Model-Driven Web Engineering for the Automated Configuration of Web Content Management Systems

Jurriaan Souer[1], Thijs Kupers[1], Remko Helms[2], and Sjaak Brinkkemper[2]

[1] GX, Wijchenseweg 111, Nijmegen, The Netherlands
{jurriaan.souer,thijs.kupers}@gxwebmanager.com
http://www.gxwebmanager.com

[2] Department of Information and Computing Sciences,
Utrecht University, Utrecht, The Netherlands
{r.w.helms,s.brinkkemper}@cs.uu.nl
http://www.cs.uu.nl

Abstract. With the growing use of Web Content Management Systems for the support of complex online business processes, traditional implementation solutions proofed to be inefficient. Specifically the gap between business requirements and the realized Web application should be closed. This paper presents the development of a modeling tool for the automated configuration of Web Content Management Systems (WCM) which aims to reduce the complexity and increase the transparency of implementations. It allows business users to configure the business processes without technical support. We combine fragments of existing Web Engineering methods and specify an abstract and concrete syntax based on a domain model and end user analysis. The resulting WebForm Diagram has been implemented in a prototype and validated by subject matter experts. This research is part of a project to develop the Web Engineering Method (WEM) which provides an overall method towards a full coverage of the specification, design, realization, implementation and maintenance of WCM-based Web applications.

1 Introduction

The World Wide Web has evolved towards a platform for sophisticated enterprise applications and complex business processes. In result, the effort of time specifying and developing these Web applications reflects the complexity of these applications and business processes. An industry solution to improve development time and stability is a Web Content Management (WCM) system which is product software with out-of-the-box functionalities and allow for specific customizations [26].

Customizing WCM systems to implement business processes is a difficult task. In this context we developed the Web Engineering Method (WEM) as a method to manage and control web applications and web sites based on WCM systems [28]. WEM describes a complete development and implementation process of Web Engineering based on WCM systems as defined by Kappel et al [8]: requirements analyses, design, implementation, testing, operation, and maintenance of high-quality Web applications [25].

The central problem we are addressing in this paper is that there is a gap between the requirements analysis in WCM implementations (usually defined by business users or online marketers) and the realization of those requirements (usually done by technical developers or software engineers). Therefore our leading research question is: ‘how to automatically configure a WCM System based on requirements’. Note that we do not try to create a WCM system, but configure an exsisting WCM for the development of Web Applications. We focus on the configuration of business processes and defined the following goals:(1) Develop an unambiguous form definition to model a business process; (2) Automatically configure a WCM system based on the business process model; and (3) Generate a business process model based on a configured WCM system. This paper presents a new approach for an automated configuration of WCM systems using a modeling tool. The contribution of this research consists of a Model Driven Engineering (MDE) approach for the configuration of WCM software and a unique concrete and abstract syntax to model WCM systems. Secondly, we implement the MDE specification as a prototype and used it to model an actual project situation.

The rest of this paper is organized as follows: In Section 2 we introduce the WebForm Diagram which is the result of this research. Section 3 elaborates on the problem area analysis which resulted in the WebForm Diagram including a domain model, a user analysis and a comparison of existing web engineering methods. In section 4 we formally specify the WebForm Diagram and describe the development of a prototype which is evaluated in section 5. Finally conclusion and future work are discussed in Section 6.

2 The WebForm Diagram

The WebForm Diagram is a visual language for the specification of online business processes. It is developed to cover all online form variables within a web-based application.

As an example of the realized WebForm Diagram we refer to Figure 1 that models a user registration scenario from a real life case: it illustrates how a new user can enter his account information, some personal information and after validation by e-mail and text messaging he is allowed to enter the registered area. However, there are some conditions which the system and user should meet and during the process some system processes and database access are initialized. The purpose of this research is to allow functional users to model such a form and automatically configure a WCMS to support this process. We detail this form in the following sections.

2.1 Steps, Routers, Validation, and Handlers

In the WebForm Diagram, the screens of the form dialogue which are actually presented on a Website are called **steps** and are visualized with rounded rectangles. A single step can have multiple *formfields* such as text input, list item, checkbox, etc. A typical first step of a user registration process has two formfields: desired username and password. The form displayed in Figure 1, has 5 steps starting with ‘Account Info’. Individual formfields are not displayed in this top level view of the WebForm Diagram. However, the diagram allows users to detail steps.

