An ADM-based Method for Migrating CMS-based Web Applications: Extracting ASTM Models from PHP Code

Feliu Trias, Valeria de Castro, Marcos López-Sanz and Esperanza Marcos

Grupo de Investación Kybele, Universidad Rey Juan Carlos,
C/Tulipan, s/n. 28933 Móstoles, Madrid, Spain

Abstract. In recent years, Architecture-Driven Modernization (ADM) is gaining increasing acceptance in software reengineering processes of existing systems. It can help reduce reengineering costs by automating the activities, such as extracting models from the source code. This is specially crucial in the reengineering of CMS-based Web applications. At time of writing there are no methods that could be used in that context. Hence, we define an ADM-based method for migrating CMS-based Web applications. In the context of this method, we present in this paper the implementation of the text-to-model (M2T) transformations to extract models from PHP code. These models conform to Abstract Syntax Tree Metamodel (ASTM) a standard metamodel proposed by ADM. To implement these transformations we performed three activities: 1) definition of a PHP grammar, 2) mapping PHP grammar elements to elements of ASTM and 3) implementation of a model extractor. To show the feasibility of our approach we use a real example of PHP code from a CMS-based Web application implemented in Drupal.

1 Introduction

In recent years, the necessity for migration of existing systems to other technological platforms has become one of the major problems faced by organizations since these migration activities are carried out following ad-hoc software reengineering processes, thus entailing expensive costs and high risks for organizations [1].

To solve this, the Object Management Group (OMG) proposes the Architecture-Driven Modernization (ADM) an initiative which advocates for the application of MDA (Model-Driven Architecture) [2] principles to formalize the software reengineering process. ADM provides several benefits such as reducing development and maintenance costs and extending the life cycle of the legacy systems [3]. ADM develops seven standard metamodels to represent the information involved in a software reengineering process, but only three of them are available: *Abstract Syntax Tree Metamodel* (ASTM) [4], *Knowledge Discovery Metamodel* (KDM) [5] and *Structured Metrics Metamodel* (SMM) [6]. These metamodels allow developers to manage software reengineering processes in an integrated and standardized manner as well as saving them time and effort creating their own metamodels [7]. In this paper we focus on the ASTM. This metamodel allow to represent the syntax of the source code.

Trias F., de Castro V., López-Sanz M. and Marcos E..
An ADM-based Method for Migrating CMS-based Web Applications: Extracting ASTM Models from PHP Code.
DOI: 10.5220/0004602900850092
In *Proceedings of the 1st International Workshop in Software Evolution and Modernization* (SEM-2013), pages 85-92
ISBN: 978-989-8565-66-2

Models conform to ASTM allows to capture most of the knowledge from the source code and reduce the complexity to generate models at higher abstract level, such as KDM models.

In the last years, the volume of digital content managed by organizations has increased dramatically. To solve it, organizations have based their Web applications on Content Management Systems (CMS) since they are platforms which allow users to collect, manage and publish content in a robust and reliable manner [8].

The CMS market is constantly evolving and organizations experiment the necessity of migrating their CMS-based Web applications to another CMS or to a new version of the same CMS because their current one has become obsolete and it does not meet their needs. Therefore, they find it necessary to start a process of reengineering. In previous works [9] we have conducted a systematic literature review to assess the state of the art on CMS-based Web applications in the context of MDA. One of the main conclusions derived from such review was that there are no methods that could be used for migrating CMS-based Web applications. Accordingly, we defined an ADM-based method for migrating this kind of Web applications. Up to now, our method is focused on open-source CMS such as Drupal [10], Joomla! [11] or Wordpress [12] because of their features and their spread use and relevant acceptance in the market [13]. Most of these CMSs are implemented in PHP.

The work presented is framed in the first phase of our method where we extract models from the source code by means of text-to-model (T2M) transformations. In this paper we present the implementation of these transformations which extract models from PHP code. The models obtained conform to ASTM (ASTM models). To the best of our knowledge, there are no ADM tools for the extraction of models from PHP code. As such, there is any proposal for the generation of ASTM models.

