Model-driven Web Engineering in the CMS domain: a preliminary research applying SME

Kevin Vlaanderen[1], Francisco Valverde[2], and Oscar Pastor[2]

[1] Universiteit Utrecht, Padualaan 14 3584CH Utrecht, the Netherlands,
k.vlaanderen@students.uu.nl
[2] Centro de Investigación en Métodos de Producción de Software, Universidad Politécnica de Valencia, Spain,
fvalverde@pros.upv.es,
opastor@pros.upv.es

Abstract. In recent years, the use of Content Management Systems (CMS) as the core tool to define a Web Application has gained popularity. However, the Model-Driven Web Engineering methods are not well fitted into the CMS domain. The main reason is that these methods are mainly focusing on the data and navigation aspects. To address this problem we propose in this chapter the use of Situational Method Engineering in order to detect the potential issues and improvements of a Web Engineering method in the CMS domain. Specifically, the suitability of the OOWS method in the context of CMS-based web applications is evaluated by means of a user-registration use case. From the results of this evaluation, a list of current limitations of the OOWS Method in the CMS domain are detected. Additionally the improvements that can be applied from a SME perspective are introduced.

Key words: Web Engineering, Method Engineering, Model-Driven Development

1 Introduction

Over the last years, the number of Web Applications developed by industry has increased dramatically. However, the quality of those applications is often poor and they are difficult to maintain because of the lack of precise techniques to guide the development. For that reason the principles proposed by the Web Engineering (Deshpande et al., 2002) community have been widely applied. Specifically these principles have been introduced from a Model-Driven Engineering perspective. The main result has been a wide array of Model-driven Web Development methods (see section 2) in order to improve Web Applications quality.

The development of specialised methods has certainly improved the efficiency of Web development processes. However, it is difficult to choose a Web Engineering method that is the most suitable in all the different domains. Mainly because there is no silver bullet method that can address the heterogeneous domains which are facing the Web development. With the arrival of the "Web 2.0", this issue has become more obvious since current Web Engineering Methods are

mainly designed to develop data-intensive Web Applications. As a consequence the methods must be revisited to take into account the social perspective of this new application paradigm. In other words, when a new Web Application domain needs to be supported the method must be adapted, extended or redefined.

Many researches agrees that it is not a reasonable approach to develop a new Web Engineering method from scratch for any possible domain. In this context, applying the Method Engineering principles provides interesting advantages. Method engineering is "the discipline to design, construct and adapt methods, techniques and tools for the development of information systems" (Brinkkemper, 1996). Situational Method Engineering (SME) is a type of Method Engineering that focusses on the method adaptation to a particular situation, as is described in (Saeki, 2003) and (Ralyté et al., 2003). This adaptation can be summarised in four generic steps (van de Weerd et al., 2006): (1) Identify concrete method needs, (2) Select candidate methods that meet some of the identified needs, (3) Analyse the methods and store the relevant method fragments in a method base and (4) Assemble a new method from useful method fragments. Applying SME to Web Engineering methods has two main advantages:

1. Building a common method-base allows the analyst to define a method for a concrete domain using previous and validated method-fragments.
2. A method can be easily adapted taking into account the needs of a concrete project.

In this chapter we present how the SME principles can be applied to a Web Engineering method in order to improve its suitability for a particular domain. The domain chosen is the Web Content Management system (CMS): a system used to manage the content (texts, images, resources, electronic documents) of a Web site. The main difference with traditional Web Applications is that the content is created or added dynamically by the Web users.

The Web Engineering method selected in this work to be improved is OOWS (Fons et al., 2003). This method extends OO-Method (Pastor & Molina, 2007), an automatic code generation method that produces the equivalent software from a system conceptual specification. With the possibility of creating code directly from a conceptual model, it enables drastic improvements in the time and resources required for the creation of Web Applications. However, the OOWS method lacks expressiveness to define Web Applications in some domains as CMS. To be able to apply OOWS in more fields, of which CMS-based Web application development is one, a better understanding and a more detailed specification of the method is needed.