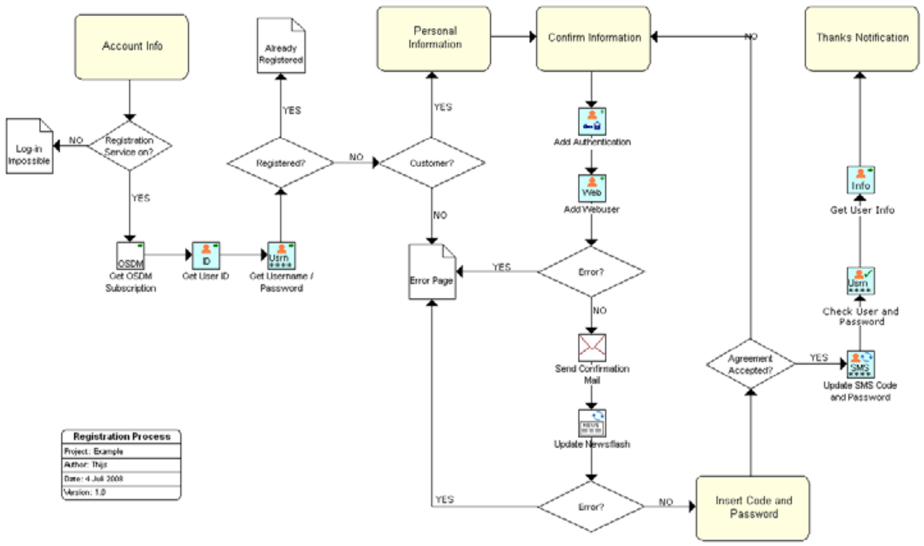


Fig. 1. Example of a business process modeled with the WebForm Diagram

Steps are connected with **routers** which are visualized by arrows between steps. After each step, a form is routed to another step. In the form in Figure 1, after the user enters his username and password in the 'Account info' step the system will try to route him to the step 'Personal Information'.

The flow of the routers depends on certain conditions which are known as **validations**. These validations are visualized with a diamond. For example, the user can only register when the registration service is set to 'on' which is the first validation after step 'Account Info' in Figure 1. Otherwise he will be routed to the web page 'Log-in Impossible'.

Between steps, the WebForm Diagram can perform actions which we call **handlers** and are visualized with a small square containing an icon inside representing the functional behavior of the handler. A handler can for example access a database to check the user id, send a confirmation e-mail or check the user credentials with an authorization server by SOAP.

Hence a complete online business process is a logical flow of steps which are routed in the correct way, with validated input fields and correct handling of the data.

2.2 Formalizing the WebForm Diagram

To develop our WebForm Diagram, we formally specified the syntax to define the logic. The WebForm Diagram relies on the Model Driven Engineering paradigm [10]. MDE is in accordance with the objectives of this research: configuration of a software system based on a model [20]. MDE uses models as a primary artifact in the development process and consists of a Domain Specific Modeling Language (DSML) and a transformation mechanism [29] reflecting the structure, behavior or requirements of a given domain and transforms the model into different implementations respectively.

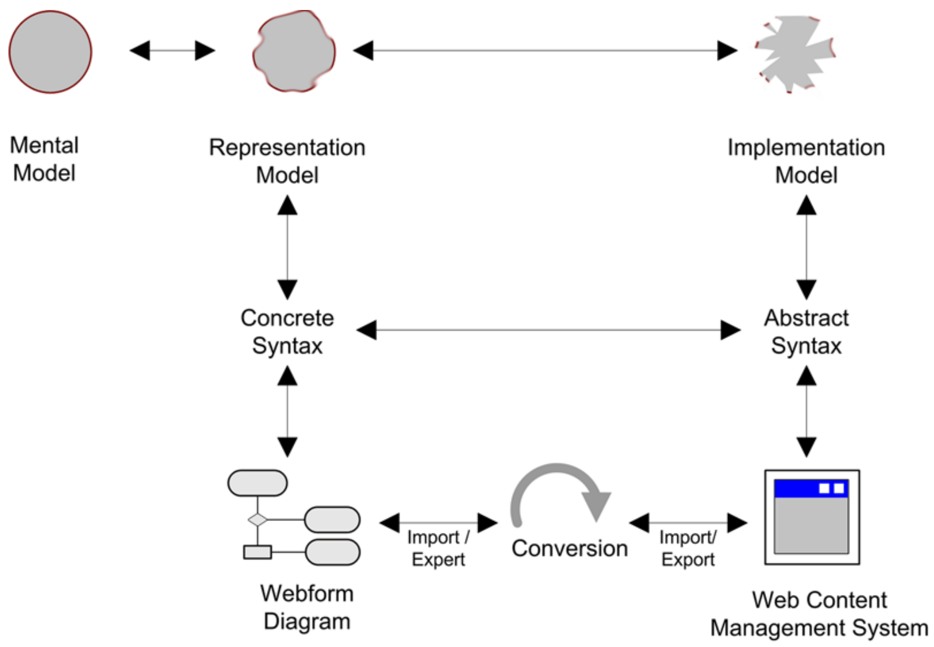


Fig. 2. Research Approach

A DSML is a graphical and executable specification language that is specialized to model concepts within a given domain by means of notations and abstractions. To develop our WebForm Diagram we therefore analyze the problem area with the following three activities: (1) what needs to be modeled, using domain modeling [7];(2) what is the expected outcome, with user analysis as defined by [4] resulting in a mental model, a representation model and an implementation model; and (3) Comparing concepts of existing models to identify key concepts which we could use in our model. Similar to Situational Method-Engineering [2], [17], we selected method fragments from existing modeling languages and assembled our new modeling language out of these fragments.

