Towards a Pragmatic Model Driven Engineering Approach for the Development of CMS-based Web Applications

Jurriaan Souer and Thijs Kupers

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

Abstract. Most enterprises utilize Web Content Management System (CMS) for the development and maintenance of their web applications. A CMS provides a high quality platform and creates web applications based on software configuration instead of software engineering from scratch. Although there are numerous advantages to implementing a CMS, there are two downsides not solved: for complex applications there is still a need for software developers and architects to configure the software based on business-user requirements, and there is no easy way to see how the application is implemented. This paper presents a pragmatic Model Driven Engineering method that allows the business-users to create CMS-based web applications themselves based on a business model without the need for software engineering and architects. With these business-users in mind, we minimized complexity by implementing only the modeling tools they need resulting in a useful web form diagram. The web form diagram allows business users to model generic business processes which are transformed into a CMS-specific configuration. The web form Diagram is implemented in a CASE environment a validated by means of a case study and an end user evaluation.

1 Introduction

Most enterprises use Web Content Management software (CMS) to develop and maintain their web applications. A CMS is a software product with out-of-the-box functionalities for web applications and provide technical users with a standardized platform for web development [15]. Most CMS-based web applications can be realized with merely configuration of the CMS without writing actual code. Moreover a CMS allow non technical users – or business users – to manage, control and update their web applications without the need for technical support [16]. Still, configuration of a CMS for the creation of more advanced and interactive web application can be complex and time consuming. Business analysts know exactly what the interaction should look like, but they only define the user requirements. This paper addresses the problem of how business users would be able to generate interactive CMS-based web applications based on their user requirements.

Model Driven Engineering (MDE) is an evolving and promising approach to software development which could help bridge the gap between requirements analysis and implementation [11]. MDE approaches deal with the provision of models, transformations between them and code generators to address software development. One of the main advantages of this approach is the definition of a conceptual structure where the models used by business analysts can be traced towards more detailed models used by software engineers. MDE proposals, and more specifically its OMG specification, the MDA (Model Driven Approach) [3], constitute an important tool for the alignment of the business view and the information system view. MDA is a relatively new paradigm, which aims at providing a standard baseline for model-driven development. The goal of MDA is to (semi)automate the process of software development from requirements to code using an interoperable set of standards. Although MDA provides useful insights and constructs on how to develop software based on models, its generic application makes it complex for business analysts. Others have tried Use Case based tools/techniques [14], [17] but the generic characteristics which makes this approach applicable in most areas prevent business users from accepting it because of complexity. In this paper we develop a pragmatic MDE approach with a single purpose: to allows business users to configure a CMS to create an interactive web application based on a user requirement model.

In [16] and [15] we introduced the Web Engineering Method (WEM) as a web engineering approach for the implementation of web content management systems in particular to obtain high maintainability and give business owners the ability to manage and control web applications. We call these type of web applications CMS-based web Applications. WEM is integrated in a traditional implementation method consisting of Orientation, Definition, Design, Realization and Implementation. This paper continues this research and elaborates on the Design phase. In [7] we defined the Design phase of WEM. This paper continues this research with an implementation of the Business Process Model from the Design Phase of WEM.

The rest of this paper is organized as follows: In Section 2 we describe the Approach we used to analyze the problem area. Section 3 elaborates on all the aspects of the model which we call the web form diagram. In section 4 we describe the implementation of the web form diagram, a case description and some considerations. We validated the resulting web form diagram with the end users which is described in section 5. Finally conclusion are discussed in Section 6.

2 Approach

As described in the introduction, the goal of this research is to let business analysts define requirements which will configure a CMS for the creation of a web application. This requires a logical sound requirements model which can be transformed into a configuration of a CMS. A similar construction can be found in the MDA literature where they are referred to as a Platform Independent Model (PIM) and a Platform Specific Model (PSM) [3]. Since we want business analysts model a CMS we answer the following questions:

- what elements of a CMS need to be modeled;
- how the business analysts would model the diagram in an ideal situation; and

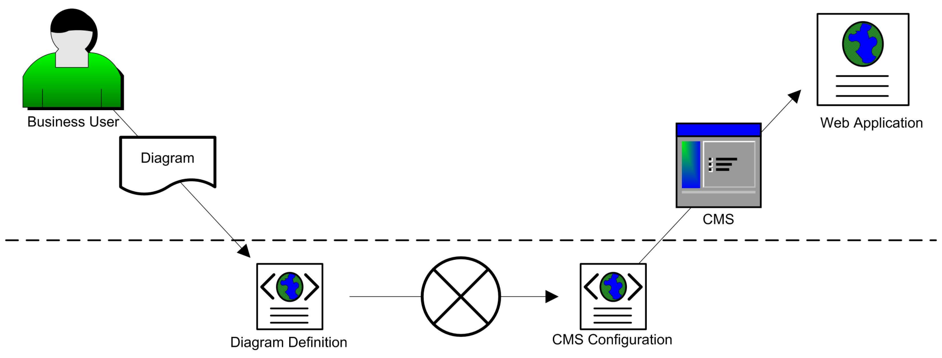

Fig. 1. approach

- what existing method fragments could we use in our case to prevent ‘reinventing the wheel’.

