IMPROVEMENT OF A WEB ENGINEERING METHOD APPLYING SITUATIONAL METHOD ENGINEERING

Kevin Vlaanderen
Universiteit Utrecht, Padualaan 14 3584CH Utrecht, The Netherlands

Francisco Valverde, Oscar Pastor
Department of Information Systems and Computation, Technical University of Valencia, Spain

Keywords: Web Engineering, Method Engineering, Model-driven development.

Abstract: In recent years, the Web Engineering community has introduced several model-driven methods in order to simplify Web Application development. However, these methods are too general and mainly focus on data intensive Web Applications. A solution to this problem is the Situational Method Engineering. This approach allows the creation or improvement of a web engineering method by reusing method fragments from previous methods. This way, a method suitable for a concrete project or domain can be designed. In this work, the OOWS method metamodel is defined with the purpose of applying Situational Method Engineering. Because of this metamodel, OOWS method fragments can be formalised and used to improve the efficiency of another Web Engineering Methods. Furthermore, the suitability of the OOWS method in the context of CMS-based web applications is evaluated through a user-registration case study. The results of this evaluation, is a list of current limitations of the OOWS Method in the CMS Web Systems domain and possible solutions.

1 INTRODUCTION

As a reaction to the early, chaotic way in which Web Applications were being created in the first years after the introduction of the Internet, (Deshpande et al., 2002) introduced the term Web Engineering. According to them, Web Engineering is the application of systematic, disciplined and quantifiable approaches to development, operation and maintenance of Web-based applications. Over the last years, the term 'Web Application' has become rather common inside the Web Engineering domain, referring to the new family of software applications especially designed to be executed on the web (Pastor et al., 2003). However, the quality of those applications is often poor and they are difficult to maintain. In more recent years, several methods for the design and/or implementation of Web Applications have been developed (see section 2).

The development of specialised methods has certainly improved the efficiency of Web development processes. However, it is difficult to define a Web Engineering Method that is the most suitable in all the domains. The main drawback of these approaches is that a generic method is used for any Web Application development. Furthermore, current Web Engineering Methods are mainly designed to develop data-intensive Web Applications. Therefore, when a new Web Application domain needs to be supported the method must be adapted, extended or redefined.

To develop a new Web Engineering method from scratch for any possible domain is not a reasonable approach. 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 in 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:

Vlaanderen K., Valverde F. and Pastor O. (2008).
IMPROVEMENT OF A WEB ENGINEERING METHOD APPLYING SITUATIONAL METHOD ENGINEERING.
In *Proceedings of the Tenth International Conference on Enterprise Information Systems - ISAS*, pages 147-154
DOI: 10.5220/0001712101470154

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.

The purpose of this work is to apply SME to a Web Engineering method in order to improve the development of Content-Management Systems. A Content Management system (CMS) is a system used to manage the content (texts, images, resources, electronic documents) of a Web site. The main difference from traditional Web Applications is that the content is created or added dynamically by the users of the Web.
The Web Engineering method selected in this work to be improved is OOWS. This method extends OO-Method, 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 We b 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.
The first step to apply the SME to OOWS is to define a meta-model of the OOWS-method to be able to fill the method-base. By creating a meta-model of the method, the OOWS method fragments can be easily detected. As a consequence these method fragments can be used to develop new and improved methods or to decide which are useful for a concrete domain. Besides creating the method meta-model, it is wise to analyse the current suitability of OOWS in the context of CMS-based Web applications. This way it is possible to make a better decision about which OOWS method fragments need to be replaced or to be improved. To carry out this analysis, a little case study based on a CMS application registration process has been implemented. Both the method meta-model and the evaluation, provides the foundation for further research regarding SME and OOWS.
The rest of the paper is organised as follows: section 2 presents some related work about Web Engineering methods. Section 3 briefly introduces the OOWS Web Engineering method. Section 4, describes how the OOWS method meta-model has been built. Section 5 describes the use-case defined to analyse the OOWS Method and the improvements needed. Finally section 6 presents our conclusions and further research.

2 RELATED WORK

In the last years several Web Engineering methods have proposed a process to develop Web Applications. 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 and Rossi, 1995), WebML (Ceri et al., 2000), WSDM (De Troyer and Leune, 1998) or UWE (Koch and Kraus, 2002). Therefore it is possible to say that there is a common agreement about the steps a method engineering 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. This fact leads to define 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 and Kraus, 2002). In this work 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 improve both methods.

3 THE OOWS WEB ENGINEERING METHOD

OO-Method (Pastor and 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 Concep-

tual 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 (see fig.3). 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).

