platforms.»

3.1 Toolkit View

The Toolkit view includes all the other views and represents the logical aggregation of shared business concepts and modules. It also provides configuration through the use of tagged values. The Toolkit view has tagged values to identify the owner of the toolkit, the type of repository data and the connection information to access the repository.

3.2 Entities View

The Entities view contains the Domain and the BusinessEntities views, and describes the domain and business entities and their relationships at different abstraction levels.

Domain View. The Domain view defines the problem domain entities commonly captured from a domain analysis. It is possible to specify the attributes of the domain entities and the relationships among them using associations, aggregations or inheritances. Each domain entity is represented by a XisEntity stereotype, which in turn contains one or more attributes, defined as XisAttributes stereotypes. Both stereotypes are used, instead of just simple UML Classes or Attributes, to provide useful information to the M2T transformations.

Regarding this case study, the MySuppliers application only manages two domain entities: Product and Supplier, as shown in Figure 2. The Supplier entity has a n-ary association with the Product entity.

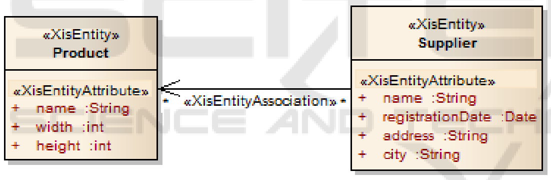


Figure 2: Domain view of the MySuppliers.

Business Entities View. The BusinessEntities view defines high-level entities called business entities. Each business entity is defined as a XisBusinessEntity that aggregates XisEntities. The goal of these business entities is to provide context to the use cases and interaction spaces, which will be useful during the model transformation stages. A XisBusinessEntity defines a master entity (from the Domain View) through a XisMasterAssociation and can also define detail and reference entities through XisDetailAssociations and XisReferenceAssociation respectively. Both association types contain a tagged value named "filter" that allows the restriction of the entity's attributes that can be used in the context of the respective XisBusinessEntity. The definition of a master entity restricts the set of detail and reference entities that can be used by a XisBusinessEntity. Both detail and reference entities must be associated to the master entity in the Domain View, respectively through aggregations and associations.

There are two business entities in the MySuppliers application: ProductBE and SupplierBE. The ProductBE defines the Product as master entity, whereas the SupplierBE defines the Supplier as master entity and the Product as a reference entity.

3.3 Roles View

The Roles view defines the actors or roles that interact with the system and that are specific to the defined toolkit. Each role is defined with the XisRole stereotype and can be organized in a hierarchy of sub-roles. These roles will be instantiated in the CMS roles list, but during the M2T transformation the system may detect that some CMS standard roles already exist (e.g. Administrators, Registered users) and will just add the corresponding permissions to these roles.

3.4 Modules View

The Modules view defines the intended web modules. Each module contains a UseCases and a User-Interfaces views, which define its management operations over a business entity and its module interface elements, respectively.

UseCases View. The UseCases view describes the operations that each role is allowed to perform regarding a specific business entity. A CMS module is represented as a XisModule, which may contain XisModuleConfigurations, represented as attributes and operations. Permissions to the operations are defined by the XisRole-ModuleAssociation established between a XisRole and a XisModule.

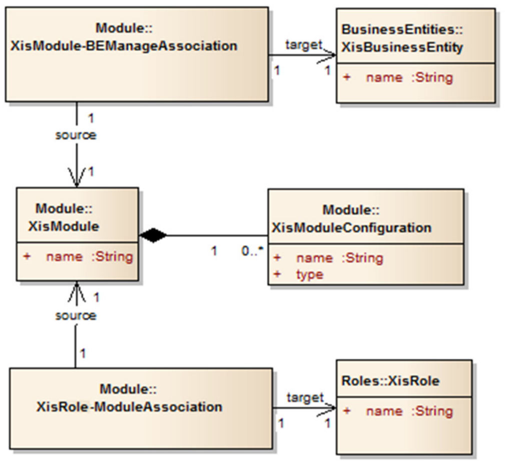


Figure 3: Metamodel of the UseCases view.

Figure 3 shows these stereotypes and their relationships. The UseCases view provides the higher-level information of a CMS module, namely: which data is manipulated; the operations that can be performed; and who can perform which operation.

The MySuppliers application is composed of two modules: Product and Supplier. Figure 4 illustrates the UseCases view for the Product module, where the Product Manager role can perform all available operations (view, create, edit and delete) while the Supplier Manager role can only view information.

