MONAD: A Model Driven Software Product Line to Create Domain Specific Websites

Juan David Villa, Jaime Chavarriaga and Rubby Casallas
Grupo de Construcción de Software, Universidad de los Andes, Bogotá, Colombia

Keywords: Model Driven Web Engineering, Web Content Management Systems, Templates.

Abstract: Websites in a specific domain, i.e., restaurants or product catalogs, share various commonalities. Besides the vocabulary, they can share aspects such as sitemap structure or presentation elements. Web development companies can take advantage of these commonalities to create reusable assets, in the process of website construction. We present MONAD, a software product line to efficiently build websites for specific business domains. MONAD introduces the concept of Domain Template, an asset intended to parametrize websites in terms of their specific domain concepts. Using model-driven techniques, domain templates allow users to reuse presentation elements, sitemap and pages structures, and automate the insertion of content into a particular website. Domain templates are implemented using platform independent models; therefore they can be used to create websites on different web content management systems, i.e., Joomla! or wiki systems.

1 INTRODUCTION

To build a website, a company must design the website's visual appearance, define its content and finally create the individual pages. Even when wikis or WCMS are used as platforms to build the website in a more flexible and manageable way, these tasks are still time consuming and the resulting presentation, sitemap, content and structure are highly coupled.

As a result, Graphical design companies, such as TemplateMonster and Allwebco, offer *presentation templates* to reuse graphic designs. Hosting companies such as Google Sites and Weebly offer *templates* that in addition to the graphic design, include as well sitemaps and sample pages facilitating the reuse of predesigned website structures for a specific business domain.

However, a current limitation of templates is that website creators manually customize the template by modifying the predefined structures and manually insert the concrete data of the website into the template. Furthermore, the customized templates are specific to a platform and cannot be used with other technologies or hosting providers.

(Martinez et al., 2009), (Souer et al., 2011) have proposed the creation of *software product lines* to implement websites in a more efficient and effective manner. Our work is in the same track but it proposes a different approach to build the product line. Our main contributions are the diverse configuration possibilities to create concrete websites.

The rest of this paper is organized as follows. Section 2 presents related work on website creation approaches. Section 3 presents MONAD, our model driven product line approach to create websites on the same domain. And section 4 presents conclusions and further work.

2 RELATED WORK

There are several model driven web engineering (MDWE) approaches that propose the use of models to generate database-based web applications using platforms such as Java or PHP. In general terms, these approaches: (1) separate concerns in diverse platform independent models, (2) define a strategy to integrate them, and (3) transform them into platform specific models to finally produce the code. (Moreno et al., 2008)

There are fewer works in the case of model-driven approaches for building websites on Web Content Management Systems (WCMS) such as Wikis, Joomla! or Wordpress. The current approaches take inspiration from the MDWE works.

There are at least two major challenges for these approaches: 1) Populate a website from existing business domain data. 2) Decouple the website from

Villa J., Chavarriaga J. and Casallas R..
MONAD: A Model Driven Software Product Line to Create Domain Specific Websites.
DOI: 10.5220/0004368400870092
In *Proceedings of the 9th International Conference on Web Information Systems and Technologies* (WEBIST-2013), pages 87-92
ISBN: 978-989-8565-54-9

an specific WCMS. For the first challenge, (Díaz et al., 2011) propose a solution to harvest models from databases. For the latter, (Souer et al., 2011) propose the creation of a website platform independent model and then, a transformation to an specific WCMS. As a business domain model is not used in this approach, the insertion of the content in the website is not covered, requiring a manual process once the website has been deployed into a WCMS platform.

(Diaz and Martin, 2009) define a domain model and using transformation rules, generate a website Platform Specific Model (PSM). Because platform-specific models are used, websites are not generated on different platforms. Additionally, the structure and presentation elements are specified inside the transformation's rules. When a different structure for the website is needed, a web designer must define new transformation rules resulting in difficult changes to be implemented.

The strategy of creating a model driven product line is present in (Martinez et al., 2009) (Souer et al., 2011). We follow the same track but, as we show in the next section, our approach is different to theirs in the way we build the assets of the product line during *domain engineering* and in the way we derive a concrete website during *application engineering*. The main contribution of our work is the diverse configuration possibilities that we provide to the website creator. These include the flexibility of choosing the structure of the website, the presentation and the WCMS platform.