4 DEFINING 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 and 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 created in this work consists of three PDDs that represent the main phases of the method. Owing to space constraints not all PPDs are shown, but can be checked in (Vlaanderen, 2007). The activities that made up each phase and describe the OOWS method process are briefly introduced below:

1. *Requirements Modelling*: The aim of the requirements modelling phase is to gather the user needs in order to build the Web Application Conceptual Model. The PDD is composed of three 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.

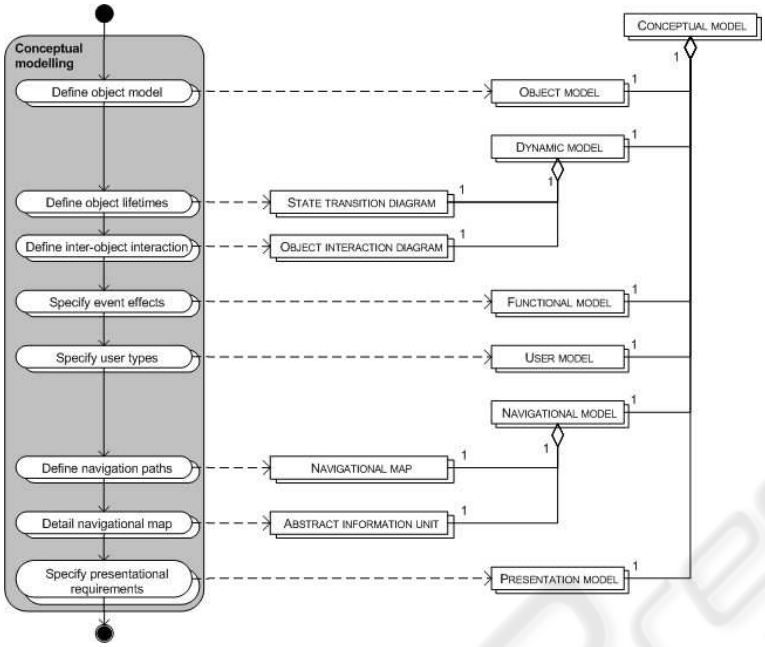

Figure 1: Process-Deliverable Diagram for OOWS Conceptual Modelling phase.

(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.

The output of this phase is a Web Application requirements specification. Using model to model transformations, it's possible to obtain a first draft 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 define 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 3. Eight 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 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.

(c) Detect the interactions between objects and specify the communication between them (Dynamic Model).

(d) 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.

(e) Specify the different user types that can access the Web application (User Model).

(f) Define the Navigational Contexts that a particular user can access (Navigational Map).

(g) Define the information and services available in a Navigational Context (Navigational Model).

(h) Specify presentation requirements (layout, information order criteria) for a Navigational Context (Presentation Model).

All the activities related to OO-Method models specification (a to d) define the behaviour and data structures (objects) of the system. On the other hand, the OOWS models activities (e to f) 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 d) 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.

3. *Implementation*: The main purpose of this phase is generating the software product. Four activities are needed:

(a) Define the application architecture. By default a three-layer architecture Web Application is generated. However, it's possible to obtain only the web interface or a set of web services.
(b) Select the technological target platform (currently PHP, ASP .NET or JSF).
(c) Define the compiler configuration information. This information is related to technological issues such as database configuration, Web server directory and so on.
(d) Send the OOWS Conceptual Model and the configuration options file to the Model Compiler.

This last phase is carried out using both OlivaNOVA and OOWS tools introduced in section 2. The result of this phase is the code of the final Web Application.

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 and which must be adapted into the CMS-domain.

5 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 meta-model 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 activity diagram for the use-case. When the user gets to the registration page, he will have to enter some basic account info like username, email and password. After this, he will be asked for personal information, like name and address. When the information is confirmed by the user, the user will be inserted into the database by the system. A confirmation code is generated and sent to the supplied email address. The user-account will not be active until the user either clicks the link in the email or enters the confirmation code on the website. By analysing the results of this attempt we can also learn how potential method problems can be overcome.

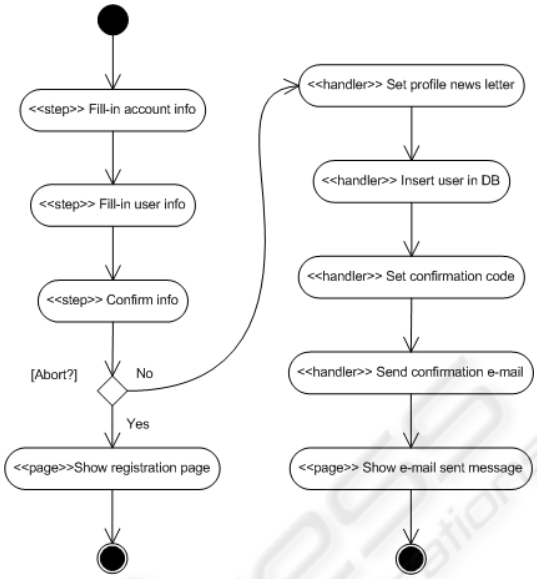

Figure 2: Activity Diagram for the simple registration use-case.

