CMS-oriented Modeling Languages: An Attempt to Assist Model-driven Development in CMS Domain

Vassiliki Gkantouna[1][a] and Giannis Tzimas[2][b]
[1]*Department of Computer Engineering and Informatics, University of Patras, Greece*
[2]*Department of Electrical and Computer Engineering, University of Peloponnese, Greece*

Keywords: Web Modeling, Content Management Systems, CMS-based Web Applications, Model-driven Development.

Abstract: Nowadays, Content Management Systems (CMSs) are widely used as the underlying development platform for building complex Web applications. However, despite their widespread use, existing MDWE methodologies have focused mainly on traditional Web applications, and thus, they cannot support the model-driven development of CMS-based Web applications. Given that MDWE methodologies are driven by the expressiveness of the modeling languages which are being used within their context, the failure of the existing MDWE methodologies to support the automated development of CMS-based Web application is probably caused by the absence of modeling languages able to capture the particular development context of CMS platforms. To address this problem, we propose a new genre of modeling languages, called CMS-oriented modeling languages, which are particularly defined over the specific development context of CMS platforms. We provide a general framework to support their definition in three main stages, involving the analysis of the target CMS platform, the creation of its domain model and the formal definition of the CMS-oriented modeling language. In this way, the proposed framework supports the definition of CMS-oriented modeling languages, which can lay the foundation for the development of MDWE methodologies for CMS-based Web applications, thus enabling model-driven development in CMS domain.

1 INTRODUCTION

Content Management Systems (CMSs) play an increasingly important role in the evolution of the World Wide Web, since almost half of the websites today use some form of CMS as their main development platform (W[3]Techs, 2019). They provide development teams with standardized software platforms that significantly facilitate and speed up the development of Web applications, the so-called CMS-based Web applications, while maintaining high quality and low-cost implementation without requiring extensive programming expertise.

Despite the widespread use of CMSs, the development of CMS-based Web applications is still entirely performed via the manual configuration and customization of the underlying CMS platform. However, as the complexity of CMS-based Web applications is increasing, these tasks are becoming more and more troublesome and error-prone, often resulting in higher development and maintenance costs. On this basis, we argue that the adoption of the model-driven paradigm in CMS development, supporting the automated generation of CMS-based Web applications, can be of a great potential. More specifically, in the model-driven approach, CMS users will be able to design CMS-based Web applications from a higher level of abstraction, and then, their design models will be turned into running applications, not based on manually performed repetitive, tedious and error-prone development tasks such as configuration and customization, but rather, based on automated model transformations that end-up with automatic code generation. This can significantly increase prototyping speed and eliminate errors that could have been probably derived from the traditional manual development process, while increasing productivity, consistency and overall application quality.

To support the adoption of the model-driven development paradigm in CMS domain, in this paper,

[a] https://orcid.org/0000-0003-1612-2822
[b] https://orcid.org/0000-0002-4073-7256

Gkantouna, V. and Tzimas, G.
CMS-oriented Modeling Languages: An Attempt to Assist Model-driven Development in CMS Domain.
DOI: 10.5220/0008363103720379
In *Proceedings of the 15th International Conference on Web Information Systems and Technologies (WEBIST 2019)*, pages 372-379
ISBN: 978-989-758-386-5

we address a fundamental issue that is currently hindering its realization. This issue refers to the failure of the existing Web modeling languages to capture the development context of CMS platforms, and thus, their inability to provide a modeling framework to support the model-driven development of CMS-based Web applications. This is mainly due to the fact that these languages are generic (they have been designed for the general case of Web applications where development starts from scratch and not based on a preconfigured software platform such as CMSs), and thus, they provide generic and abstract notation which is actually quite far from the standard development context of CMS platforms. This makes it hard for designers to determine how CMS-based Web applications could be modeled within the context of these languages, and subsequently, how the design models could be automatically turned into CMS source code implementation, given that there is not any kind of mapping between their modeling framework and the actual development context of CMSs. To address this problem, we introduce a new genre of modeling languages, called CMS-oriented modeling languages, which are specifically designed for the category of CMS-based Web applications. More specifically, they are designed in such a way so that their modeling framework is in direct correspondence with the particular development context of the target CMS. This serves a twofold purpose. First, it allows for a modeling environment in which designers can produce the design model of CMS-based Web applications in terms of modeling primitives that correspond to the actual elements supported by the target CMS with which they are already familiar with. Second, it facilitates the development of tools to perform the automated interpretation of design models into running CMS-based Web applications, since now there is a clear mapping between modeling and implementation level, facilitating the transition from modeling to source code level. Based on this, we argue that the proposed genre of CMS-oriented modeling languages is a significant first step towards laying the foundations of model-driven development in CMS domain in order to support the automated generation of CMS-based Web applications.