3 MONAD

Our framework presented in Figure 1, is a model driven software product line (MD-SPL) used to generate websites. This SPL uses a set of model-based assets to generate websites supporting variability on structure, presentation and the WCMS platform.

MONAD defines two main processes for an SPL: (1) *Domain Engineering*, where the domain expert and website designer define the website's variability and the assets that allow to implement it, creating elements 1, 2, and 3 of figure 1; and (2) *Application Engineering*, where a website creator defines a business domain model and configures the variability to derive a set of websites using the assets defined in the domain engineering, performing steps 4, 5 and 6 of figure 1.

To illustrate the processes for domain and application engineering in MONAD, we present an SPL for generating websites for product catalogs. This case study was introduced by (Diaz and Martin, 2009) to illustrate model-driven website generation. We extend the case study to include website variability and present our approach.

MONAD supports three types of variability: (1) website structure, (2) website presentation, and (3) WCMS platform.

Variability on website structure refers to possible variations in the general structure of the website (i.e., the arrangement of the web pages, the types of web pages, and the content to be included in each type of web page). For instance, in a product's catalog website, a way to show the information of the products in the catalog, is to create for each product a webpage showing a product's particular information. On the other hand, a second option is to display all the products in a single webpage, using a list, where each list item contains a product's information.

Variability on website presentation refers to possible variations in the visual design, mainly by selecting presentation templates that configure the icons, colors and font types of the website.

Variability on WCMS platform refers to possible WCMS platforms on which MONAD should deploy the websites.

3.1 Domain Engineering

3.1.1 Building of Website Assets

In MONAD, we introduce a new type of asset called *Domain Template* that allows web designers and domain experts to define the website's structure, specify the rules of how to generate the content and define where to include it in the website structure.

Domain Template. is composed by the following set of models: (1) a domain-specific meta-model describing the vocabulary and elements of the business domain, (2) a website model specifying the website structure and the different page elements designed for this business domain, and (3) a weaving model indicating in which parts of the website model must be included which elements of the domain.

Business Domain Meta-model. defines the vocabulary and the elements in the domain. For instance, in the manufacturing domain, a business domain meta-model must be defined considering elements such as catalog, category and product.

During application engineering, for each type of website, a domain model must be created based on this meta-model. Figure 2 shows an example meta-model for product catalogs for manufacturing companies.

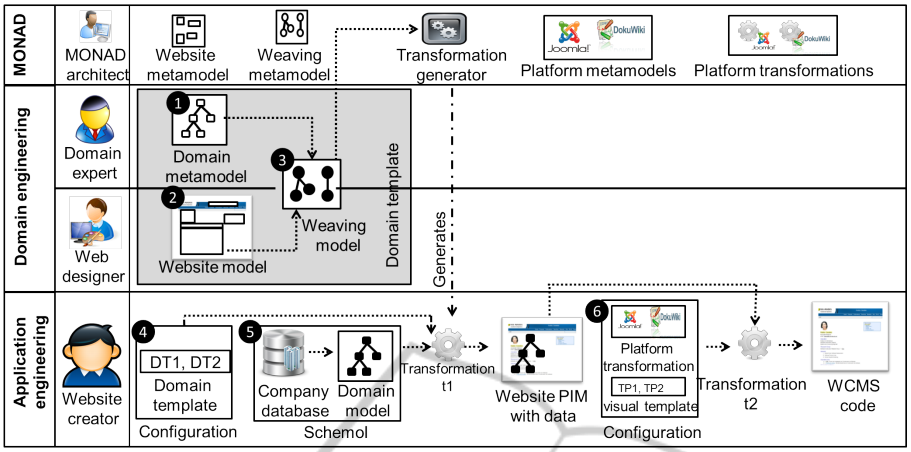

Figure 1: MONAD MD-SPL.