5.1 User Registration Use-case

The Navigational Model for this use-case is shown in figure 3. 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 has 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.

5.2 Issues Detected and Method Improvements

After modeling the use case, the resulting implementation has been analysed. From this analysis some improvements have been detected to define a method more suitable to the CMS domain. These method improvements can be summarised in three critical parts to be solved:

1. *Handling the 'Steps' involved in the Input of the Information.* Usually in Web Applications, a service execution is divided into several steps. For example, when a customer places an order in an e-commerce application, first he fills in his personal data, next he introduces the delivery options available and finally the payment information. In the current method, the definition of this kind of "processes" is not an easy and straightforward job. Services are not classified attending to its complexity. As a consequence the compiler generates the same interface to execute a service: a form with all the arguments. If the number of arguments is high, the usability is compromised. This issue is not only related to service execution. When the user wants to recover a concrete piece of information from a big amount, it's useful to navigate through several steps to filter the information. The user-registration service is an example of this kind of information input. To accomplish this task using OOWS, several services and navigational contexts were used to generate a wizard-like interface. Since a navigational context is represented as a new web page, several linked navigational contexts can represent a wizard to the final user. However, this solution is not compliant with the navigational context semantics. If a user navigates to other contexts it is with the aim of performing a different service. Therefore, the inclusion of a new method fragment to model complex processes is needed. This method fragment will be a new activity in the Conceptual Modelling phase, after defining the Navigational Model (activity 2.g). The purpose of this activity is to mark how data input must be grouped. Currently, work is being done by (Torres and Pelechano, 2006) to support business processes inside the OOWS-Method using BPM diagrams. This work could help to implement activities that consist of multiple steps, like the entering of user-information as shown in this use-case.
2. *User Information Profile.* As stated in section 4.1, the user-registration and login processes provided are generally not suited for CMS web-applications. In this kind of systems, user login or id is not enough. To manage accurately the user rights over the content, user personal information (preferences, rights etc.) must be available along the system. This information is used to adapt which content the user can receive or which navigational contexts can be accessed. User modeling is currently supported in the OOWS Method (activity 2.f). Therefore in this case, no new method activity is needed. Nevertheless, new primitives for modeling individual users are required. In a CMS system, user-profile is a critical requirement in order to decide what content a user can edit, publish etc. As a consequence, the activity 2.f must be redefined to include more expressiveness. For instance, for each user logged in, we need to know what his rights are (see information, execute services, navigation links available) over the AIUs of a Navigational Context.
3. *External Services Integration.* Certain processing, like creating a confirmation code that is used to validate the provided email-address and enable the account, can not be modelled conceptually. With the purpose of providing external functionality, OOWS (using OO-Method conceptual models) provides two mechanisms: (1) The inclusion of 'Legacy Classes' (activity 2.a), that enable the creation of an object-like interface of a external system, and (2) 'User Functions', which are simply stub-services that can be manually defined in the code. However, how and when this external functionality is defined is not clear in the method. In the implementation of this use-case, this problem has been solved by using a combination of specific attributes, services and preconditions to ensure a confirmation code is generated and the user-supplied code is correct, before the user is 'authorised'. This approach is not recommended because it adds a lot of complexity to the conceptual model. Since external functionality needs to be included, this activity must be introduced in the OOWS Method. Firstly, in the Requirements Modelling phase defining which user task are related to external services (activity 1.c). And secondly, when the OO-Method Class Diagram is specified (activity 2.a) distinguishing between the structure of the system and the external services is needed. Again, this method adaptation is not only related to CMS Systems, but is critical in this domain.