To implement the T2M transformations we have carried out three activities: 1) definition of the PHP grammar, 2) mapping of PHP grammar elements to elements of ASTM, and 3) implementation of a model extractor. We use a real example of PHP code from a CMS-based Web application implemented in Drupal to show the feasibility of our approach.

The rest of this paper is organized as follows: Section 2 explains the related works. Section 3 presents our ADM-based method for migrating CMS-based Web applications. Section 4 explains the activities performed to implement the model extractor and we illustrate the explanation with a real example of PHP code and, finally, Section 5 presents the conclusions and future works.

2 Related Works

In this section we present some of the ADM-based approaches found in the literature and we compare them with our method for migrating CMS-based Web applications. Due to space limitations only we present the most representative ones.

Krause et al. present in [14] *DynaMod,* a method which addresses model-driven modernization of software systems. It considers static and dynamic analysis for extracting models conform to KDM from Java code. Sadovykh et al. present in [15] a method to extract an UML (Unified Modeling Language) model containing the most

persistent part of a system from C++ code. Perez-Castillo et al present in [16] a reengineering process called PRECISO to recover and implement Web Services in automatic manner from relational databases. They extract a model conform to a SQL-92 metamodel from SQL-92 code. Bruneliere et al present in [17] *MoDisco* an extensible approach for model-driven reverse engineering which allows extracting platform models from Java, XML and JSP. They define a metamodel for each code. Reus et al present in [18] a reverse engineering process for recovering UML models from PL/SQL code. Vasilecas et al presents in [19] a process which derives business rules from a legacy system. They extract ASTM models from Visual Basic code. Table 1 we compare the ADM-based approaches presented with our method.

Table 1. Related works.

Approach	Source code	Metamodel	Context
Krause et al present (DynaMod)	Java	KDM	Legacy systems
Sadovykh et al	C++	UML	Legacy systems
Perez-Castillo et al (Preciso)	SQL-92	SQL-92	Data base / Web services
Bruneliere et al (Modisco)	Java, XML and JSP	Java, XML and JSP	Legacy systems
Reus et al	PL/SQL	UML	Data base
Vasilecas et al	Visual Basic	ASTM	Legacy systems
Our ADM-based method	PHP	ASTM	CMS-based Web applications

The second column in Table 1 refers to the source code from which the approach extracts these models. As we can observe none of them addresses the extraction of models from PHP code. The third column refers to the metamodel which the extracted models conform to. On the one hand, two of the approaches bet for defining their own metamodels such as Perez-Castillo et al with the SQL-92 metamodel and Bruneliere et al with Java, XML and JSP metamodels. On the other, hand two approaches use the UML metamodel. Although UML is a standard, it is not proposed by ADM. Finally, two of the approaches use the ADM standard metamodels. Krause et al use KDM and Vasilecas et al extracts models conform to ASTM. The third column specifies the context of the approach. If the approach is framed in a generic reengineering context we categorize it as a *legacy systems* context. Two of the approaches are focus on *data base* context and one of them also in *web services* context. None of them is framed in the context of CMS-based Web applications.

3 Research Context: an ADM-based Method for Migrating CMS-based Web Applications

As we said in the introduction our work is focused in the implementation of the T2M transformations framed in an ADM-based method for migrating CMS-based Web applications. This method is composed of three phases: a) reverse engineering phase, b) restructuring phase and c) forward engineering phase. In this section we present those phases and their tasks. Fig. 1 outlines the whole ADM-based migration method. In it we have highlighted with dashed lines the scope of the work presented in this paper.

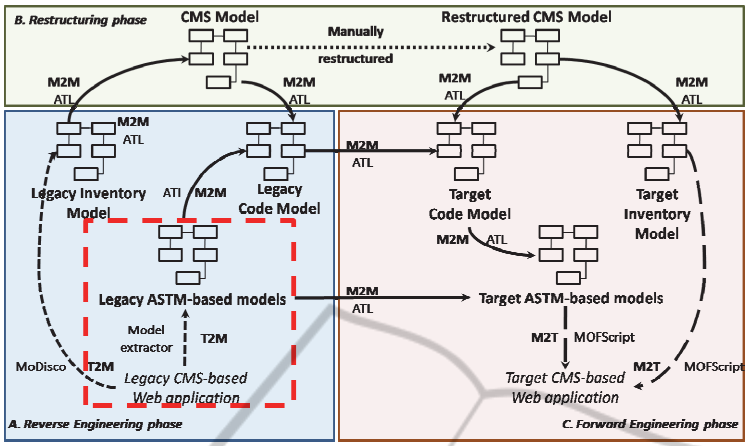

Fig. 1. ADM-based method.