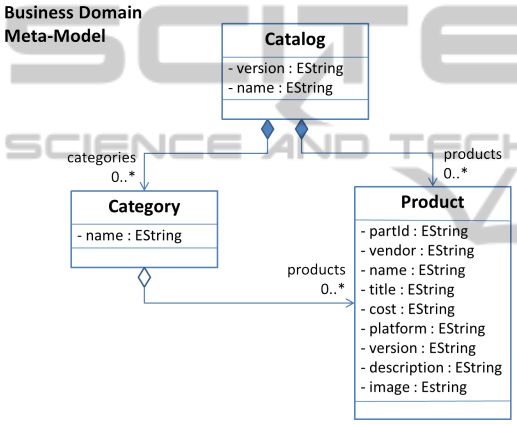

Figure 2: Example meta-model for the domain of product catalogs.

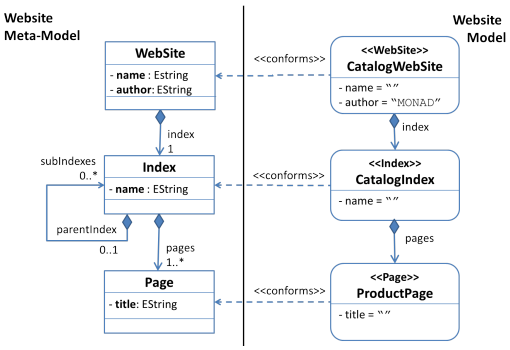

Figure 3: Website metamodel excerpt.

Website Model. describes the website sitemap and the structure of each webpage. The MONAD framework defines a meta-model to create these models, figure 3 presents an excerpt.

Weaving Model. During domain engineering, a website designer and a domain expert create a weaving model (Didonet Del Fabro and Valduriez, 2009) to relate business elements to the website structure. Described as element 3 in figure 1, this weaving model relates a business domain meta-model to a website model.

Figure 4 shows an overview of a domain template for a product's catalog including all the three models. In the figure, the dotted arrows represent references in the weaving model that indicate which elements in the website model must be created from the elements of a business domain model. These references also include information about content that must be assigned to some attributes in the website elements. For instance, for each Product in the business domain model, a web page must be created including the name of the product as the title.

In addition, other considerations must be taken into account when collections are involved in the weaving. Figure 5 presents an example. The original website model, *website model* inside *domain template* in figure 5, is defined with only one product page, however, when a *mapping rule* is defined, the *source element* can be a collection, in this case the products of the catalog, view *Mapping Rule3*.

Therefore, in the final website model, *sample final website model*, there should be a product webpage, *productPage1* and *productPage2*, for each product in the domain model, *product1* and *product2*. A *calculated value* is used to indicate in the weaving model, that the source element is a collection. A calculated value is an OCL query expression. In the case of *MappingRule3* the calculated value is the OCL expression *catalog.products*.

MONAD's *transformation generator* uses the resulting domain template to derive a M2M transformation. This transformation takes a concrete business model and produces an independent platform website

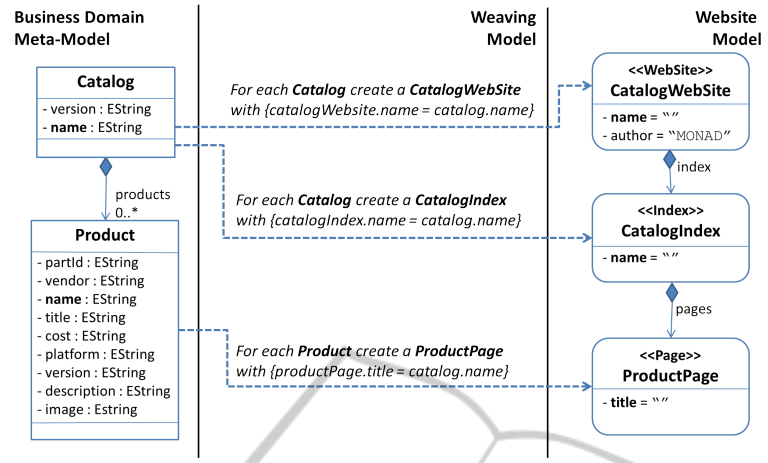

Figure 4: Domain template example.