This chapter extends the work previously presented by the authors in (Vlaanderen et al., 2008). In that work SME was applied to define the OOWS method metamodel and to detect the different method fragments. In addition, an analysis was carried on by means of a case study to evaluate the suitability of OOWS in the context of CMS-based Web applications. In this chapter, we perform a new evaluation applying a more complex case study based on a CMS application registration process. This new evaluation has led to extend the previously

defined OOWS method metamodel. Both the new method meta-model and the evaluation, provide a preliminary research about how a common method-base for the CMS domain can be defined.

The rest of the chapter is organised as follows: section 2 presents the background about some related work on Web Engineering methods and introduces the OOWS Method. Section 3, describes the improved OOWS method metamodel that was defined in previous works. Section 4 describes the new use-case defined to analyse the OOWS Method and the improvements needed. Finally section 5 presents the conclusions of the preliminary research.

2 Background

In the last years several Web Engineering methods have proposed a process to develop Web Applications. Specifically, they have encouraged a Model-driven Engineering point of view to improve the development. The generic method proposed can be summarised in five main steps: 1) Requirements gathering, using use cases or a textual specification 2) Definition of the domain model, which gathers the entities of the Application by means of a UML-like class diagram 3) Definition of the Navigational Model, which represents the navigation between the application nodes, 4) Specification of the presentation and design aspects and 5) The final implementation phase. Taking into account little differences and different conceptual models, this five step process is followed by approaches such as OOHDM (Schwabe & Rossi, 1995), WebML (Ceri et al., 2000), WSDM (De Troyer & Leune, 1998) or UWE (Koch & Kraus, 2002). Therefore it is possible to say that there is a common agreement about the steps a Web Engineering method must have.

However, in practice, a rigid or generic method doesn't fit well for every Web Application domain. For instance, a very exhaustive Requirements gathering phase may be not necessary in small-size Web Applications. In addition as (Preciado et al., 2005) has stated, the current methodologies do not have enough expressiveness to model new domains such as Rich Internet Applications. The first solution to these issues has been to redefine the methods. For instance the WebML method (Bozzon et al., 2006) has introduced a set of new conceptual models. However, this is not a long-term solution because the Web is evolving continuously, therefore, Web Engineering methods must evolve simultaneously. This fact leads to defining Web Engineering methods more flexibly applying the SME principles. An example of a CMS-based Web-Application design method developed using SME is the GX WebEngineering Method (WEM) (van de Weerd, 2005; van de Weerd et al., 2006). WEM is currently used at GX creative online development, a web technology company in the Netherlands, to implement web-applications using their own content management system, called GX WebManager. (van de Weerd, 2005) describes how this method has been created and improved through the use of already existing (proprietary) methods such as (Koch & Kraus, 2002). In this chapter the approach defined in WEM is

applied to OOWS. The main purpose is to define the OOWS method using the SME principles and to find method fragments which can be improved.

2.1 The OOWS Web Engineering Method

OO-Method (Pastor & Molina, 2007) is an Object Oriented software production Method to automatically generate information systems. OO-Method models the system in different abstraction levels, distinguishing between the problem space (the most abstract level) and the solution space (the lowest abstract level). The system is represented by a set of Conceptual Models that represents the static structure of the system (Class Diagram) and the behaviour (State and Functional Diagrams). OOWS (Fons et al., 2003) is the OO-Method extension used to model and to generate Web Applications. The OOWS Web Engineering Method adds three models that describe the different concerns of a Web Application:

- *User Model*: A User Diagram allows us to specify the types of users that can interact with the Web system. The types of users are organised in a hierarchical way by means of inheritance relationships.
- *Navigational Model*: This model defines the system navigational structure. It describes the navigation allowed for each type of user by means of a Navigational Map. This map is depicted by means of a directed graph whose nodes represent Navigational Contexts and their arcs represent navigational links that define the valid navigational paths over the system. Basically, a Navigational Context represents a Web page of the Web Application at the conceptual level. Navigational Contexts are made up of a set of Abstract Information Units (AIU), which represent the requirement of retrieving a chunk of related information. AIUs are views defined over the underlying OO-Method Class Diagram. These views are represented graphically as UML classes that are stereotyped with the "view" keyword and that contain the set of attributes and operations which will be available to the user.
- *Presentation Model*: Using this model, we are able to specify the visual properties of the information to be shown. To achieve this goal, a set of presentation patterns is proposed to be applied over our conceptual primitives. Some properties that can be defined with this kind of patterns are information arrangement (register, tabular, master-detail, etc), order (ascendant/descendent), or pagination cardinality.

These models are complemented by the OO-Method models which represent functional and persistence layers. THe OOWS development process is compliant with Model Driven Architecture (MDA) principles, where a Model Compiler transforms a Platform Independent Model (PIM) into its corresponding Software Product. This MDA transformation process has been implemented by means of a case tool and a model compiler The OOWS Model Compiler generates the code corresponding to the user interaction layer, whereas OLIVANOVA (www.care-t.com), the industrial OO-Method implementation, generates the business logic layer and the persistence layer. Further details can be found in (Valverde et al., 2007).

3 The OOWS method meta-model

The meta-modeling technique used in this paper to obtain a view of the OOWS-method is based on the proposal of (Saeki, 2003). The technique uses a combination of two UML diagrams (see fig.1) as is described in (van de Weerd & Brinkkemper, 2007).

- On the left hand side, an adaptation of the UML activity diagram is used for modeling the process of the method. This diagram consists of method activities, sub-activities and transitions between them.
- On the right hand side an adaptation of the UML class diagram that models the concepts: A set of objects which share the same attributes, operations, relations and semantics, used for capturing the deliverable view of a method.

These two diagrams are integrated in a straightforward way after being built. Some of the activities from the UML Activity diagram are associated to concepts from the UML class diagram through a dotted line. The resulting model is called Process-Deliverable Diagram (PDD) where each process step represents a method fragment.

To validate the meta-model proposed in this work, expert-validation has been applied. The meta-model has been checked and revised by both a PhD-student and a full-PhD working intensively on and with OOWS. The OOWS method meta-model defined in this chapter consists of the two PDDs that represent the Requirements Modeling and Conceptual Modeling phases of the method. The metamodel presented here introduces the new method fragments that were detected in (Vlaanderen et al., 2008). Owing to space constraints the implementation phase is not show as no modifications were introduced. However, further information about this phase and more detailed PDDs can be checked in (Vlaanderen, 2007). The activities that made up both phases are briefly introduced below:

1. *Requirements modeling*: The aim of the requirements modeling phase is to gather the user needs in order to build the Web Application Conceptual Model. The PDD is composed of four activities:
   a) Description of the main purpose of the system as a summarised textual definition of the systems final goal.
   b) Identification of the different tasks the user expects to achieve when interacting with the web application. Each task is described using UML Activity diagrams in terms of input and output activities that the user has to perform.
   c) Description of each user-task from the user-system interaction. For each task a textual description of the input and output data structures/functionality is provided.
   d) Categorisation of the user-tasks into performed by the system or imported from external systems.

   The output of this phase is a Web Application requirements specification. Using model to model transformations, it's possible to obtain a first draft