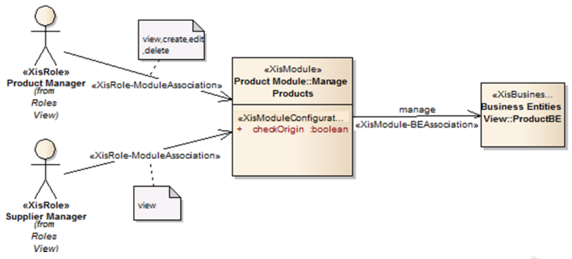

view,create,edit ,delete
«XisRole» Product Manager (from Roles View)
«XisRole-ModuleAssociation»
«XisModule» Product Module::Manage Products
«XisModuleConfigurat...
+ checkOrigin :boolean
manage
«XisModule-BEAssociation»
«XisBusines... Business Entities View::ProductBE
«XisRole-ModuleAssociation»
«XisRole» Supplier Manager (from Roles View)
view

Figure 4: UseCases view of the MySuppliers.

User-Interfaces View. The User-Interfaces view comprises the InteractionSpace and the NavigationSpace views, which define the UI screens of each module and the flows between them.

InteractionSpace View. The InteractionSpace view defines each screen (or "interaction space") detailing all UI elements and associated events. Each screen is defined as a XisInteractionSpace associated to a XisModule through a XisIS-ModuleAssociation and aggregates one or more XisWidgets. A XisInteractionSpace is defined by the "isEntry" and "isDefault" tagged values. Only one XisInteractionSpace per module can have the "isDefault" value set to true, causing the CMS to initiate the module in this screen. But more than one XisInteractionSpace can be defined as an entry screen (if "isEntry" is set to true), allowing the CMS to define multiple views of the same module. The XisWidget represents a web control or UI element and is refined in two groups: XisSimpleWidgets (e.g. button, label or input field) and XisCompositeWidgets (e.g. controls that group other widgets like a grid or a panel). XisWidgets can have actions associated with them, which trigger events or navigation flows. This view is the most labor-intensive view of the XIS-CMS language, because it represents all of the UI elements and their layout and allows the aggregation of multiple UI elements. Thus it is recommended to use the smart approach to generate this view, leveraging the M2M transformations. After that, the designer can customize and refine directly the generated views if they do not fulfill the desired requirements.

Considering the MySuppliers application, the SupplierIndex is the entry screen of the Supplier module. It contains a XisGrid widget that describes a grid with columns for each supplier's attributes: name, registrationDate, address and city. The XisGrid has a context menu associated with the actions "View" and "Edit". The "View" action causes the navigation to the SupplierView screen and the "Edit" action to the SupplierEdit screen in edit mode. The SupplierIndex also contains a XisButton with the "Create" action that allows navigating to the SupplierEdit screen in create mode.

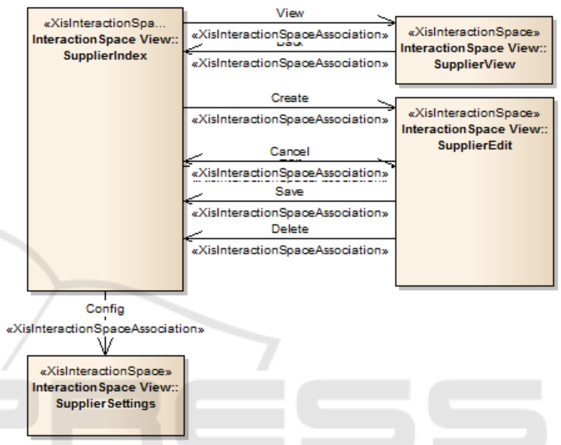

«XisInteractionSpa... Interaction Space View:: SupplierIndex
View
«XisInteractionSpaceAssociation»
«XisInteractionSpaceAssociation»
«XisInteractionSpace» Interaction Space View:: SupplierView
Create
«XisInteractionSpaceAssociation»
«XisInteractionSpace» Interaction Space View:: SupplierEdit
Cancel
«XisInteractionSpaceAssociation»
Save
«XisInteractionSpaceAssociation»
Delete
«XisInteractionSpaceAssociation»
Config
«XisInteractionSpaceAssociation»
«XisInteractionSpace» Interaction Space View:: Supplier Settings

Figure 5: NavigationSpace view of the MySuppliers.

NavigationSpace View. The NavigationSpace view defines the navigation flow between the several screens accessible to the end-user when interacting with a module. The navigation flows between the XisInteractionsSpaces are defined by XisInteractionSpaceAssociations. Such associations have the "actionName" tagged value to identify the action that triggers the navigation. This view also provides an overview of the structure of a module and its interactions, which is useful both to technical and non-technical stakeholders.