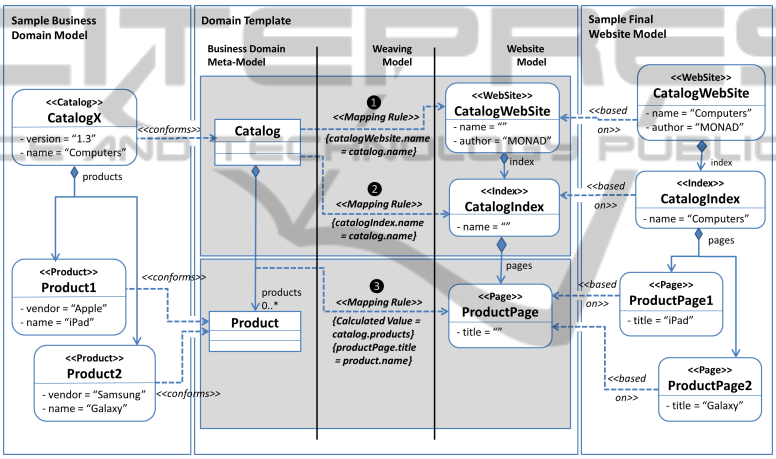

Figure 5: Final website model.

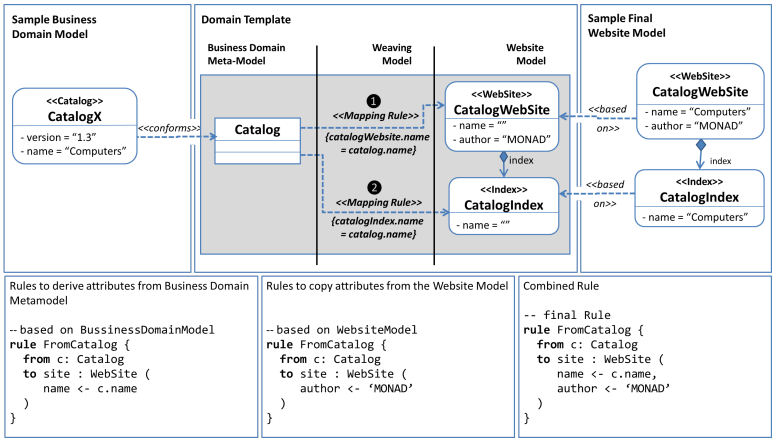

Figure 6: ATL rule generated by the Transformation engine.

model with the content from the business model and the structure from the website model.

Figure 6 shows how MONAD's transformation engine uses *Mapping rule 1* to create a transformation rule in the Atlas Transformation language (ATL). (Jouault et al., 2008) First, it creates an assignment from the domain model *c.name to name*, second it creates an assignment from the original website model

'MONAD' to author and third it merges them into the final rule. *Mapping rule 2* is processed in the same way.

3.2 Application Engineering

3.2.1 Website Configuration

In MONAD, *Step 4* in Application Engineering is the configuration of the structure of the desired website. In this step, website creators must create a configuration model with the expected structure for the website.

Currently, MONAD allows customized variations in the website structure. Instead of using a predefined domain template, website creators can modify a domain template to vary some aspects of the generated website. In these cases, a new domain template must be specified instead of selecting one from the already defined feature model. This configuration and the domain model are taken as input by transformation t1 of figure 1.

3.2.2 Website Derivation

After the previous configuration, to derive a website, two steps are followed. *Step 5* is performed by the website creator who has to provide a business domain model.

The next step, *Step 6* requires the website creator to configure the desired website, this includes selecting a WCMS (e.g. Joomla! or Wiki systems) and a visualization template.

transformation t2 is a M2T transformation that uses (1) the configuration with the target WCMS and appropriate visual template, and (2) the website PIM with the corresponding information, generated by *transformation t1*, and creates the websites in a particular WCMS, using the specific platform API and databases.

Specification of a Business Domain Model. To generate a specific website, website creators must provide a business domain model with the data of the corresponding company. This model has to be conform to the business domain meta-model defined in the domain engineering. Using Schemol (Díaz et al., 2011), we can derive automatically this business domain model from the databases and tables where the information is stored.