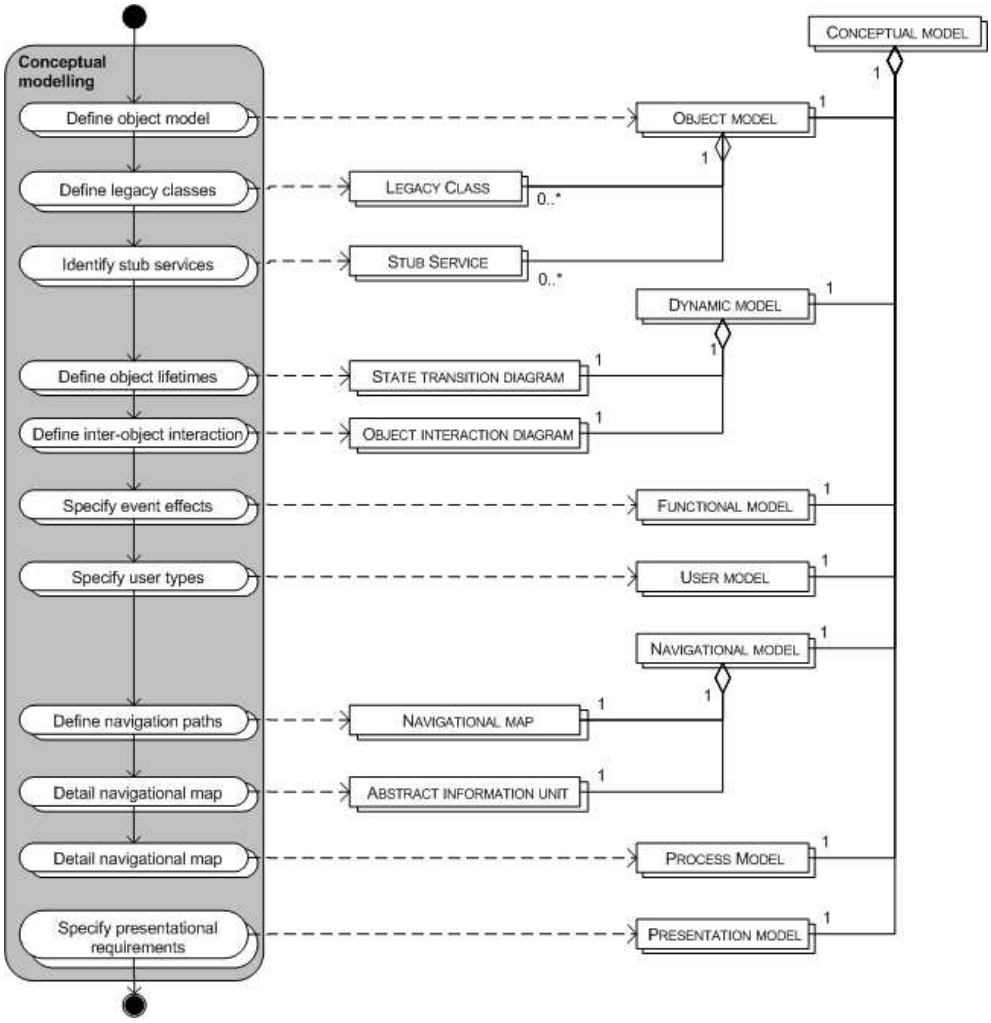

Fig. 1. Process-Deliverable Diagram for OOWS Conceptual Modeling phase

of the Web Conceptual model. Further details about this phase and the activities involved can be found in (Valderas et al., 2005).

2. *Conceptual modeling*: In this phase, the OOWS and OO-Method models that describe the Web system are defined. Since OOWS is an extension of OO-Method, these models are part of OOWS as well. These models and their purpose were explained in section 2.1. Ten activities compose this phase as the PDD in the figure 1 shows. For each activity the conceptual model built is stated:

a) Define the OO-Method Class Diagram (Object Model). This model provides information about the entities, their properties and the services that define the information system.
b) Define the external entities (Legacy classes, external services) that interact with the Object Model
c) Define the possible states of an object and their lifetime using a UML State Transition Diagram (Dynamic Model). This model constrains what kind of services can be executed in the particular state of an object.
d) Detect the interactions between objects and specify the communication between them (Dynamic Model).
e) Specify the object changes after an event occurrence (Functional Model). These changes are specified using a set of logic formulas applied over the object attributes.
f) Specify the different user profiles (preferences, rights etc.) that can access to the Web application (User Model).
g) Define the Navigational Contexts that a particular user can access (Navigational Map).
h) Define the information and services available in a Navigational Context (Navigational Model).
i) Define the complex processes that involves several Navigational Contexts (Process Model).
j) Specify presentation requirements (layout, information order criteria) for a Navigational Context (Presentation Model).