To support the definition of CMS-oriented modeling languages for the various CMS platforms available today, we propose a general framework comprising by three major stages: (i) the analysis of the target CMS platform, (ii) the creation of its domain model including all the key elements of its development context that need to be captured by the CMS-oriented modeling language, and (iii) the formal definition of the CMS-oriented modeling language, i.e., the formalization of its modeling primitives. In this way, the proposed framework supports the definition of modeling languages which have the expressiveness to capture the development context of CMSs, and thus, represent the CMS-based Web applications as models consisting of modeling primitives which are in direct correspondence with the actual CMS-specific elements of the target CMS, making the transition from modeling to source-code level considerably easier and enabling the model-driven development in CMS domain.

To support our case, we present a case study in which we have applied the proposed framework on the popular open-source Joomla! CMS platform, resulting in the definition of a language for modeling Joomla!-based Web applications. The remaining of this paper is organized as follows: Section 2 provides an overview of related work and discusses the contribution of this work. Section 3 presents an overview of the proposed framework, while Section 4 presents in detail the case study for the Joomla! CMS. Finally, Section 5 discusses conclusions and future work.

2 RELATED WORK

Despite the widespread use of CMSs and their relevant acceptance in the global market, the vast majority of Model-Driven Web Engineering (MDWE) methodologies (Rossi et al., 2016) have so far ignored the domain of CMSs and they have been primarily designed to support the development of traditional Web applications in which development starts from scratch and not based on a preconfigured software platform such as CMSs. This makes them inappropriate for supporting model-driven development of CMS-based Web applications. It is only during the last decade that a limited number of MDWE methodologies has emerged for addressing model-driven development in CMS domain. Among them, in (Trias, 2014), authors present a model-driven approach for the automated migration of websites built on the open-source CMS platforms of Drupal, Joomla! and WordPress. In (Souer et al., 2009), authors argue that customizing CMS-based Web applications is a difficult task and that there is a gap between the requirements analysis in this type of Web applications and its implementation. Therefore, they propose a model-driven method to configure automatically a CMS-based Web application based on requirements. In (Saraiva and Silva, 2009), authors present a model-driven method for the development

of CMS-based Web applications based on two modelling languages situated at different abstraction levels, i.e., the CMS Modelling Language (CMS-ML) at a higher abstraction level and the CMS Intermediate Language (CMS-IL) at a lower one. This method is composed of two phases: the modelling phase, where the CMS-based Web application is modelled at two different abstraction levels, and the deployment phase, where the deployment artefacts that implement the CMS-based Web application are generated from the model defined at lower abstraction level in the previous phase. However, in all aforementioned approaches, a major problem is that they all have model transformation issues which hinders their widespread adoption by the CMS community. Since they rely on modeling frameworks which are not directly related to the development context of the target CMS, it is difficult to automatically transform the design model into running CMS-based Web application.

At this point lies the contribution of the proposed framework. Our main goal is to define modeling languages having a modeling framework which is in a 1:1 correspondence with the actual CMS development context, so that the transition from modeling to source-code level will become considerably easier. This clear mapping between modeling and implementation level will significantly facilitate the development of tools to perform the automated interpretation of design models into running CMS-based Web applications, and thus, enable model-driven development in CMS domain.

