A Model-Driven Process to Migrate Web Content Management System Extensions

Dennis Priefer[1,2], Peter Kneisel[2], and Gabriele Taentzer[1]

[1] Philipps-Universität Marburg, Germany
{taentzer}@mathematik.uni-marburg.de
[2] KITE - Kompetenzzentrum für Informationstechnologie, Technische Hochschule Mittelhessen, Germany
{dennis.priefer,peter.kneisel}@mni.thm.de

Abstract. Developing and maintaining software extensions for Web Content Management Systems (WCMSs) like Joomla, WordPress, or Drupal can be a difficult and time consuming process. This poster presents a model-driven process which addresses typical challenges during the migration of software extensions for WCMSs. We introduce JooMDD as a prototypical environment for the development and maintenance of Joomla extensions. JooMDD consists of a domain-specific modelling language for WCMS extensions, a reverse engineering tool to create models based on existing WCMS extensions, and a code generator for software extensions, which can be used to enrich Joomla-based applications. The use of JooMDD within our research demonstrates the application of a model-driven migration process for WCMS extensions.

Keywords: Model-Driven Development, Web Content Management Systems, Joomla, Software Extensions, Software Migration

1 Introduction

Today's web mainly consists of dynamic web applications, often instances of web content management systems like Joomla, WordPress, and Drupal. These systems provide the necessary functionality for administrators to create these instances independent of their experience in web development.

One of the biggest advantages of using a WCMS as platform for dynamic websites is the functional extensibility through standardised extension types. Through the use of APIs, extensions can be implemented without changing the platform itself. The dependency on these APIs has a large impact on the maintenance of developed extensions. If the API of the underlying WCMS platform changes, e.g. through a new major release, extensions must be changed as well. Normally, developers have to change their extension's code by hand. This can lead to inconsistencies within the extension, if the responsible developers do not meticulously update all altered dependencies. Consequently, if developers have to migrate multiple extension, the overall effort can increase exponentially. The same applies to both platform internal and platform external extension migration.

Due to the amount of schematically recurring code in standard WCMS extensions, independent of their underlying platform, we propose a *model-driven* approach to migrate WCMS extensions faster and more easily in comparison to manual migration. In addition, we obtain the typical benefits of model-driven approaches, such as reusability and enhanced code quality.

Using a model-driven approach for software migration on a higher abstraction level is seen as a promising approach in today's research. In [4] the authors introduce a meta-model for the definition of migration processes. These processes are based on model-driven sub-processes as are ours. The reengineering method and reverse engineering tool as presented in [5] deal with the migration of complete CMSs. As our work progresses, we plan to incorporate these approaches into our research and check their suitability for the migration of WCMS extensions. In [3] we consider similar model-driven approaches such as presented in [1], whereby existing work deals with complete systems, web applications in general, or the data of concrete WCMS instances and not the migration of their extensions' code.

Our main contribution is a **concept for a model-driven migration process explicitly for WCMS extensions** and prototypical tools to migrate Joomla extensions. Our set of tools encompasses:

– a domain-specific modelling language for the abstract description of WCMS extensions,
– a transformation tool for creating models from existing Joomla extensions (supporting usual extension types like components, modules, plugins, and libraries),
– a code generator for installable Joomla extensions based on an extension model

2 Model-Driven Migration Process

As an alternative to manually performed migration of the source code, a model-driven migration process of WCMS extensions is done at a higher abstraction level. This allows the use of common model-driven engineering practices, like model refactoring for improving the software quality.

Figure 1 illustrates the migration process of installable WCMS extensions which is divided into three main steps (cf. [2] and [5]):

Reverse Engineering: Existing extension packages (code) are used as input for an automated text-to-model (*T2M*) transformation. This transformation should be as complete as possible. To consider individual code fragments, we suggest to create code models, which contain platform-specific code fragments and can be bound to abstract extension models.

Model Migration: Models can be refactored, extended, or migrated to models based on differing modelling languages through model-to-model (*M2M*) transformations. These transformations can be performed semi-automatically (e.g. model refactoring), or manual (e.g. model extension).

Forward Engineering: Through an automated model-to-text (*M2T*) transformation, models can be transformed to software code, in our case to installable WCMS extensions.