A. Reverse Engineering Phase, this phase is composed of three tasks: i) *knowledge extraction*, the extraction of ASTM models from PHP code (the work presented). ASTM allows us to represent the code in a proper and non-ambiguous way and reduce the complexity to generate the KDM models which are at higher abstract level; ii) *the generation of the KDM models*, from ASTM models we automatically generate KDM models. The two KDM models we generate are: Code Model and Inventory Model. It is worth noting that we obtain the Inventory Model from the legacy code by using MoDisco tool [17], iii) *the generation of the CMS Model*, using M2M transformations we generate automatically the CMS Model from the information captured in KDM models. This CMS Model conforms to the CMS Common Metamodel presented in [20].

B. Restructuring Phase, the CMS Model are manually restructured by the developer taking into account the specific features of the target CMS platform.

C. Forward Engineering Phase, this phase defines the top-down development process comprised of the next three tasks: i) *generation of the target KDM models*, from the restructured CMS Model we generate the target Code Model and the target Inventory Model that represent the implementation of the target CMS-based Web application; ii) *generation of the target ASTM model*, we generate the target ASTM model from the target Code Model and the target Inventory Model and iii) *code generation*, we generate the software artifacts that compose the architecture of the target CMS-based Web application (folders and file skeletons) and the code that implements them.

4 Implementation of the T2M Transformations

In this section we explain the activities for the implementation of the T2M transformations to extract ASTM models from PHP code which is the focus of the work that we present in this paper.

We use a real example of PHP code from a CMS-based Web application implemented in Drupal to illustrate the different activities performed in this implementation and to show the feasibility of our approach.

4.1 Definition of the PHP Grammar

The first activity is to define the PHP grammar using EBNF language [21]. It was one of the most tedious and time-consuming activities because we had to resolve the left-recursivity conflicts among the elements defined in the grammar. These elements are classified into two groups: 1) expressions (such as *logicalOr* or *function call*) and 2) statements (such as *assignment* or *conditional*).

Expressions are the cornerstone of the PHP language since they represent those elements which evaluate to a certain value, e.g. arithmetic expressions such as *$var+5+3* which evaluates to a number or logical expressions such as *$var or $tar* which evaluates to a logical value. Fig. **2** shows the correspondence of a function definition implemented in PHP with its specification in the defined EBNF-based PHP grammar.

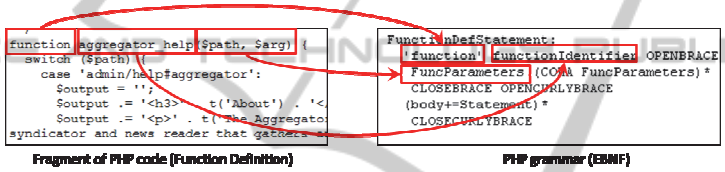

Fig. 2. Correspondence of PHP code with EBNF-based PHP grammar.

From the PHP grammar and using the Xtext framework we obtain automatically three artifacts: 1) a metamodel, 2) a textual editor and 3) a parser implemented in Java that allow us to recognize the elements of the PHP grammar from code written in PHP. We decided to use Xtext framework because this parser facilitates us the implementation of the model extractor (third activity) since we use the methods implemented in this parser to recognize the PHP elements that will be mapped to the ASTM model.

4.2 Mapping PHP Grammar Elements to Elements of ASTM

In the second activity we define the mappings between the elements of the PHP grammar and the elements of ASTM. The definition of these mappings are necessary to implement the model extractor in the third activity. As we can see in Table 2 a *VariableDefStatement* of the PHP grammar ($var=3;) is mapped to a *VariableDefinition* of the ASTM or a *Addition* expression in the PHP grammar (9+3) is mapped to a *BinaryExpression*.