3 PROPOSED FRAMEWORK

In this section, we present an overview of the proposed framework for supporting the definition of CMS-oriented modeling languages. It is composed of three major stages as depicted in Fig. 1. As one can notice, the first two stages involve (i) the analysis of the target CMS platform and (ii) the creation of its domain model in an effort to determine its particular development context. This is required in order to acquire all the necessary information that will allow us subsequently in the third stage to define a modeling framework for the CMS-oriented modeling language that will be CMS-specific, i.e., it will be composed of modeling primitives which will be in a 1:1 correspondence with the actual elements supported by the CMS development context.

The first stage involves the analysis of the target CMS platform under a number of different viewpoints in order to determine the key elements of its development context. These viewpoints are the following: (i) *CMS architecture* allowing the identification of the standard building blocks (i.e., the basic content item types) provided to developers for composing the pages of CMS-based Web applications, (ii) *CMS database schema* allowing the identification of the underlying data schemas upon which the standard building blocks operate so that we can acquire knowledge on how the modeling elements corresponding to them will be mapped into the development context of the CMS in order to facilitate the transition from modeling to source code level, (iii) *CMS front-end interface design* determining how the target CMS supports the organization of pages in terms of the available content item types, so that we can acquire knowledge on how to conceptualize the page template within the modeling framework of the proposed CMS-oriented modeling language, and (iv) *CMS core features* determining the built-in functionalities provided by the target CMS in order to define modeling primitives for capturing such features.

Subsequently, the second stage involves the creation of the domain model for the target CMS platform, including the detail description of each one of its key elements identified in the previous stage. These elements include the content types supported by the target CMS, the content items defined over these types, the built-in front-end views for publishing these items, etc. For each of these elements, the domain model accommodates a detailed list of their attributes and the relationships existing among them. To collect all these information, a thorough inspection of the CMS back-end administration environment must be performed.

At the end of these two first stages, one can acquire all important information about the development context of the target CMS, providing knowledge on how to shape the modeling framework, (i.e., the various modeling primitives) of the CMS-oriented modeling framework in the next stage so that these two can be in 1:1 correspondence.

Finally, the third stage involves the formal definition of the CMS-oriented modeling language based on the specifications elaborated in the two previous stages. More specifically, for every CMS-specific element identified in the CMS domain model, a corresponding modeling primitive is defined. Their formalization can be realized by using the notation of existing modeling languages (which will be extended for modeling CMS-based Web applications) or by defining a whole new syntax and

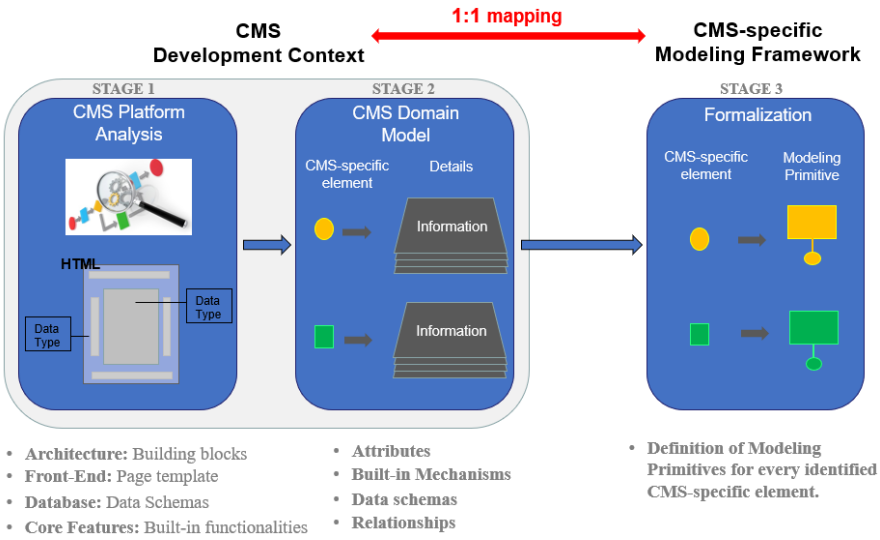

Figure 1: Overview of the proposed framework.

notation. In this way, the proposed framework supports the definition of CMS-oriented modeling languages in which the modeling framework is in a 1:1 correspondence with the particular development context of the target CMS platform, making the transition from modeling to source-code level considerably easier, and thus, facilitating significantly the development of tools for the automated interpretation of design models into running CMS-based Web applications.