All the activities related to OO-Method models specification (a to e) define the behaviour and data structures (objects) of the system. On the other hand, the OOWS models activities (f to j) define a Web interface. OO-Method models must be defined before the OOWS models since there is a relationship between them (for instance, a user type is linked to a class from the Object Model). This constraint implies that activities from a) to e) must always be carried out before defining the OOWS Models. In addition, for each model to be built, a detailed PDD (Vlaanderen, 2007) describes the sub-activities needed. The output of this phase is an OOWS model that specifies the web system.

The OOWS method metamodel provides a clear representation of the method as a whole. Moreover, a clear description of the activities involved in the Web Application specifications are classified and explained. This metamodel is the starting point to detect method fragments, including those that have to be adapted to the CMS-domain.

4 Analysis of the OOWS method in the CMS domain

The main interest of this work is how the OOWS method can be adapted to define CMS Web Applications. In the previous section, the OOWS method metamodel has been introduced in terms of method fragments or activities. In order to detect which current method fragments should be adapted, a use case from

the CMS-domain has been modeled. The use-case used for evaluation purposes, describes a possible user-registration process, which is an important part of every CMS-based web-application. Figure 2 shows the first part of the sequence diagram for the use-case.

First the user gets to the registration page where he must introduce all the information required. The information to enter are some basic account items such a username, the email address, a name, etc. In addition a cell phone number is required because the user does not have to provide a password; this will be automatically created by the system and send to the user through a text-message. After introducing all the data, a 'please wait'-page will be shown while the data is being checked. At this point, the created password will be sent to the user and an automatically created confirmation-code is sent to the user's email-address. The user-account will not be active until the user enters the received password on the website, and either clicks on the link in the received email or enters the confirmation code on the website's 'confirm'-page. When this validation process is finished, the user will be redirected to the "referrer-page" from which the registration process was started. Finally, when the users logs to the system, a "session" object will be initialised with the information provided in the registration.

This use-case is heavily based on the previous work (Vlaanderen et al., 2008) presented. However, some more complex requirements have been added and the validation process has been altered. As the sequence diagram shows (see Figure 2), the main reason why this use-case is called 'complex' is because of the number of different systems involved: The UDB (user database), the SMSC (service for sending text messages), the e-mail server and the authorisation system.