At the time of defining these mappings we realized that some elements from the PHP grammar cannot be mapped to elements of ASTM. For that reason, we extended the ASTM with the specific elements of the PHP code. Some of these elements are: *xor* operator (xor)**,** not *identical* operator (!==), *supressWarning* operator *(@)* or

instance of operator (instanceof).

Table 2. Mapping PHP grammar elements to ASTM elements.

Group	PHP Grammar element	ASTM element
Statements	VariableDefStatement	VariableDefinition
	FuncDefStatement	FunctionDefinition
Expressions	Addition	BinaryExpression
	FunctionCall	FunctionCallExpression

Otherwise, we needed to redefine existing elements of ASTM to make the mapping possible. For example, we had to redefine the attribute *condition* of the *ForStatement* element. This attribute is defined as a required and specifies the condition of a *for* statement. We had to redefine it as optional to allow map the *for* statements without condition permitted in PHP to the *ForStatement* element in ASTM.

Other redefined elements are: *swithStatement*, *compilationUnit* and *arrayAccess*. Due to space limitations we do not explain each of them.

4.3 Implementation of a Model Extractor

Finally, in the third activity we implement the model extractor to obtain ASTM models from PHP code. This model extractor is implemented in Java. For its implementation we use: 1) on the one hand, the parser obtained by Xtext in the first activity to recognize the syntax elements from code written in PHP, 2) on the other hand, the API in Java obtained automatically from ASTM by using the Eclipse Modeling Framework (EMF) [22], to generate the elements of the ASTM models.

Fig. **3** shows how our model extractor identifies a function definition written in PHP and extracts a *FunctionDefinition* element in the ASTM model. As we can see in Fig. **3** the fragment of PHP code conforms to the PHP grammar defined in EBNF and the model generated conform to ASTM.

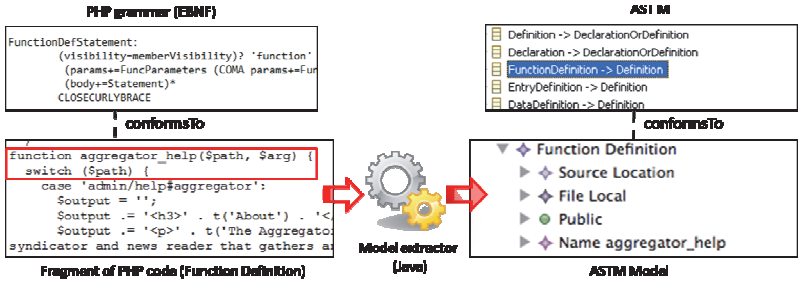

Fig. 3. Extracting a function definition in PHP to ASTM model.

5 Conclusions and Future Works

Regarding the advantages that ADM provides to software reengineering processes

and after concluding that there are no ADM-based methods focused on the migration of CMS-based Web applications, we consider interesting to define an ADM-based method for migrating this kind of Web applications. The method that we define is focused on Web applications based on open-source CMS such as Drupal, Joomla! or Wordpress which are implemented in PHP.

The work presented is framed on the first phase of this ADM-based method where we extract models from the source code by means of text-to-model (T2M) transformations. In this paper we presented the implementation of these T2M transformations which extract ASTM models from PHP code.

To implement these transformations we performed three activities: 1) definition of the PHP grammar, 2) mapping PHP grammar elements to elements of ASTM, and 3) implementation of a model extractor.

We consider the first activity as one of the most tedious and time-consuming because of the necessity of resolving the left-recursivity conflicts among the elements of the PHP grammar. From the PHP grammar and by using Xtext framework we obtained automatically: 1) a metamodel, 2) a textual editor and 3) a parser implemented in Java that allow us to recognize the elements of the PHP grammar from code written in PHP. During the second activity we needed to extend the ASTM with specific elements of the PHP code such as *xor* or *not identical* operators. Otherwise, we redefined some of the existing elements of ASTM such as *ForStatement*, since their standard definition hindered the mapping. For the implementation of the model extractor in the third activity, we used the parser of the PHP grammar obtained in the first activity and the API in Java of ASTM generated by executing EMF. Both, the parser and the API facilitated us the task of implementation.

According to the related works we can conclude that there is no any ADM approach to extract models from PHP code as well as there are a few approaches for the generation of ASTM models. We think that the low use of ASTM is because the last and unique version (1.0) was submitted rather recently, in January 2011.