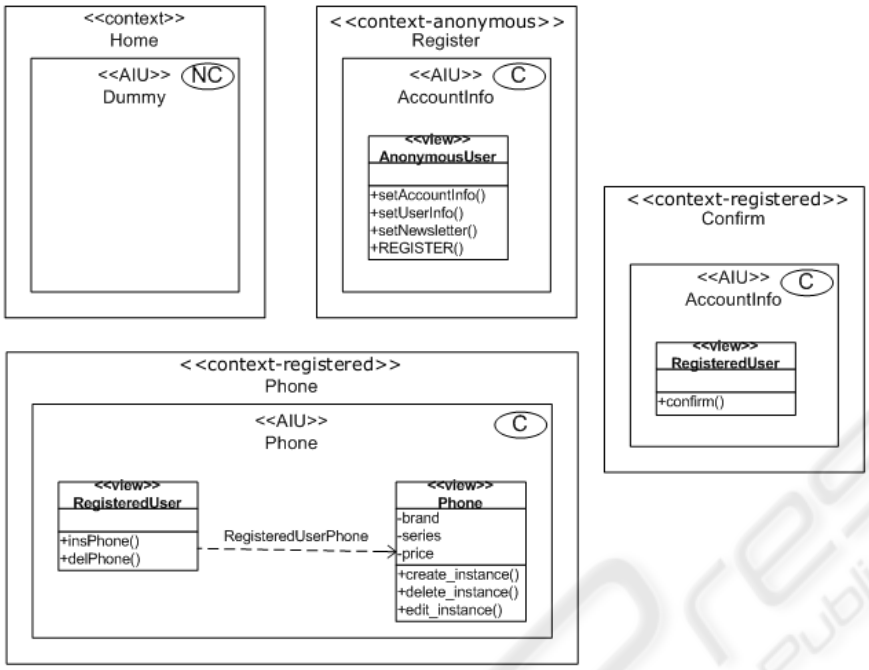

Figure 3: Navigational contexts for the User registration use-case.

6 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 Enginering method in order to improve the support for a specialised domain: Content Management Systems. With the purpose of applying the SME principles to the OOWS Web Enginering Method two artifact have been introduced: 1) A formal description of the method by means of a meta-model and 2) an analysis in the CMS domain, to detect which method fragments need to be re-adapted. On the one hand, the meta-model for the OOWS Method proposed in this work allow to decide on the possible use and re-use of certain parts of the method. On the other hand, the analysis of the method in the light of CMS-based Web Applications has provided a new view on which certain decisions might be made in the future. The results presented in this paper have the aim of providing a better view of the OOWS Method and its possibilities and impossibilities. 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's expected that future versions of the tools and the frameworks will improve the code generation. After detecting the improvements needed, new method fragments should be included inside the OOWS Method. Future works should analyse other Web Engineering Methods, such as WEM, from a SME perspective in order to find this kind of fragments. The final goal of this research work must be to create a new CMS Web Engineering Method from a common knowledge method-base.

ACKNOWLEDGEMENTS

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

REFERENCES

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

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

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

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

Fons, J., Pelechano, V., Albert, M., and Pastor, O. (2003). Development of web applications from web enhanced conceptual schemas. In Song, I.-Y., Liddle, S. W., Ling, T. W., and Scheuermann, P., editors, *ER*, volume 2813 of *Lecture Notes in Computer Science*, pages 232–245. Springer.

Koch, N. and Kraus, A. (2002). The expressive power of uml-based web engineering. In Schwabe, D., Pastor, O., Rossi, G., and Olsina, L., editors, *2nd International Workshop on Web-Oriented Software Technology*, volume 2548 of *Lecture Notes in Computer Science*, Malaga. Springer-Verlag.

Pastor, O., Fons, J., and Pelechano, V. (2003). Oows: A method to develop web applications from web-oriented conceptual models. In *3rd International Workshop on Web Oriented Software Technology*, Oviedo. Luis Olsina, Oscar Pastor, Gustavo Rossi, Daniel Schwabe.

Pastor, O. and Molina, J. C. (2007). *Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling*. Springer-Verlag, Berlin.

Ralyté, J., Deneckère, R., and Rolland, C. (2003). Towards a generic model for situational method engineering. In *Advance Information Systems Engineering*, number 2681 in Lecture Notes in Computer Science, page 1029. Springer-Verlag, Berlin.

Saeki, M. (2003). Embedding metrics into information systems development methods: An application of method engineering technique. In *15th International Conference on Advanced Information Systems Engineering*, volume 2681 of *Lecture Notes in Computer Science*, pages 374–389, Klagenfurt. Springer-Verlag.

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

Torres, V. and Pelechano, V. (2006). Building business process driven web applications. In Dustdar, S., Fiadeiro, J. L., and Sheth, A., editors, *4th International Conference on Business Process Management*, volume 4102 of *Lecture Notes in Computer Science*, pages 322–337, Vienna. Springer-Verlag.

Valderas, P., Fons, J., and Pelechano, V. (2005). Transforming web requirements into navigational models: An mda based approach. In Delcambre, L. M. L., Kop, C., Mayr, H. C., Mylopoulos, J., and Pastor, O., editors, *ER*, volume 3716 of *Lecture Notes in Computer Science*, pages 320–336. Springer.

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

van de Weerd, I. (2005). Wem: A design method for cms-based web implementations. Master's thesis, Utrecht University, Utrecht.

van de Weerd, I. and Brinkkemper, S. (2007). Meta-modeling for situational analysis and design methods.

van de Weerd, I., Brinkkemper, S., Souer, J., and Versendaal, J. (2006). 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). Master's thesis, University of Utrecht.