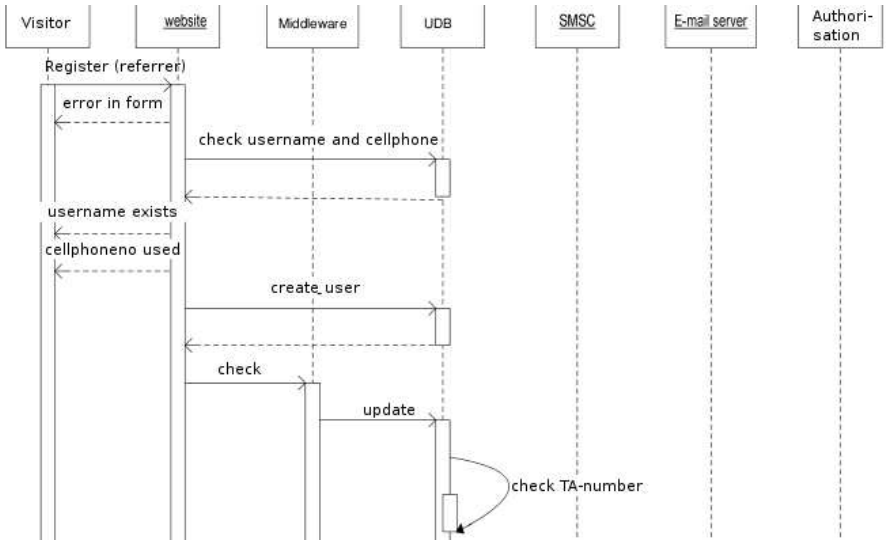

Fig. 2. Partial Sequence Diagram for the complex registration use-case

4.1 User Registration use-case

The use-case presented was modelled using the OOWS conceptual primitives introduced in section (2.1. Firstly, every user-type has a Navigational Context 'Home', on which currently nothing is shown except links to the other available contexts. The AnonymousUser has a 'Register'-context, which allows the entering of all the required information that is needed for executing the register operation. The RegisteredUser has, next to its 'Home'-context, two additional contexts: 'Confirm', which allows the entering of the confirmation code, and 'Phone', which shows all the related Phone entities and the available 'Phone'-services (InsPhone and DelPhone).

The resulting application provides a 'register' service to every 'AnonymousUser' that enters the system. When the user executes that service, he gets prompted for all the required info. Upon completion of the form a new 'RegisteredUser' is created. When the user logs on to this account, he will have the opportunity to 'Confirm' the email address by entering the confirm-code through another service. If the confirm-code is correct, the 'authorised' attribute is set to true, indicating that the user receives full access to his account. Based on this attribute, decisions can be made regarding which services, objects and attributes are visible and/or active.

The authentication mechanism provided by the OOWS-generated Web Applications is a simple approach to user-registration and login. By default a standard login screen is created in which existing users can enter only their login information (id and password) to identify themselves. For this use-case, the user-registration process had to be altered. It is certainly possible to do this at the coding-level, but this is not ideal. A better solution is to do it at the conceptual level. However, this currently requires a complex solution.

4.2 Issues detected and method improvements

The use-case presented in this chapter, was modelled and generated as the OOWS method proposes. Taking into account the lessons learned in the requirements and conceptual modeling phases and the semi-automatically generated implementation, several issues were detected. In order to improve the OOWS method, those issues have been described and a specific solution has been proposed. It is important to note that these issues/solutions are not specific of the CMS domain. Therefore this evaluation has provided a general improvement of the method in other domains. The issues detected can be classified in three main points:

1. *Automatic generation of user information:* When a new user is registered in a CMS, not all the user information is introduced. As the use-case illustrates, a first password must be automatically generated by the system and sent to the user, in order to check if the e-mail address is correct. However, the OOWS User Model doesn't support the definition of this kind of behavior. Each user defined in the model is associated to a class from the Structural

Model which defines their properties and services. Therefore, the default information that is generated when a new user is created is defined in the class construction service. Though, it is possible to associate default values to the user creation service, complex values as a randomly generated password or initialization conditions cannot be introduced. In order to support this requirement, a new step should be defined in the method to optionally extend the definition of the User Model. This new step named, "Dynamic User Information" is required to define the user information which cannot be decided at modeling time. The first user password is an example of such kind of information. However, another interesting example is the automatic language selection taking into account from which network has the user accessed the application. As the step 2.f only includes an static user profile, this optional step to dynamically define the profile must be included.

2. *Dealing with session information from different users:* This issue has a strong relationship with the previous requirement. The user information has not only to be initialised but also, is very common to be accessed while the user is logged into the system. The password or user identification is a very common example, which is not only related to the CMS domain, of this kind of information. For that reason the technological framework that supports the OOWS method, stores by default the user login into a session object. With this session information, the Navigational Map of the user (i.e. which navigational contexts are available) is defined. However, only the login is stored and other information as the password, login-time or new content added is not. In addition, OOWS does not provide any conceptual primitive to access to the session-only information, for example to filter a context according to the type of user logged. Only the information defined in the Structural Model and therefore, that is stored in the database can be used in a particular context. To solve this issue the "Define Object Model" method step must be redefined. After the Object Model has been specified, the analyst should explicitly point out which information will be persisted in the session cache. With the purpose a new optional method step is introduced: "Session Information Specification". In this the step the analyst must use the "Session" stereotype for each class attribute which must be session-persisted. Hence, when the navigational contexts are defined, the session attributes will be available to retrieve information or to establish filter conditions.
3. *Dynamic navigation to the referrer and the 'please-wait' pages:* In the OOWS method the available navigations are defined in the Navigational Model. These navigations are always related to association relationships between two classes from the Object Model, linking information instances related (for example the books written by a specific author). However, as the case study has shown, not all the navigations are triggered to retrieve related information. In particular, in the user registration web page two navigations are automatically performed: 1) a transition to a feedback Web Page to wait until the registration operation is finished and 2) a transition to the Web Page, known as the referrer Page, from which the user started the registra-