Regarding the MySuppliers application, the Supplier module's NavigationSpace view is composed of eight XisInteractionSpaceAssociations and four XisInteractionSpaces (see Figure 5): (1) SupplierIndex shows all the suppliers; (2) SupplierView shows a read-only screen with the information of the selected supplier; (3) SupplierEdit allows the creation/edition of a supplier; and (4) SupplierSettings represents a custom settings screen to configure the maximum number of products.

Table 1 summarizes the main stereotypes defined in each view of the XIS-CMS language and the stereotypes used from other views. For example, the UseCases view uses stereotypes defined in the BusinessEntities and the Roles views in order to define which entities are managed and the corresponding access permissions.

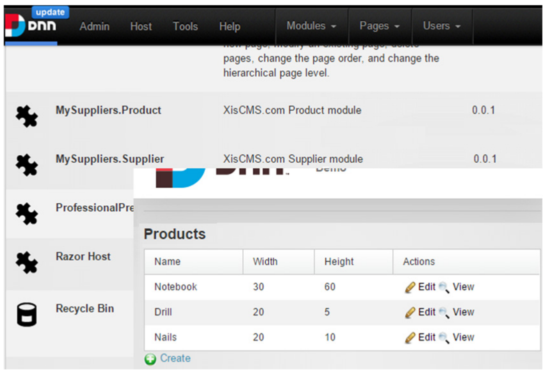
Table 1: XIS-CMS views and stereotypes.

View	Stereotypes	Uses
Domain	XisEntity	
	XisEntityAttribute	
	XisEntityAssociation	
BusinessEntities	XisBusinessEntity	XisEntity
	XisMasterAssociation	
	XisDetailAssociation	
	XisReferenceAssociation	
Roles	XisRole	
UseCases	XisModule	XisBusiness Entity
	XisModuleConfigurations	XisRole
	XisModule-BEManageAssociation	
	XisActor-ModuleAssociation	
InteractionSpace	XisInteractionSpace	XisModule
	XisIS-ModuleAssociation	
	XisWidgets	
	XisSimpleWidget	
	XisCompositeWidget	
	XisForm	
	XisField	
	…	
NavigationSpace	XisInteractionSpaceAssociation	XisInteractionSpace

4 XIS-CMS FRAMEWORK

XIS-CMS approach only becomes relevant when combines the language with a support MDD-based framework. The XIS-CMS framework is implemented using the Model Driven Generation (MDG) Technologies provided by Sparx Systems Enterprise Architect (EA) along with the EMF (Eclipse Modeling Framework). XIS-CMS suggests a development process comprising the following steps: (1) the definition of the required XIS-CMS views using the Visual Editor; (2) the validation of those views through the Model Validator; (3) the generation of the User-Interfaces views using the Model Generator; and (4) the generation of the application's source code for the target CMS platform through the Code Generator. If the models generated after step (3) do not meet all the designer's requirements, the process should return to step (1) for a new iteration where the designer performs his corrections. With the exception of step (1), which is manual, the other three steps are automatic, only requiring the designer to trigger their execution.

The XIS-CMS framework is composed of four modules: Visual Editor, Model Validator, Model Generator, and Code Generator. First, the Visual Editor is implemented on top of EA through the development of an MDG Technology plug-in. This plug-in allows the definition of the XIS-CMS language as a UML profile fully compliant with the OMG specification for UML2. Additionally, it also allowed the creation of toolboxes, diagram types and diagram templates customized to the XIS-CMS language. Second, the Model Validator is implemented as a plug-in leveraging EA's Model Validation API. This solution allowed to programmatically define validation constraints, assigning severity levels (error or warning) to them, and in runtime to navigate directly to the error-causing element. Despite that, it is not an OMG standard, like OCL. The Model Validator plays a decisive role in the XIS-CMS approach, namely detecting errors produced by the designer at an early stage of development, enforcing the quality of the models and, consequently the quality of the generated models and code. Third, the Model Generator is also implemented using EA's environment, namely through EA's Automation Interface. It allows accessing the repository containing the created models, as well as creating new ones. The Model Generator is the responsible for performing the M2M transformations, using the information of the Domain, BusinessEntities, Roles and UseCases views. Fourth, the Code Generator is based on Acceleo, a template-based code generator framework available as an Eclipse plug-in. Acceleo implements the MOF Model to Text Language standard and allows defining code templates for any kind of model compatible with EMF. The code templates are composed of regular text (static part of the template) and annotations (dynamic part of the template) which are replaced by values of the model during generation time. For now, the XIS-CMS framework supports the generation of CMS modules applications for the DotNetNuke (DNN) platform. It is important to emphasize that whenever the designer requires the support of other CMS platforms, he only needs to define the corresponding code templates using Acceleo. Figure 6 shows the generated toolkit modules of the MySuppliers application and a sample of the Product module.



pages, change the page order, and change the hierarchical page level.