The DSML in this research consists of a comprehensive abstract syntax and an understandable concrete syntax with a graphical notation [6]. The concrete syntax resembles the representation model of the end user and the abstract syntax represents the objects which need to be configured in the WCM System. We developed the model as a prototype based on the graphical notation of the concrete syntax which we resulted in the WebForm Diagram. The prototype is developed in MetaEdit+ since it allows us to build our own specific development environment. The definition can be exported into an XML format and is converted into the WCMs compliant XML which by itself can be imported into the WCM system. An overview of the approach is illustrated in Figure 2.

The evaluation consists of both a functional assessment in which we used the prototype tool to define a real project situation, and an expert validation in which we interviewed end users.

3 Problem Area Analysis

This research has been carried out within GX, a software vendor of Web Content Management system. GX develops GX WebManager, a Web Content Management System based on open standards such as OSGi as implemented in Apache Felix, Spring MVC and the Java Content Repository by Apache JackRabbit.

3.1 Domain Model

We use a domain model to analyze the elements and their relationships. Domain Modeling helps identify the key concepts which need to be modeled as well as generalizations which relates the entities on a higher abstraction level and is a meta-model of the objects of the modeling language. A relevant functional component within the WCMS – in the context of this research – is the 'Advanced Form Module': a functional component which allows editors to develop business processes based on advanced forms.

Using a domain model, we identified all the elements that need to be modeled by the WebForm Diagram and is the foundation of the abstract syntax.

3.2 User Analysis

In the next step we identified the expected model by interviewing end-users using a threefold model as proposed by Cooper [4]: **Mental model** providing the system, its components and the way they interact from an End-user (or functional) perspective; **Implementation model** reflecting the actual system implementation composing of all

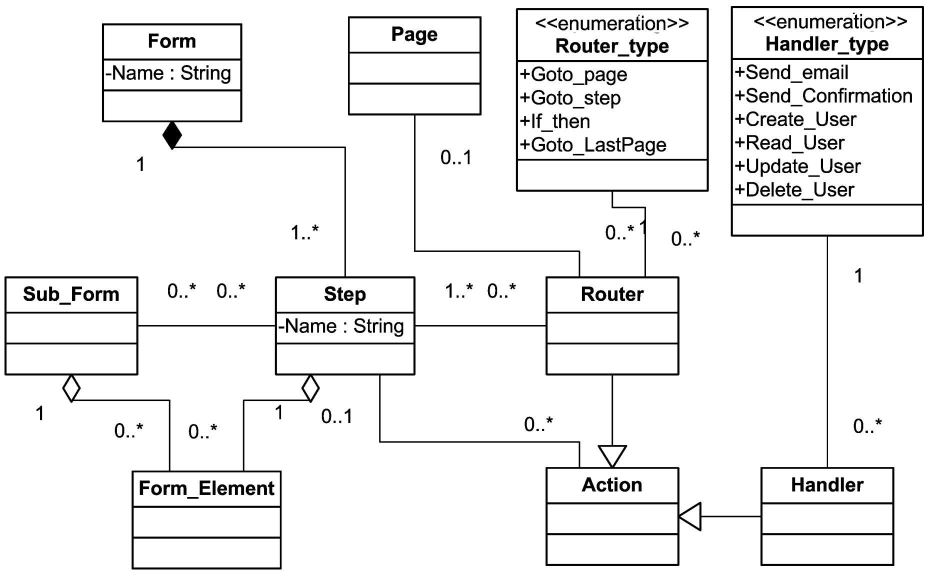


Fig. 3. Excerpt of Domain Model

components, their relationships and the way they interact; and **Representation Model** representing the implementation to the end-users. One should try to create a representation model as close to the mental model as possible [5]. Within a modeling language, the representation model consists of the graphical notation.

We interviewed six users (3 senior technical architects and 3 senior functional consultants). All six users defined a form as a set of *Steps* on the one hand, and a *Flow* defining the order of the steps on the other hand. They visualized it as a set of square blocks (the steps) with arrows in between (representing the order of the steps, or 'flow'). After they defined the steps and the flow on a high level they started to specify the steps in detail. In other words: the users expected at least 2 levels of abstraction: (1) high level defining the steps and the flow between the steps, and (2) a more detailed level specifying the contents of the step. Each step has multiple input fields from a certain type (e.g. text field, password field, radio button, etc), buttons and information. Two architects suggested adopting Business Process Modeling Language (BPMN) as a solution for the modeling language especially since WCM systems are often integrated into other systems and BPMN is a well known standard for defining processes [23] and used within Web Engineering [1]. When a router was conditional, a diamond was suggested with multiple outbound lines. The users did not have any clear ideas how to cope with database connections and handlers except for the idea of using an object to define that there should be a handler on that specific place. With the user analysis we identified the mental model of the end users.

3.3 Comparing Existing Models

In this section we describe a comparison and selection of web application modeling approaches in order to fill a method base. We continue with our previously filled method base which we gathered using Situational Method-Engineering [17] , consisting of the following approaches: Object Oriented Hypermedia Design Model [21], WebML[3], UML-based Web Engineering [11] and Object Oriented Web Solutions [15].