tion process. Both navigations are not related to specific relationships from the Object Model, so they cannot be introduced in the Navigational Model. The OOWS method supports the definition of a navigation transition when the service execution is finished, but not while is performed as the "please-wait" page requires. Furthermore, every time the service is finished, the user will be redirected to the same context. In other words, the target of a defined navigation is always the same. To solve this problem the Navigation Model definition must be extended to include dynamic navigations. These new type of navigations are defined in an extension of the traditional "Detail Navigational Map" method step. The main difference is that the dynamic navigations have a condition to define when the transition is performed. Hence different targets can be defined according to which condition is satisfied. For that reason, these navigations are not restricted to navigational contexts which have a structural relationship between their classes.

5 Conclusions and Further Research

Currently, Web applications are focused on more specialised tasks besides the simple information retrieval. It is not hard to imagine that every Web Application domain has its own requirements and therefore introduces its own problems. Creating an all-encompassing method might not be the best approach. A better way to handle this is the creation of domain-specific adaptations of the method. In this paper, Situational Method Engineering has been applied to the OOWS Web Engineering method in order to improve the support for a specialised domain: Content Management Systems. The results of this preliminary research have been: 1) A formal description of the method by means of a meta-model, 2) an analysis in the CMS domain, to detect which method fragments need to be re-adapted and 3) an improved method meta-model to address the improvements detected. The results presented in this chapter have the aim of providing a better view of the OOWS Method and its possibilities and impossibilities. However, the same reasoning line and lessons learned can be applied in another Web Engineering methods. From the experiences introduced in this chapter and in previous works (Vlaanderen et al., 2008), we can summarise the method improvements in three main points:

1. In the CMS domain the user information management is a key requirement. Although, several Web Engineering Methods introduce a User model, the expressiveness is not enough to deal with the modelling of a CMS. Specifically, the OOWS User Model does not provide mechanisms to store all the information required by a CMS system. However, in many Web Applications a taxonomy of users is enough for identification purposes. Applying the SME principles, an additional step to enhance the user modelling can be easily included in the method to meet the domain requirements.
2. Web Applications are evolving to include external services and even to be part of a complex business process. To model these complex requirements

new conceptual models, as Services Models or BPMN-based diagrams, are required to gather the new expressivity. Several approaches as OOWS or WebML have extended their methods to achieve this goal. Nevertheless, that approach implies that the method has a more difficult application. SME can be used to define when these complex models must be defined because a simple Web Application, does not required a complex Business Process.

3. Web Engineering methods have been focused on the definition of Navigational Models to deal with the Hypermedia requirements. These models links the data relationships but they not define dynamic navigations according to some conditions or links to web pages not intended to retrieve data. Since the current Navigational Models have been widely use to define Web Applications, SME could add the definition of complex navigations without impacting on the method.

In addition to the methodological changes explained here, another technological issues regarding CMS web-systems were detected. Examples are the management of multimedia content, aesthetic presentation or error prevention. However, these issues must be solved from an implementation point of view instead of methodological. It is expected that future versions of the tools and the frameworks will improve the code generation.

Finally, the goal of this preliminary research must be to encourage the use of SME among the different Web Engineering methods. Hence, a common method-base can be built instead of a generic Web Engineering Method to deal with any domain or a set of new specific-domain methods. As a consequence, if new methodological requirements need to be introduced, the current method will be extended using a fragments from the method base. Future works will define this common method base applying the same process that we have introduced in this chapter.