In order to exemplify the concepts behind the proposed framework, in the next section, we present its application on the popular open-source Joomla! CMS platform.

4 CASE STUDY: THE JOOMLA! CMS

With more than 3% of the Web running on Joomla! and a CMS market share of more than 6%, Joomla! powers the web presence of millions organizations worldwide and is the second most used CMS platform on the Web. Therefore, to illustrate the potential of our approach, we present the application of the proposed framework on the Joomla! CMS resulting in the definition of a modeling language oriented towards Joomla!-based Web applications.

4.1 Stage 1: Analysis of Joomla! CMS Platform

In this stage, we performed the analysis of Joomla! under the following viewpoints in order to determine the key elements of its development context.

Architecture Viewpoint: The main building blocks provided to web developers for building Joomla!-based Web applications are *components* and *modules*. Components are the main functional units of Joomla!, which are used to render the main content area of the page. Each component has one or more "views" that control how the content is displayed on the main content area of the page. Modules are typically used to render content in the secondary content areas of a page, i.e., peripherally from the main content area. As one can find in (Marriott and Waring, 2013), Joomla! supports various categories of components and modules.

Database Schema Viewpoint: We have identified and recorded all the standard data schemas which are required for supporting the creation, management and operation of the various types of components and modules supported by Joomla!. For each table included in the database schema, we have identified its properties and the relationships existing among them. This information will subsequently guide us on how to properly shape (in terms of attributes and relationships) the various modeling primitives of the Joomla!-oriented modeling language in order to facilitate transition from modeling to implementation level. For example, the display of a menu module on

Content Type	Content Items	Components	Modules
Articles:	Article, Category	Single Article Article Categories Article Category Blog Article Category List Featured Articles Archived Articles	Most Read News Flash Latest Articles Archive Related Articles Article Categories Article Category
Contacts:	Contact, Category	Single Contact Contact Single Category Contact Categories Featured Contacts	
News Feeds:	News Feed, Category	Single News Feed News Feed Category News Feed Categories	Feed Display
Banners:	Banner, Category		Banner
Users:	User, User Group	Login Form Registration Form Username Reminder Request Password Reset	Latest Users Who's Online Login

Figure 2: The key elements of Joomla! CMS.

a page involves 4 different tables of the underlying database. As a result, when we will subsequently define a modeling primitive for representing the menu module, we will know how this modeling element must be mapped into the development context of Joomla!, e.g., on which tables of the underlying database schema a model compiler must create records, in order to automatically create this menu.

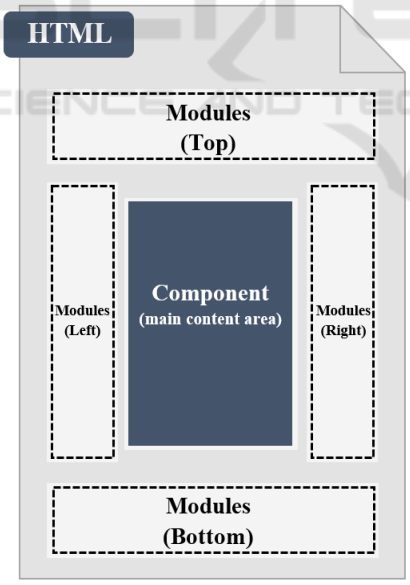

Figure 3: Template of a page in Joomla!-based Web applications.

Front-end Design Viewpoint: The template (i.e., the structure) of pages in Joomla!-based Web applications is depicted in Fig. 2, where one can notice a page is composed by exactly one component rendering content in its main content area and by a number of modules rendering content in its secondary content areas, i.e., in the sidebars or at the top and bottom area of the page. Subsequently in the third stage, we will rely on this template in order to conceptualize the structure of pages within the context of the Joomla!-oriented modeling language.