In [14] we compared existing methods with a comparison matrix. In this research additional requirements were gathered which resulted in an adjusted comparison matrix. We compare existing models based on the Cooperative Requirements Engineering With Scenarios Framework (CREWS) [18] as it has been successfully used for classification of software [19], [24]. The adapted CREWS framework classifies modeling methods by four views: Content, Form, Purpose, and Customization process. Each view has a set of associated facets, and each facet is characterized by a set of relevant attributes [24]. We adapted the CREWS framework with similar views, adding a domain view as a separate view. Each view has different aspects describing the properties of the modeling language which we compared. **Content view**: analyzes which existing knowledge is being used within the model; **Form view**: identifies the structure and the graphical notation of the modeling language; and **Domain view**: compares the entities within the domain with the modeling concepts of the modeling language. We detail the Content View. The Content View analyzes to which extend knowledge is being used. It is analyzed on the following three aspects: (1) *Abstraction*, (2) *Context* and (3) *Argumentation*.

Abstraction has one attribute reflecting the possibility of an approach to incorporate different abstraction levels. There are three possible values for this attribute: Flexible (multiple levels possible), Fixed (multiple levels possible, but when a level is chosen it cannot be changed) or Unique (only one level). In our case, the preferred value is 'Flexible' since the user analysis asked for a multiple detailed abstraction.

Context consists of three operators: Internal (model the internal system), Interaction (model the interaction between the system and its environment) and Contextual (model the environment and its context). In our case, the preferred values are 'Internal' and 'Interaction' since the scope of this research is modeling the user interaction (business processes) and the configuration of an internal system.

Argumentation consists of an 'Argument' which defines if the model allows for providing arguments. The preferred value in our case is 'Argument' set to TRUE since it would allow end users to provide arguments for their decisions.

An example of the Content View table is displayed in Figure 1. Based on the Content View we conclude that the Business Process Model (OOWS) and the Activity Diagram (UWE) are the most suitable.

Table 1. Content View Table

Aspect	**Attribute**	**OOHDM**		**WebML**	**UWE**	**OOWS**	***Preferred***
		User Interaction Diagram	**Coceptual Class Schema**	**Business Process Diagram**	**Activity Diagram**	**Business Process Model**	
Abstraction	Abstraction	Unique	Unique	Flexible	Flexible	Flexible	Flexible
Context	Intern	–	TRUE	TRUE	TRUE	TRUE	TRUE
	Interaction	TRUE	–	–	TRUE	TRUE	TRUE
	Contextual	–	–	–	TRUE	–	
Argumentation	Argument	–	TRUE	–	TRUE	TRUE	TRUE

Similar to the CREWS framework we made a Form view and a Domain view. We concluded based on the Form View that – again – both the Activity Diagram (UWE) and the Business Process Model (OOWS) are interesting because of their functional perspective. In the Domain View, we compared the entities from the earlier defined Domain Model with the concepts of the different approaches. Our modeling language will be a construct of multiple modeling languages. Formally defined:

$$MM \subseteq \left(\bigcup_{i=1}^{n} MM_i \right) \cup MM_{new}$$

where MM_1, MM_2,..., MM_n are the meta models of the analyzed modeling languages and MM is the meta model of our new modeling language. MM_{new} reflects the elements which are not covered by the selected meta models but which are necessary within our new modeling language. The relationship between the entities of the Domain Model and the meta model is defined by function *map*.

$$map : MM \rightarrow D$$

Each entity within the Domain Model should be defined in the meta model. We therefore define the following:

$$\forall d \in D(\exists m \in MM[map(m) = d])$$

Based on the Domain View, the Conceptual Class Schema (OOHDM) seems useful, although it lacks the diversity of entities which makes it hard for business users to use it. The Activity Diagram (UWE) and the Business Process Model (OOWS) provide good possibilities to define the flow of a form. However, they do not provide a way to define the different elements within a form. The Business Process Diagram (WebML) is somewhere in between: it provides the possibilities to define the different elements within the form, but is not that sophisticated for defining the form flow.

Taken all three views in account, we conclude that the Activity Diagram (UWE) and the Business Process Model (OOWS) are the closest to our target modeling language: they provide a way to define the interaction between user and the system, have multiple abstraction levels, a functional perspective and have a strong focus on defining the flow of the form. However, they lack the possibility to define the different elements within the form. The User Interaction Diagram (OOHDM) is more suitable in this particular area because it represents the form elements in a compact and well-organized way. Also the UML Class Diagram (being used by most of the Web Engineering methods) can define different form elements. The Business Process Diagram from WebML has other possibilities to model these forms, but has a less compact notation to define the user interaction with the system.

We therefore selected the Business Process Model, the User Interaction Model and the UML Class Diagram as the base models for our modeling language.