MySuppliers.Product	XisCMS.com Product module	0.0.1
MySuppliers.Supplier	XisCMS.com Supplier module	0.0.1
ProfessionalPre		
Razor Host		
Recycle Bin		

Products

Name	Width	Height	Actions
Notebook	30	60	Edit View
Drill	20	5	Edit View
Nails	20	10	Edit View

Create

Figure 6: Modules generated by XIS-CMS.

5 EVALUATION

XIS-CMS has been subject to a two-fold evaluation. The first evaluation consisted in the development and analysis of case studies using XIS-CMS, and their comparison with manually programmed modules. This phase produced successful results. However, the development of these case studies focused on CRUD operations and gave minimum support to custom operations.

The second phase involved the realization of user test sessions to evaluate XIS-CMS from the perspective of users not involved in this work. The tests session occurred in laboratory sessions and involved a total of ten users with at least a Bachelor degree and ages between 22 and 48 years old. Six participants had previous experience in the development of CMS applications, namely in module development, and eight had professional experience in the IT field. The test session was composed of a short presentation introducing XIS-CMS, followed by a simple demonstration. Then each user received a document with the installation instructions and how to implement a case study regarding managing appointments in a medical clinic. In the end, the users were asked to fill an online questionnaire to provide their opinion. The questionnaire was divided in three sections concerning the assessment of the following aspects of XIS-CMS: Language, Framework and Approach, and the results were promising, namely 3,87 (Language), 4,55 (Framework) and 4,2 (Approach), in a scale from 1 (Very Low) to 5 (Very Good). Although the sample of participants was small, the results were relevant and provided enough feedback to detect flaws, especially considering that it was an evaluation of the first version of XIS-CMS.

6 RELATED WORK

Although the idea of using MDD approaches to develop web applications is not new, it is not widely applied to the domain of CMS and therefore is still not fully explored and challenged.

JooMDD (http://icampus.thm.de/joomdd) applies a reverse-engineering method to obtain the model for the Joomla CMS platform. It has two components: one for developing extensions and another for managing the web application structure. Despite it achieved some success (Priefer, 2014), it is a platform-specific approach and not satisfying our goal of platform independence.

The CM Data Architect and the Web Application Architect are tools developed by IBM resulting from the attempt to implement a MDD approach to the IBM Content Management System (Deshpande et al., 2005). These tools separate the views from the architect (to manage the entities involved and the relationships between these entities) and the user interface designer (to create and manage the UI artifacts). Although it performs M2T transformations to accelerate the generation of source code, it does not use M2M transformations to generate the user interface view. Thus, the user interface designer must manually develop it.

The Web Specific Language (WSL), developed in the University of Oxford, adopts MDD to generate web applications from models with a focus on the data and workflows (based on workflows engines). WSL is a textual language that uses syntax-to-metamodel and M2T transformations engines to create the artifacts needed to run the application. These engines were implemented using the EMF. Although WSL is an easy and comprehensive language due to its closeness to natural language, it lacks the visual modeling environment. Furthermore, despite adopting a platform-independence approach, it is more oriented to applications focused on integration and automatic services than on web applications with user interaction (Svansson and Lopez-Herrejon, 2014).

Similarly to XIS-CMS, CMS-ML (de Sousa Saraiva and da Silva, 2009) is a graphical language and is organized in multiple views. However, CMS-ML has a wider scope than XIS-CMS in the sense that it allows specifying website templates and how toolkits will be deployed in a given website. In turn, XIS-CMS targets the development on a toolkit level assuming that the user has to manually load each toolkit and manage its configurations. Extending XIS-CMS to support the definition of website templates and multi-tenancy is an interesting future work that we intend to research. XIS-CMS shares some concepts with the Toolkit View of CMS-ML. Namely, both languages have Domain and Roles views; CMS-ML's WebPage and WebComponent concepts are equivalent to XIS-CMS's XisInteractionSpace and XisWidget, respectively (Saraiva, 2013). Additionally, comparing to CMS-IL, XIS-CMS is in a higher level of abstraction and does not provide concepts that allow specifying too technical features. Also, XIS-CMS is supported by a technological framework, while CMS-ML and CMS-IL do not.