Core Features Viewpoint: Joomla! provides a large number of built-in features, such as user management, access control, content archiving, RSS feeds, etc. (the detail list can be found in (Marriott and Waring, 2013)). Subsequently, we will take into account all these built-in functionalities during the formalization of the modeling framework of the proposed Joomla!-oriented modeling language, so that it can provide the modeling capabilities for capturing these built-in features too.

4.2 Stage 2: Creation of the domain model for the Joomla! CMS

In this stage, we performed a thorough inspection of the Joomla! back-end administration environment, focusing particularly on the configuration settings used for the creation, management and publishing of its key elements. Fig. 3 presents these elements where one can notice the basic content types supported by Joomla! CMS, the content items defined over these types, and the built-in front-end views (i.e., components and modules) used for publishing these items. For each of these elements, we have identified and recorded the detail list of its properties, the configuration parameters used by the built-in mechanisms supporting its creation, management and publishing, the data schemas upon it operates and its relationships with other elements.

Joomla! Basic Content Items		
Content Item	**Attributes**	**Data Table**
Article	- <**ContentItemCreation**> - <**ContentItemPublishing**> - <**ContentItemMetadata**> - **Title:** article's title - **Category:** the articles category that this article is assigned to - **Introtext:** stores the text of the article up to the Read More break. If there is no Read More break, it stores the entire article text. - **Fulltext:** In case of a Read More break, it stores the full text of the article. - **Featured:** assign article to featured blog layout. - **Access:** access level group allowed to see this item - **Tags:** assign tags to the article - **Hits:** number of hits or the article - **Key Reference:** store information referring to an external resource - **Author:** the author of the article's content - **External Reference:** optional reference used to link to external data sources	**jos_content**

Figure 4: The properties of the Single Article component.

All this information has been documented in the domain model of Joomla! CMS platform which is freely available in (Gkantouna, 2019). An instance of the domain model is presented in Fig. 4 which presents the properties of the "Single Article" component along with a short description of them.

It is important to mention that since the domain model produced in this stage provides a comprehensive documentation of the development context of Joomla! CMS, it can be used as a reference model for the definition of new modeling languages oriented towards the Joomla! CMS platform or even for developing extensions to existing modeling languages for Joomla!-based Web applications.

In the next stage, we rely on the information provided in the domain model in order to produce the definition of the modeling primitives of the proposed Joomla!-oriented modeling language.

4.3 Stage 3: Formal Definition of the Joomla!-oriented Modeling Language

In this stage, we produce the formal definition of the CMS-oriented modeling language based on the specifications elaborated in the two previous stages. More specifically, for every key element previously identified in the domain model of Joomla! CMS, we define its corresponding modeling primitive, having as attributes and relationships all the properties and relationships documented for this element in the domain model. This way, there is a direct correspondence between the modeling primitives and the actual elements specified in the development context of Joomla!.

To formalize the modeling framework of the proposed Joomla!-oriented modeling language, we have adopted the notation of WebML and formalize the specification of the various modeling primitives by means of WebML elements. There are two main reasons for choosing WebML. On one hand, WebML is one of the most popular MDWE approaches so far that is being used in the industry (Rossi et al., 2016). On the other hand, for every key element of Joomla! that need to be captured by the Joomla!-oriented modeling language, WebML allows us to define two equivalent representations of them, as follows:

- an intuitive visual formalism that can be easily communicated among the members of the development team, even by the non-technical users which are actually the common case in CMS domain,
- a textual formalism, i.e., an XML-based textual specification of the captured component or module type, including the detailed list of its attributes as they have been identified in the domain model of Joomla! CMS.

The main benefit of this dual representation of the key elements of Joomla! CMS platform is that it significantly facilitates the subsequent development of a modeling environment to support: (i) the design of Joomla!-based Web applications in terms of intuitive visual representations (formalisms) of the actual components and modules types specified in the development context of Joomla! CMS platform, and (ii) the mapping of the visual design models into automated source-code implementations of Joomla!-based Web applications, simply by providing the corresponding textual formalism of the involved visual formalisms into code generators.