4 Specifying the WebForm Diagram

To allow end users to define models which precisely express their expectations of the business process, and automatically translate models into the configuration of a WCM system as defined by its architecture, we need to formally specify our model. Similar to any other modeling language the WebForm Diagram consists of syntax and semantics. The syntax defines the rules, constructs and its decomposition where the semantics defines the meaning and interpretation of the constructs. The syntax and notation should be separated when developing a graphical language [6]. We therefore use a concrete and abstract syntax similar to the author in [4]. We elaborate on the concrete and abstract syntax in the following sections.

4.1 Concrete Syntax

The graphical notation of the concrete syntax is the representation model of the Web-Form Diagram. It should therefore match the mental model of the user which defined a web form in concepts of steps, routers, formfields and dependencies. More formalized we define form f in the following nonuple:

$$f = \{N, S, FE, V, H, C, P, B, E\}$$

Where form f consists of: N is the set of nodes, S the set of steps, FE the set of form elements, V the set of validators, H the set of handlers, C the set of conditions, P the set of web pages, B the set of blocks, and E the set of edges. We elaborate on one element to illustrate how the elements are defined – it goes beyond the scope of this paper to elaborate on each of these elements.

Form Element: A Form element is similar to a Data Entry (User Interaction Diagram) or an Attribute (UML). It is a superclass of elements which are presented to a visitor in a single step of the business process. The set of form elements *FE* can be defined as a union of different formfields where *INPUT* is the set of input types, *BUTTON* the set of buttons and *INFO* the set of information elements. Each step has zero or more form fields in a specific order. The function *fields* provides a set of form elements for a given step:

$$fields : S \rightarrow \acute{S}(FE)$$

where *Ś(FE)* is the set of all possible tuples of form elements. The function *fields* has the following characteristics: each form element is linked to one single step. This can be formalized in the following axiom:

$$\forall f \in FE \, \forall s_1, s_2 \in S \, \forall i, j \in \mathbb{N}$$
$$[occ(f, fields(s_1), i) \wedge occ(f, fields(s_2), j) \Rightarrow s_1 = s_2]$$

The function *occ()* determines if an element *f* exists on position *i* within the given sequence and since form element can only exist *once* within a single step we add the following characteristic:

$$\forall f \in FE \, \forall s \in S \, \forall i, j \in \mathbb{N}$$
$$[occ(f, fields(s), i) \wedge occ(f, fields(s), j) \Rightarrow i = j]$$

Similar to the definition of the formelement, we defined all the other elements of the Form.

4.2 Abstract Syntax

Where the concrete syntax corresponds to the representation model, the abstract syntax is a representation of the implementation model and is a formalization of the Domain Model. There are similarities between the Abstract and the Concrete Syntax, such as *Steps*, *Validation rules*, *Handlers*, *parameters* and *pages*. However, when validation rules and handlers which are connected to a single step are configured within the WCM, they are executed in a consecutive order. Edges are therefore transformed into one of two different concepts: a sequential number or a *router*. The sequential number defines the order of execution from the formvalidations, handlers and routers. An edge which leads to a different *step* or *page* will be transformed into a *router* since it will route to a new step or a page. The actual configuration of the router depends on the conditions of the object. An examples is that a user will only be routed under certain preconditions. Formalized, the WebForm Diagram *f'* is a septupel:

$$f' = \{N', S', FE', V', H', R', P'\}$$

Where form *f'* consists of: *N'* is the set of nodes, *S'* the set of steps, *FE'* the set of form elements, *V'* the set of validators, *H'* the set of handlers, *R'* the set of router, and *P'* the set of edges. Note that Conditions, Blocks and Edges as defined in the concrete syntax are defined within this definition. An Edge for instance is translated into a Router in the abstract syntax. Also in the Abstract Syntax, there are axioms defining the constraints. Example: each router can only be attached to one single step.

$$\forall r \in R' \forall s_1\, s_2 \in S'$$

$$[r \in routersS'(s_1) \wedge r \in routersS'(s_2) \Longrightarrow s_1 = s_2]$$

4.3 WebForm Diagram Modeling Tool

We have defined the concrete and abstract syntax. We then developed a modeling tool to implement the WebForm Diagram as a prototype. To build our prototype of the WebForm Diagram we used MetaEdit+ [9]. This application is both proven in building a CASE tool as well as providing computer aided support for method engineering [27].

The integration of MetaEdit+ and the WCMS is facilitated by XML: the WebForm is exported from MetaEdit+ as an XML file. This XML file however is MetaEdit+ specific which we transform into an XML resembling the Concrete Syntax (GXML). We then translate the Concrete Syntax (GXML) into the WCMS compliant XML (GXML') using a conversion tool. Since it is a prototype, we have written the conversion in general-purpose languages Java and XSLT. The GXML' can be imported into the WCM system. The transformation process works both ways: WCM form definitions can be exported to XML and can be transformed and imported into back into MetaEdit+.