6 Acknowledgements

This work has been developed with the support of MEC under the project SESAMO TIN2007-62894.

References

Bozzon, A., Comai, S., Fraternali, P., & Carughi, G. T. (2006). Conceptual modeling and code generation for rich internet applications. In D. Wolber, N. Calder, C. H. Brooks, & A. Ginige (Eds.), *Icwe* (p. 353-360). ACM.

Brinkkemper, S. (1996, April). Method engineering: engineering of information methods and tools. *Information and Software Technology*, *38*(4), 275–280.

Ceri, S., Fraternali, P., & Bongio, A. (2000, June). Web modeling language (webml): a modeling language for designing web sites. *Computer Networks*, *33*(1-6), 137–157.

Deshpande, Y., Murugesan, S., Ginige, A., Hansen, S., Schwabe, D., Gaedke, M., & White, B. (2002). Web engineering. *Journal of Web Engineering*, *1*(1), 3–17.

De Troyer, O. M. F., & Leune, C. J. (1998, April). Wsdm: A user-centered design method for web sites. *Computer Networks and ISDN Systems*, *30*(1-7), 85–94.

Fons, J., Pelechano, V., Albert, M., & Pastor, O. (2003). Development of web applications from web enhanced conceptual schemas. In I.-Y. Song, S. W. Liddle, T. W. Ling, & P. Scheuermann (Eds.), *Er* (Vol. 2813, p. 232-245). Springer.

Koch, N., & Kraus, A. (2002, June10). The expressive power of uml-based web engineering. In D. Schwabe, O. Pastor, G. Rossi, & L. Olsina (Eds.), *2nd international workshop on web-oriented software technology* (Vol. 2548). Malaga: Springer-Verlag.

Pastor, O., & Molina, J. C. (2007). *Model-driven architecture in practice: A software production environment based on conceptual modeling*. Berlin: Springer-Verlag.

Preciado, J. C., Trigueros, M. L., Sanchez, F., & Comai, S. (2005). Necessity of methodologies to model rich internet applications. In *Wse* (p. 7-13).

Ralyté, J., Deneckère, R., & Rolland, C. (2003). Towards a generic model for situational method engineering. In *Advance information systems engineering* (p. 1029). Berlin: Springer-Verlag.

Saeki, M. (2003, June16–18). Embedding metrics into information systems development methods: An application of method engineering technique. In *15th international conference on advanced information systems engineering* (Vol. 2681, pp. 374–389). Klagenfurt: Springer-Verlag.

Schwabe, D., & Rossi, G. (1995, August). The object oriented hypermedia design model. *Communications of the ACM*, *38*(8), 45–46.

Valderas, P., Fons, J., & Pelechano, V. (2005). Transforming web requirements into navigational models: An mda based approach. In L. M. L. Delcambre, C. Kop, H. C. Mayr, J. Mylopoulos, & O. Pastor (Eds.), *Er* (Vol. 3716, p. 320-336). Springer.

Valverde, F., Valderas, P., Fons, J., & Pastor, O. (2007). *A mda-based environment for web applications development: From conceptual models to code* [Paper]. Valencia.

van de Weerd, I. (2005). *Wem: A design method for cmsbased web implementations*. Unpublished master's thesis, Utrecht University, Utrecht.

van de Weerd, I., & Brinkkemper, S. (2007). *Meta-modeling for situational analysis and design methods* [Paper]. Utrecht.

van de Weerd, I., Brinkkemper, S., Souer, J., & Versendaal, J. (2006, July). A situational implementation method for web-based content management system-applications: Method engineering and validation in practice. *Software Process Improvement and Practice*(11), 521–538.

Vlaanderen, K. (2007). *Oows in a cms-based environment: a preliminary research (pending publish)*. Unpublished master's thesis, University of Utrecht.

Vlaanderen, K., Valverde, F., & Pastor, O. (2008). Improvement of a web engineering method applying situational method engineering. In J. Cordeiro & J. Filipe (Eds.), *Iceis (3-1)* (p. 147-154).