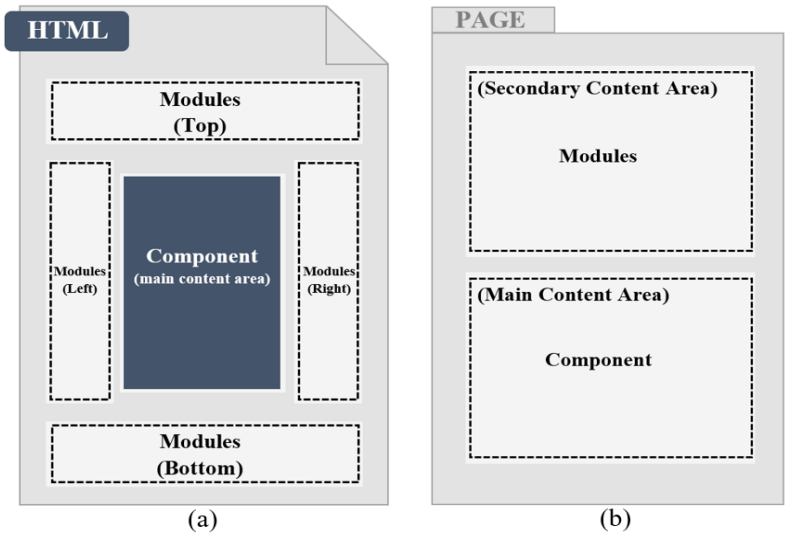

Figure 5: Conceptualization of page structure.

Additionally, it must be also noted that the use of WebML for the definition of modeling primitives that capture the development context of Joomla! CMS results into the extension of WebML with new modeling elements for Joomla!-based Web applications.

4.3.1 Capturing the Standard Database Schema of Joomla! CMS

Prior to the description of modeling primitives, it must be noted that as data model of the proposed Joomla!-oriented modeling language, we have adopted the Entity-Relationship model. More specifically, we have defined an entity for every table included in the standard database schema of Joomla! CMS platform. The relationships among the various tables of the standard Joomla! database are also maintained as relationships among their corresponding entities.

4.3.2 Capturing the Front-end Interface Design of Joomla! CMS

We have relied on the page template described in Fig. 2 and we have conceptualized the structure of pages within the modeling framework of the proposed Joomla!-oriented modeling language as depicted in Fig. 5, i.e., as the union of a "Modules" area containing all the modules displayed on a page and a "Component" area containing the Joomla! component used for displaying the main content of the page.

4.3.3 Capturing Joomla! Key Elements (Components and Modules)

For each type of Joomla! components and modules, we have defined a corresponding modeling primitive based on the information collected for this element in the Joomla! domain model. For instance, Fig. 6 presents the modeling primitive defined for the login module. As one can notice in Fig. 6, there is the front-end presentation of the login module and its corresponding visual and textual formalism in WebML. The detail list of the modeling primitives defined for all the types of Joomla! components and modules can be found in (Gkantouna, 2019).

5 CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed a new genre of modeling languages, called CMS-oriented modeling languages, which are especially defined for the category of CMS-based Web applications. We have also presented a framework to support their defini-tion in three main stages. The first stage involves the analysis of the CMS platform in order to determine its particular development context. Then, the second stage involves the creation of the domain model for the target CMS, including all the key elements of its development context that need to be captured by the CMS-oriented modeling language. Finally, the third involves the definition of the CMS-oriented modeling language, i.e., the formalization of its modeling primitives, based on the specifications elaborated in