Domain Template Transformations. Using the domain template, a High Order Transformation (HOT) (Tisi et al., 2009) is used to create the M2M transformation of *t1* of figure 1. This transformation takes a business domain model and a domain template, and transforms them into a website PIM with the corresponding information. This HOT is independent of the domain and website metamodels, therefore the same mechanism can be used for any business domain.

WCMS-specific Transformations. In contrast with the Domain Template Transformations, there are a several M2T transformations specific to a particular WCMS (described as *t2* in figure 1). MONAD uses these transformations to deploy the website on the WCMS platform selected by the website creator. For instance, to create a website in Joomla, an M2T transformation is used to generate a script of SQL commands that uses the proprietary APIs and databases to store the website content. Then, an additional tool takes this script and execute them into the selected Joomla! server to create the website. Figure 7 presents an example of a product webpage on Joomla!. In this example, from a website PSM, a product webpage is generated including the product picture and an unorder list with some information: part Id, vendor, title, cost, platform and description.

Figure 7: Generated product webpage in Joomla!

4 CONCLUSIONS

This paper presented MONAD, a strategy and a set of tools to apply Model-Driven Product Line Engineering to generate websites for companies in a similar business domain. These websites can be generated considering variability on structure, presentation and WCMS platform. We have presented two case studies: (1) one that generates websites for product catalogs and (2) other that generates websites for faculty members. The second one has been implemented using real data and models from a University and it has been used to continuously improve the approach. In this case study, there is an initial cost associated to the *domain engineering* due to the effort of creating the weaving model and the domain metamodel. However, the cost of creating the 134 websites is minimum

as the domain model can be populated automatically using Schemol, and second, the subsequent creation of the specific WCMS code is automated by means of MONAD's model transformations.

Based on these experiences, we consider that using MONAD instead of manual coding result in economic benefits in scenarios where (1) several websites are created for different companies with a similar website structure and presentation, (2) when the number of pages of the same kind in the website is considerable, (3) when the business domain can be automatically generated using mechanisms such as Schemol.

Further work in MONAD includes the addition of CMS-related concerns such as user profiles and permissions, platform provided functionality such as forums, page commentaries and multimedia content.

REFERENCES

Díaz, O., Puente, G., Cánovas Izquierdo, J., and García Molina, J. (2011). Harvesting models from web 2.0 databases. *Software and Systems Modeling*, pages 1–20.

Diaz, O. and Martin, F. (2009). Generating corporate blogs from product catalogues: A model-driven approach. In *Fifth International Workshop on Model-Driven Web Engineering (MDWE 2009)*, pages 61–75. http://ceur-ws.org/Vol-455/paper05.pdf.

Didonet Del Fabro, M. and Valduriez, P. (2009). Towards the efficient development of model transformations using model weaving and matching transformations. *Software and Systems Modeling*, 8(3):305–324.

Jouault, F., Allilaire, F., Bézivin, J., and Kurtev, I. (2008). Atl: A model transformation tool. *Science of Computer Programming*, 72(1 - 2):31 – 39. Special Issue on Second issue of experimental software and toolkits (EST).

Martinez, J., Lopez, C., Ulacia, E., and del Hierro, M. (2009). Towards a model-driven product line for web systems. In *Fifth International Workshop on Model-Driven Web Engineering (MDWE 2009)*, pages 1–15. http://ceur-ws.org/Vol-455/paper01.pdf.

Moreno, N., Romero, J. R., and Vallecillo, A. (2008). *Web Engineering: Modelling and Implementing Web Applications*, chapter An Overview of Model-Driven Web Engineering and the MDA, pages 353–382. Springer. http://www.lcc.uma.es/∼av/Publicaciones/07/OverviewMDA4WE07.pdf.

Souer, J., Joor, D.-J., Helms, R., and Brinkkempe, S. (2011). Identifying commonalities in web content management system engineering. *International Journal of Web Information Systems*, 7(3):292–308. 10.1108/17440081111165901.

Tisi, M., Jouault, F., Fraternali, P., Ceri, S., and Bézivin, J. (2009). On the use of higher-order model transformations. In Paige, R., Hartman, A., and Rensink, A., editors, *Model Driven Architecture - Foundations and Applications*, volume 5562 of *Lecture Notes in Computer Science*, pages 18–33. Springer Berlin / Heidelberg.