After our experience with ASTM we can state that this metamodel has saved us time and effort creating our own metamodel and has met satisfactory our necessities.

As future works we consider the refining of the model extractor and the implementation of the remaining phases of our ADM-based method for migrating CMS-based Web applications.

Acknowledgements

This research has been partially funded by the Project MASAI (TIN-2011-22617) from the Spanish Ministry of Science and Innovation.

References

1. Chikofsky, E. J., Cross, J. H.: Reverse Engineering and Design Recovery: A Taxonomy. IEEE Software. 7, 13-17.
2. Mellor, S. J., Scott, K., Uhl, A., Weise, D.: Model-Driven Architecture. In: (Eds.), J.-M.B.

and Z. B. (ed.) Advances in Object-Oriented Information Systems. pp. 233-239. Springer-Verlag Berlin Heidelberg 2002 (2002).
3. Pérez-castillo, R., García, I., Guzmán, R. D., Caivano, D., Piattini, M.: Database Schema Elicitation to Modernize Relational Databases. 14th International Conference on Enterprise Information Systems. pp. 126-132. Springer Berlin, Wrocław, Poland (2012).
4. Abstract Syntax Tree Metamodel, http://www.omg.org/spec/ASTM/1.0.
5. Pérez-Castillo, R., de Guzmán, I.G.-R., Piattini, M.: Knowledge Discovery Metamodel-ISO/IEC 19506: A standard to modernize legacy systems. Computer Standards & Interfaces. 33, 519-532 (2011).
6. Structured Metrics Metamodel, http://www.omg.org/spec/SMM/.
7. Cánovas Izquierdo, J. L., García Molina, J.: Extracting models from source code in software modernization. Software & Systems Modeling. 1-22 (2012).
8. Boiko, B.: Understanding Content Management. Bulletin of the American Society for Information Science and Technology. 28, 8-13 (2001).
9. Trias, F., De Castro, V., López-Sanz, M., Marcos, E.: A Systematic Literature Review on CMS-based Web Applications. ICSOFT (2013).
10. Drupal CMS, http://drupal.org/.
11. Joomla! CMS, http://www.joomla.org/.
12. Wordpress CMS, http://wordpress.org/.
13. Shreves, R.: Open Source CMS Market Share. , Bali, Indonesia (2011).
14. Krause, H., Porembski, M., Stahl, T., Steinkamp, M., Wittm, N., Straße, A.: DynaMod Project: Dynamic Analysis for Model-Driven Software Modernization. Engineering. pp. 1-2.
15. Sadovykh, A., Vigier, L., Hoffmann, A., Grossmann, J., Ritter, T., Gomez, E., Estekhin, O.: Architecture Driven Modernization in Practice: Study Results. 2009 14th IEEE International Conference on Engineering of Complex Computer Systems. pp. 50-57. Ieee (2009).
16. Perez-Castillo, R., de Guzman, I.G.-R., Caballero, I., Polo, M., Piattini, M.: PRECISO: A Reverse Engineering Tool to Discover Web Services from Relational Databases. 16th Working Conference on Reverse Engineering. pp. 309-310. Ieee (2009).
17. Bruneliere, H., Cabot, J., Jouault, F., Madiot, F.: MoDisco: A Generic And Extensible Framework For Model Driven Reverse Engineering. International conference on Automated software engineering - ASE '10. pp. 173-174 (2010).
18. Reus, T., Geers, H., Deursen, A.V.: Harvesting Software Systems for MDA-Based Reengineering. Second European Conference, ECMDA-FA. pp. 213-225 (2006).
19. Vasilecas, O., Normantas, K.: Deriving Business Rules from the Models of Existing Information Systems. 95-100 (2011).
20. Trias, F.: Building CMS-based Web Applications Using a Model-driven Approach. Sixth International Conference on Research Challenges in Information Science (RCIS). pp. 1 - 6 (2012).
21. ISO/IEC 14977:1996 - EBNF, http://www.iso.org/iso/catalogue_detail.htm?csnumber=26153.
22. Budinsky, F.: Eclipse Modeling Framework, (2008).