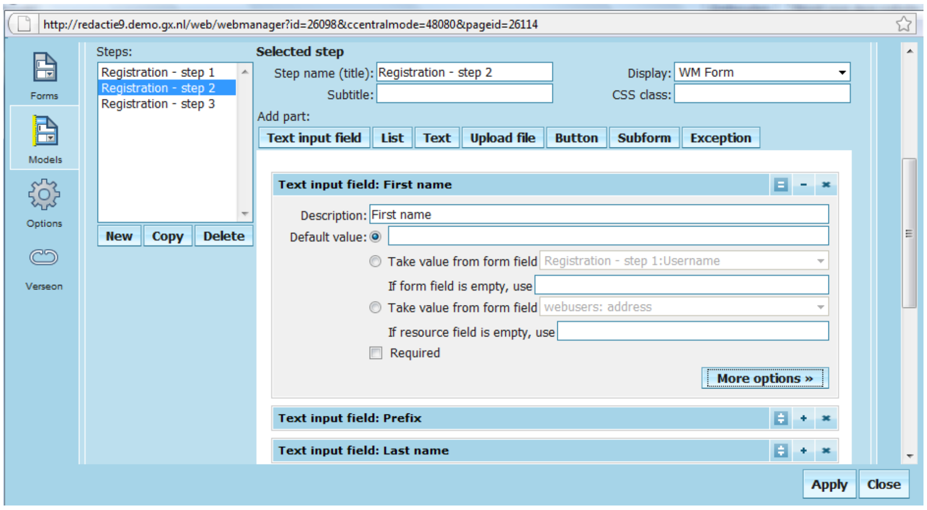
5 Evaluation

We evaluated the WebForm diagram from a functional assessment to see if business users were able to develop a business process in a modeling tool to automatically configure a WCMS, thereby closing the requirements gap and making the implementation more transparent. The reduction in complexity is determined by interviewing experts on the matter.

5.1 Case Validation

To test the application in a real life situation, we took a real case situation and implemented a business process from an existing website using the WebForm Diagram. The complete implementation has been tested both ways: i.c. Designing a WebForm Diagram in MetaEdit+, XML transformation and importing it into the WCMS; and the same process in the opposite order to visualize a defined WebForm in MetaEdit+. The organization of the case is a large Dutch telecommunications provider which provides telephone, internet and television services to individuals and organizations within the Netherlands.

A WebForm Diagram is modeled by a business user, exported into the MetaEdit+ specific XML, transformed the exported XML and then imported into the WCMS. The import went without any problems and the resulting form in the WCMS is displayed in


http://redactie9.demo.gx.nl/web/webmanager?id=26098&ccentralmode=48080&pageid=26114

Steps:
Registration - step 1
Registration - step 2
Registration - step 3

Selected step

Step name (title): Registration - step 2
Subtitle:
Display: WM Form
CSS class:

Add part:

Text input field: First name

Description: First name
Default value: ●
○ Take value from form field Registration - step 1:Username
If form field is empty, use
○ Take value from form field webusers: address
If resource field is empty, use
☐ Required

Text input field: Prefix

Text input field: Last name

Fig. 4. WebForm Diagram converted into a WCMS Form Module

Figure 4. The form had the correct configuration of handlers and routers which could also be tested by registering a new user using the newly configured form.

To gain transparency of a WCMS implementation, we also validated the process in the opposite direction: creating a WebForm Diagram based on an existing form configuration in the WCMS. This process is slightly more difficult since not all aspects of the existing form can be extracted into an XML. The form we use for this validation is a user sign in form allowing users to access a secure page. This form consists of two steps: Step 1 allows the user to enter his credentials (username/password) and submit them to the web application. Although the form is not that complex, it is an often used business process in online transactions. If the credentials are correct the user will be routed to step 2, the secure page. If the credentials are not correct he will be routed back to the sign in page and will get an error message. We exported this form and transformed it back to the MetaEdit+ specific XML and imported it to present the WebForm Diagram without any problems.

Based on the functional validation, some limitations concerning this implementation where identified preventing it from putting it directly into practice. Within the WCM system, there is an extensive library of existing handlers and routers. Still, in projects new handlers and routers are sometimes developed for project specific purposes (e.g. connect to a specific customer legacy system). These routers and handlers are not available by default in the WebForm Diagram. This resulted in a change in the WebForm Diagram: a new placeholder handler during the modeling phase which needs to be configured or developed within the WCM System afterwards. A second limitation we found when we visualized a web form based on a predefined form definition: some routers were conditional based on volatile information (e.g. user specific session information).

This conditional information is not available in the router definition and can it was not taken into consideration when we developed the WebForm Diagram.

5.2 Expert Validation

We validated the complexity of the WebForm Diagram by interviewing eight users: four technical architects and four functional consultants. The aim of this validation is to find out if the users find the WebForm Diagram useful and applicable in implementation projects. The users were shown the WebForm Diagram as illustrated in Figure 1 with additional abstraction layers to detail the form steps. They were provided with a list of statements addressing usability, suitability, transparency (for example to use with internal and external communication) and if they would use it in actual projects. The list with statements had a 4-points scale resulting in respectively the following predicates: Strongly disagree, Disagree, Agree, and Strongly Agree. The results were then gathered and summarized.