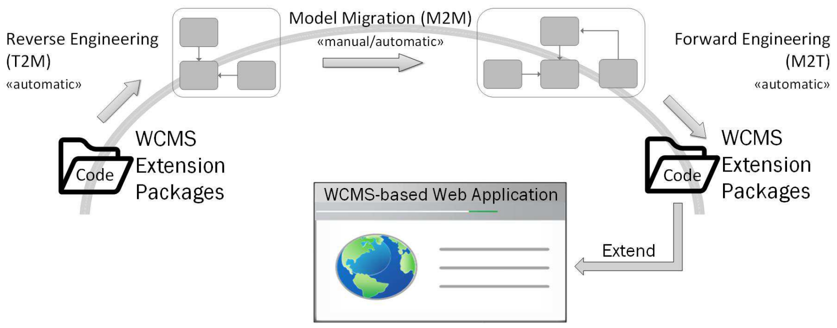

Fig. 1. Model-Driven Migration Process

3 Prototypical Realisation

To test our approach, we have developed an environment (**JooMDD**) for the model-driven migration of Joomla[3] 3.x extensions. We provide the *domain-specific language (DSL)* **eJSL**, which divides a Joomla extension into ***entities*** (data model), ***pages*** (views), and ***extensions*** (extension structure and meta data). For the creation and maintenance (or reengineering) of eJSL models, we provide plugins for the current development environments (IDEs) Eclipse, IntelliJ IDEA, and PHPStorm. These plugins consist of textual editors which support the modellers with features such as auto completion and model validation.

In addition, we developed the prototype **jext2eJSL** to support reverse engineering by a model extractor. The tool uses existing Joomla 3.x extensions (PHP, HTML, JavaScript, and SQL files) as input and creates a domain model based on the eJSL language. It supports the common Joomla extension types (*components*, *modules*, *plugins*, *libraries*, and *templates*), on the conditions that they follow the Joomla coding standards and use the typical design pattern for the particular extension type (e.g. *Model-View-Controller* within components or *Observer* within plugins), meaning that jext2eJSL searches for prescribed file and code schemes expected by the Joomla platform. In order to support WordPress and Drupal extensions, we plan to incorporate the tools presented in [5]. Even though these tools were developed to migrate web-based to WCMS-based applications, we believe the parsers could be further developed to handle the code of WCMS extension as well.

For the forward engineering step we created a **code generator** for installable extension packages for Joomla 3.x. If the Joomla platform is changed, the generator has to be updated, but because of the nature of model-driven approaches,

[3] We selected Joomla as the target platform, because it is one of the most widely used WCMSs (according to `http://w3techs.com/technologies/overview/contentmanagement/all`).

most parts of the models can be reused. In [3] we demonstrate the forward engineering process. The demonstration describes how eJSL-based models can be created within current IDEs (IntelliJ, PHPStorm, and Eclipse) and how generated extensions can be used within existing Joomla-based websites. A video can be found at `https://youtu.be/Uy_WBIjPldI`.

We applied our process, consisting of an automated model extraction, a semi-automated model reengineering, and a code generation, to reengineer conventionally developed Joomla 3.x extensions. Since the extensions already existed for the current platform, this is not a migration in the usual sense. However, the application of our approach ensures that the resulting extensions are in compliance with the quality standards implicit in the process. This step is an initial investment for further migrations.

4 Perspective

We plan to abstract the modelling language and simplify the migration process as much as possible to support further WCMSs and provide a simpler integration of our approach into existing development processes.

To ensure the correctness and usefulness of our approach we will test it extensively by migrating actual WCMS extensions. We expect the next major release of Joomla (version 4.x) within the next two years. Since we are developing and maintaining different types of Joomla extension for the academic sector, we have an adequate set of reference extensions for testing our proposed migration process. Our intention is to reduce the effort involved in the migration of the reference extensions to the new Joomla version as much as possible. In addition we plan an empirical evaluation of our approach to assess the model-driven migration speed and the quality of the migrated extensions.

References

1. M. Brambilla. *Interaction flow modeling language: Model-driven UI engineering of web and mobile apps with IFML.* Morgan Kaufmann, Waltham, MA, 2015.
2. S. Demeyer, S. Ducasse, and O. M. Nierstrasz. *Object-oriented reengineering patterns.* The Morgan Kaufmann Series in Software Engineering and Programming. Morgan Kaufman Publishers, San Francisco, 2003.
3. D. Priefer, P. Kneisel, and G. Taentzer. *JooMDD: A Model-Driven Development Environment for Web Content Management System Extensions.* In *ICSE Companion '16: Companion Proceedings of the 38th International Conference on Software Engineering*, page in press, New York, NY, USA, 2016. ACM.
4. F. J. B. Ruiz, Ó. S. Ramón, and J. G. Molina. *Definition of processes for MDE-based migrations.* In *PMDE '13: Proceedings of the Third Workshop on Process-Based Approaches for Model-Driven Engineering*, pages 1–9, New York, NY, USA, 2013. ACM.
5. F. Trias, V. de Castro, M. López-Sanz, and E. Marcos. *RE-CMS: a reverse engineering toolkit for the migration to CMS-based web applications.* In *SAC '15: Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 810–812, New York, NY, USA, 2015. ACM.