XIS-CMS does not aim to replace the role of the software developer, but creates the structure of the modules and accelerates the production of common and boilerplate code, which is a considerable portion of the final application's source code. Then, the developer can customize the source code to better refine and implement the application requirements.

7 CONCLUSIONS

In this paper we presented the XIS-CMS approach, which includes both a domain-specific language and a framework, as a solution to model and develop cross-CMS modules. XIS-CMS intends to allow both technical and non-technical stakeholders to understand and eventually define these modules and respective applications. XIS-CMS follows a MDD approach, using M2M and M2T transformations, in order to increase the productivity by automatically generating models but also other artifacts, such as source code, simplifying the development and maintenance of these applications. XIS-CMS promotes a cross-platform solution through its platform-independent modeling language, in order to increase the portability of its models regardless of the CMS platform used. The XIS-CMS language is a UML profile that applies the "separation of concerns" principle by defining multiple views to represent each aspect of the CMS modules. Namely, it comprises three main views: Entities (including the sub-views Domain and BusinessEntities), Roles and Modules (including the sub-views Use-Cases and User Interfaces) views. On the other hand, the XIS-CMS framework is based on EA MDG technologies and EMF, and provides features like a Visual Editor, a Model Validator and applies M2M and M2T transformations to generate respectively more complex models and source code ready to use in the targeted CMS.

As future work, we intend to apply XIS-CMS in more complex and real-world domains, which represent opportunities to better employ and evaluate all the concepts that XIS-CMS provides and further explore its requirements. Also, research has to be developed in order to extend XIS-CMS so that it would support the modeling and implementation of a complete CMS web application structure including, for example, the deployment and management of its menus, pages and containers.

ACKNOWLEDGEMENTS

This work was partially supported by national funds through FCT, under the projects POSC/EIA/57642/2004, CMUP-EPB/TIC/0053/ 2013, UID/CEC/50021/2013 and DataStorm Research Line of Excellency funding (EXCL/EEI-ESS/0257/2012).

REFERENCES

Boiko, B., 2005. *Content Management Bible.* John Wiley & Sons, Inc.

Da Silva, A.R., 2015. Model-Driven Engineering: A Survey Supported by a Unified Conceptual Model. Computer Languages, Systems & Structures, 43.

Deshpande, P. et al., 2005. *Model Driven Development of Content Management Applications.* COMAD.

De Sousa Saraiva, J., & da Silva, A.R., 2009. *CMS-based Web-Application Development Using Model-Driven Languages*. Proc. of ICSEA, IEEE.

Deursen, A., 2000. Domain-specific languages: an annotated bibliography. *ACM SIGPLAN Notices*, 35(6).

Hermans, F., Pinzger, M. & Van Deursen, A., 2009. Domain-specific languages in practice: A user study on the success factors. *Model Driven Engineering Languages and Systems, LNCS 5795*.

Hutchinson, J., Whittle, J. & Rouncefield, M., 2014. Model-driven engineering practices in industry: Social, organizational and managerial factors that lead to success or failure. *Science of Computer Programming*, 89.

Priefer, D., 2014. *Model-Driven Development of Content Management Systems based on Joomla.* Proc. of ACM/IEEE ASE.

Ribeiro, A. & da 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. & da Silva, A., 2014. *XIS-Mobile: A DSL for Mobile Applications.* Proc. of SAC, ACM.

Saraiva, J., 2013. *Development of CMS-based Web Applications with a Multi-Language Model-Driven Approach (PhD Thesis),* Universidade de Lisboa.

Schmidt, D., 2006. Guest Editor's Introduction: Model-Driven Engineering. *IEEE Computer,* 39(2).

Selic, B., 2008. Personal reflections on automation, programming culture, and model-based software engineering. *Automated Software Engineering,* 15(3-4).

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

Souer, J., Honders, P., Versendaal, J. & Brinkkemper, S., 2008. A framework for web content management system operation and maintenance. *Journal of Digital Information Management,* 6(4).

Souer, J. & Kupers, T., 2009. *Towards a Pragmatic Model Driven Engineering Approach for the Development of CMS-based Web Applications.* Proc. of MDWE, .

Suh, P., Ellis, J. & Thiemecke, D., 2002. *Content Management Systems.* Peer Information.

Svansson, V. & Lopez-Herrejon, R., 2014. *A Web Specific Language for Content Management Systems.* OOPSLA Workshop on Domain-Specific Modeling.