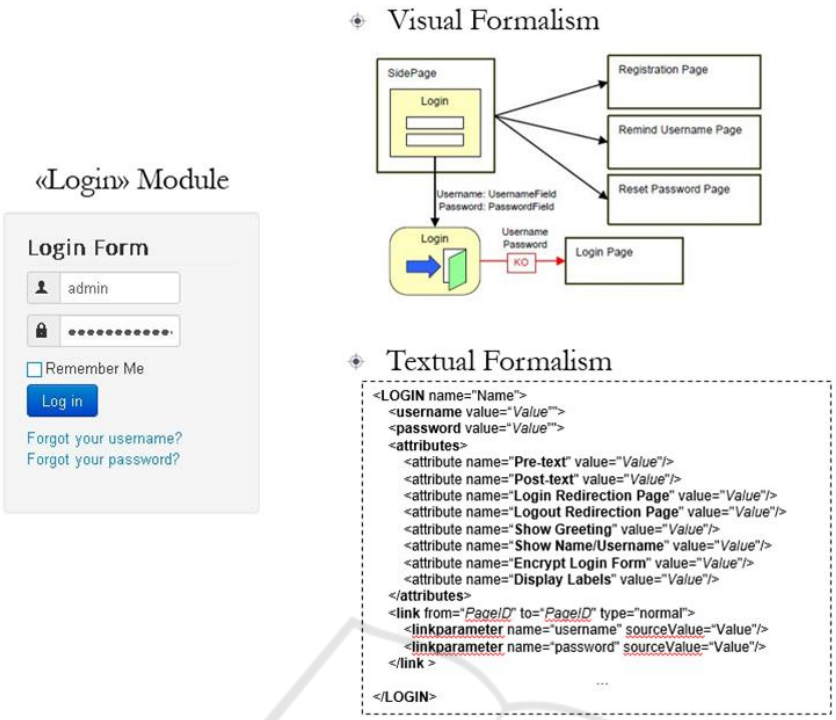

Figure 6: Modeling primitive (visual and textual formalism) for the login module.

the previous two stages. In this way, the proposed CMS-oriented modeling languages have the expressiveness to represent CMS-based Web applications as models consisting of modeling elements which are in direct correspondence with the elements specified in the actual development context of the target CMS, making the transition from modeling to source-code level considerably easier, enabling the model-driven development in CMS domain. To support our case, we have applied the proposed framework on the popular open-source Joomla! CMS platform, resulting in the definition of a modeling language oriented towards Joomla!-based Web applications.

In near future, we are experimenting on developing model compilers which given as input models specified in the proposed Joomla!-oriented modeling language, they can automatically transform them into source-code implementation of running Joomla!-based Web applications. We also plan to develop CMS-oriented modeling languages for other popular CMS platforms, such as Drupal and WordPress.

REFERENCES

Gkantouna, V. Content Management Systems Modeling, https://alkistis.ceid.upatras.gr/research/modeling/patterns, last accessed 2019/02/01.

Gkantouna, V., Tsakalidis, A.K., Tzimas, G.: Automated Analysis and Evaluation of Web Applications Design: The CMS-based Web Applications Case Study. WEBIST 2016: 130-139

Marriott, J., Waring, E.: The Official Joomla! Book, 2nd ed., Boston: Addison-Wesley Professional, 2013

Rossi, G., Urbieta, M., Distante, D., Rivero, J.M., Firmenich, S., “25 Years of Model-Driven Web Engineering: What we achieved, what is missing”, Clei Electronic Journal, Vol. 19, No. 3, (2016)

Saraiva, J. D. S., & Silva, A. R. DA.: Development of CMS-Based Web-Applications Using a Model-Driven Approach. In 4th International Conference on Software Engineering Advances (pp. 500–505). doi:10.1109/ICSEA.2009.79, (2009)

Souer, J., Kupers, T., Helms, R., & Brinkkemper, S.: Model-Driven Web Engineering for the Automated Configuration of Web Content Management Systems. In Springer-Verlag (Ed.), ICWE 2009 (pp. 124–135). Heidelberg.

Trias, F.:“An ADM-Based Method for Migrating CMS-Based Web Applications”, PhD thesis, Universidad Rey Juan Carlos, (2014).

Web Technology Surveys, Historical yearly trends in the usage of content management systems for websites, https://w3techs.com/technologies/history_overview/content_management/all/y, last accessed 2019/02/01.