The users find the Diagram clear, easy to read, and easy to use. They also think that it would improve the transparency within projects. The architect and the consultants are convinced that it would improve the requirements analysis and design with the end customer. Using this model will improve the validity of the requirements since the customer will probably understand such a diagram easier then the written descriptions. The architect can also use the model to check whether all relevant information is available for the implementation of a business process. However, they are not yet convinced that the handlers and validators will help the design phase much. The general opinion is that it depends much on how it is visualized in the modeling tool. They agree however that this visualization should keep a balance of usable icons but not an overload of details. Other icons in the diagram are quite abstract yet useful. They all agree that a WebForm Diagram will improve the visibility of a form and that end users will gain insight. They also state that they want to use the modeling tool if they have the proper tools supporting it. However, a remark is made concerning the fact that the current application as developed in MetaEdit+ is not web based in contrast with the WCM system. It would therefore require a rebuild in a web based environment to get it into production.

6 Related Work

In the field of Web Engineering, there are several research groups working on the development of MDE products. We briefly elaborate on four relevant research groups.

One research which started early with MDE is the Object-Oriented Hypermedia Design Method (OOHDM) as proposed by Schwabe and Rossi [21], [22]. OOHDM comprises of four steps: conceptual modeling, navigational design, abstract interface design and implementation. These activities are performed in a mix of incremental, iterative and prototype-like way. In OOHDM a clear separation is made between on the one hand navigation and the other hand presentation. OOHDM has introduced powerful navigational concepts for contexts, personalization and user interaction.

Koch et al. describe the UML-based Web Engineering (UWE) approach in [12]. UWE is an object-oriented, iterative and incremental approach for the development of web applications. The development process of UWE consists of five phases: inception, elaboration, construction, transition and maintenance. The approach focuses mainly on customized and adaptive systems and also do not take content management and integration of external sources into account.

Pastor et al. describe different methods with the Object-Oriented Web-Solutions Modeling approach. OOWS provides mechanisms to deal with the development of hypermedia information systems and e-commerce applications in web environments [16]. OOWS strongly focuses on the generation of the required Web Application and less about managing the content and the application afterwards. OOWS extends OO-method by means of a navigation model, a presentation model and a user categorization. OOWS comprises of two main activities: system specification and solution development. Similar to our framework: the OOWS approach is supported by a commercial tool called OlivaNova.

Ceri et al. describe in [3] their Web Modeling Language (WebML), a notation for specifying complex web sites at a conceptual level. In line with the definition of Web Engineer, the WebML approach consists of seven phases: requirements specification, data design, hypertext design, architecture design, implementation, testing and evaluation and maintenance and evolution. The WebML method is supported by a commercial Model Driven development environment called WebRatio that allows modeling and automatic generation of Web interface applications.

These models are used within our research. We compared these models based on the Cooperative Requirements Engineering With Scenarios Framework (CREWS) [18].

7 Conclusion

The purpose of this research is to reduce complexity and increase the transparency of the development of online business processes supported by Web Content Management Systems. To achieve these goals, we defined modeling tool to automate the configuration of Web Content Management Systems with a strong focus on defining the business processes. Based on user analysis and domain modeling, we presented the WebForm Diagram which utilizes fragments of established Web Engineering methods. The WebForm Diagram consists of an abstract and concrete syntax resembling the implementation model and the mental model respectively. We developed a prototype of the model in MetaEdit+ and were able to automatically configure the WCMs. Several abstraction layers in the WebForm Diagram supports the different process steps in the implementation project. We validated the WebForm Diagram by means of a prototype validation and an expert evaluation. The results from both the prototype and user evaluation were positive and promising. However, the validation presented some technical limitations which need to be addressed. The most important limitation was the fact that the modeling tool was not yet Web enabled. Another aspect to consider is the fact that we use general-purpose languages such as Java and XLST in the prototype to transform the different models while upcoming model-to-model transformations seem promising [13]. We believe that we have made an improvement in an approach for WCMS-based Web Engineering and that concepts of this research can be applied beyond the scope of WCMS. Future research includes further development of the WEM Framework and refinement of the modeling tool to support the automated configuration of WCMS.

References

1. Brambilla, M., Preciado, J.C., Linaje, M., Sanchez-Figueroa, F.: Business process-based conceptual design of rich internet applications. In: ICWE 2008: Proceedings of the 2008 Eighth International Conference on Web Engineering, Washington, DC, USA, 2008, pp. 155–161. IEEE Computer Society, Los Alamitos (2008)
2. Brinkkemper, S.: Method engineering: Engineering of information systems development methods and tools. Journal of Information and Software Technology 38(4), 275–280 (1996)
3. Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Matera, M.: Designing Data Intensive Web Applications. Morgan Kaufmann, San Francisco (2003)
4. Cooper, A.: About Face 3: The Essentials of Interaction Design. Wiley, New York (2007)
5. Fein, R.M., Olson, G.M., Olson, J.S.: A mental model can help with learning to operate a complex device. In: CHI 1993: INTERACT 1993 and CHI 1993 conference companion on Human factors in computing systems, pp. 157–158. ACM, New York (1993)
6. O. M. Group. Unified modeling language: Infrastructure, version 2.0.(2005), `http://www.omg.org/docs/formal/05-07-05.pdf`
7. Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S.: Feature-oriented domain analysis (foda) feasibility study. Technical report, Carnegie-Mellon University Software Engineering Institute (November 1990)
8. Kappel, G., Prll, B., Reich, S., Retschitzegger, W.: Web Engineering: The Discipline of Systematic Development of Web Applications. Wiley, New York (2006)
9. Kelly, S., Lyytinen, K., Rossi, M.: Metaedit+: A fully configurable multi-user and multi-tool case and came environment. In: CAiSE 1996: Proceedings of the 8th International Conference on Advances Information System Engineering, London, UK, pp. 1–21. Springer, Heidelberg (1996)
10. Kent, S.: Model driven engineering. In: Butler, M., Petre, L., Sere, K. (eds.) IFM 2002. LNCS, vol. 2335, pp. 286–298. Springer, Heidelberg (2002)
11. Koch, N.: A comparative study of methods for hypermedia development. Technical Report 9905, Institut für Informatik der LMU (1999)
12. Koch, N., Kraus, A.: The expressive power of uml-based web engineering. In: Proceedings of IWWOST 2002, pp. 105–119 (2002)
13. Koch, N., Meliá, S., Moreno, N., Pelechano, V., Sanchez, F., Vara, J.M.: Model-driven web engineering. Upgrade-Novática Journal (English and Spanish), Council of European Professional Informatics Societies (CEPIS) IX(2), 40–45 (2008)
14. Luinenburg, L., Jansen, S., Souer, J., van de Weerd, I., Brinkkemper, S.: Designing web content management systems using the method association approach. In: Proceedings of the 4th International Workshop on Model-Driven Web Engineering (MDWE 2008), pp. 106–120 (2008)
15. Pastor, O., Fons, J., Pelechano, V., Abrahao, S.: Conceptual modelling of web applications: The oows approach. In: Mendes, E., Mosley, N. (eds.) Web Engineering: Theory and Practice of Metrics and Measurement for Web Development (2006)
16. Pastor, Ó., Abrahão, S., Fons, J.: An object-oriented approach to automate web applications development. In: Bauknecht, K., Madria, S.K., Pernul, G. (eds.) EC-Web 2001. LNCS, vol. 2115, pp. 16–28. Springer, Heidelberg (2001)
17. Ralyté, J., Brinkkemper, S., Henderson-Sellers, B.: Situational method engineering: Fundamentals and experiences. In: Proceedings of the IFIP WG 8.1 Working Conference, vol. 38(4), pp. XII + 368 (2007)
18. Rolland, C., Achour, C.B., Cauvet, C., Ralyté, J., Sutcliffe, A., Maiden, N., Jarke, M., Haumer, P., Pohl, K., Dubois, E., Heymans, P.: A proposal for a scenario classification framework. Requir. Eng. 3(1), 23–47 (1998)

19. Rolland, C., Prakash, N.: Bridging the gap between organisational needs and erp functionality. Requir. Eng. 5(3), 180–193 (2000)
20. Schmidt, D.C.: Guest editor's introduction: Model-driven engineering. Computer 39(2), 25–31 (2006)
21. Schwabe, D., Rossi, G.: The object-oriented hypermedia design model. Commun. ACM 38(8), 45–46 (1995)
22. Schwabe, D., Rossi, G., Barbosa, S.D.J.: Systematic hypermedia application design with oohdm. In: HYPERTEXT 1996: Proceedings of the the seventh ACM conference on Hypertext, pp. 116–128. ACM, New York (1996)
23. Smith, H.: Business process management–the third wave: business process modelling language (bpml) and its pi-calculus foundations. Information & Software Technology 45(15), 1065–1069 (2003)
24. Soffer, P., Golany, B., Dori, D.: Erp modeling: a comprehensive approach. Inf. Syst. 28(6), 673–690 (2003)
25. Souer, J., Honders, P., Versendaal, J., Brinkkemper, S.: A framework for web content management system operation and maintenance. Journal of Digital Information Management (JDIM), 324–331 (2008)
26. Souer, J., van de Weerd, I., Versendaal, J., Brinkkemper, S.: Situational requirements engineering for the development of content management system-based web applications. Int. J. Web Eng. Technol (IJWET) 3(4), 420–440 (2007)
27. Tolvanen, J.-P., Rossi, M.: Metaedit+: defining and using domain-specific modeling languages and code generators. In: OOPSLA 2003: Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, pp. 92–93. ACM Press, New York (2003)
28. van de Weerd, I., Brinkkemper, S., Souer, J., Versendaal, J.: A situational implementation method for web-based content management system-applications: method engineering and validation in practice. Software Process: Improvement and Practice 11(5), 521–538 (2006)
29. van Deursen, A., Klint, P., Visser, J.: Domain-specific languages: an annotated bibliography. SIGPLAN Not. 35(6), 26–36 (2000)