The first question investigates what aspects of the CMS should be modeled. Since the output should be a configured CMS, we use Domain Modeling to define which aspect of the CMS need to be configured. If we know what aspects of the CMS need to be configured, the second question is about how the business analysts would ideally model these aspects. The resulting model will provide input for the configuration of the CMS. We use a User Analysis to analyze how they would ideally create the model. Finally, if we know what we want to model and how the business users want to model this, we analyze existing methods to find if we could reuse certain method fragment of these methods. In [7] we introduced the Method Association Approach in which we compared existing methods. This paper extends the table with the CREWS framework [10]. We detail these steps in the following sections.

2.1 Domain Modeling

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. As mentioned in the introduction, we focus on the definition of an interactive web form. This research is performed within GX – a vendor of web content management Software called ‘GX WebManager’. A for this research relevant functional component within the CMS is the ‘Advanced Form Module’: a functional component which allows editors to develop business processes based on advanced forms. The Advanced Form Module has been developed to create interactive forms with all available formcontrols as defined by the World Wide Web Consortium (W3C). An excerpt of the domain model is displayed in Figure 2: a **form** consists of one or more **steps** (which are the screens displayed to a website visitor). A step has zero or more **formelements** which are the input types of a form (e.g. text, list, radiobutton, hidden, passwordfield, textarea, etc). A step can have zero or more **subforms** which is

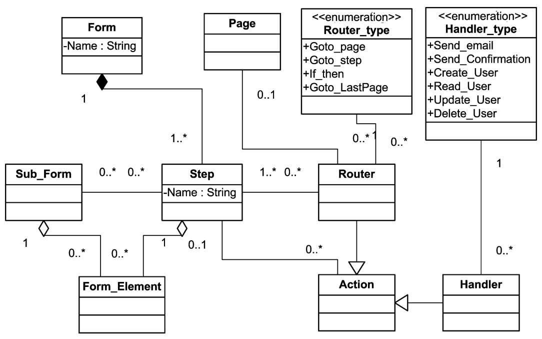

Fig. 2. Excerpt of Domain Model

a collection of formelements and allows the reuse of formelements (such as first name, last name, gender, residence).

Using a domain model, we identified all the elements that need to be modeled by the business analysts.

2.2 User analysis

In the next step we investigated how end users would ideally model the web form. We interviewed six typical end users who implement business processes during CMS implementations: 3 senior functional consultants and 3 senior architects. The group of interviewees includes both functional and technical users since they both would gain a lot with a modeling tool. The six users were interviewed individually and were asked how they would define a business process. Moreover, they were given an empty page and a pencil and were asked to draw a visual model of a business process. We then analyzed the drawings and abstracted it to a more generic model and refined it in a second iteration of interviews. In the end, the model defined by the end users should be transformed into the aspects as defined in the domain model.

All six users defined a business process as a set of *Steps* – representing a screen which a website visitor would see – and a *Flow* defining the order of the steps. They visualized the process as a collection of square blocks with arrows in between (representing the order of the steps). Users expected two levels of abstraction in the model:

1. A high level abstraction allowing them to model the complete process
2. A more detailed level specifying the formelements of the steps

On the detailed level, each step has multiple input fields from a certain type (e.g. text field, password field, radio button, etc), buttons and information (for example a help-field). Two architects suggested adopting Business Process Modeling Language (BPMN) as a solution for the modeling language especially since content management systems are often integrated into other systems and BPMN is a well known standard for defining processes [13] and used within web engineering [1].

The order of the steps was visualized with arrows. If the order 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 actions (known as ‘handlers’) except for the idea of using an object to define that there should be a handler on that specific place. The consultants suggested that they could use a temporary block in the form which they would specify later. One illustration drawn by a consultant and one drawn by an architect is displayed in Figure 4.

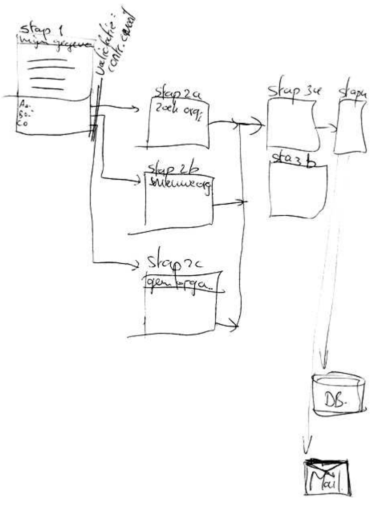

Fig. 3. Artifact created by Consultant

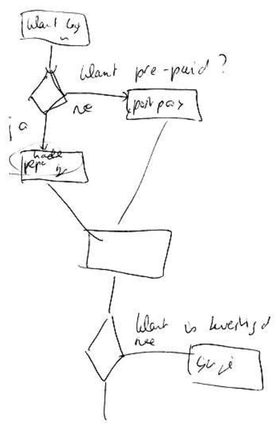

Fig. 4. Artifact created by Architect

2.3 Comparing existing methods

In this section we describe the analysis of existing web application modeling approaches in order to find which aspects we could use. We continue with our previously filled method base which we gathered using Situational Method-Engineering [9] , consisting of the following approaches: Object Oriented Hypermedia Design Model [12], WebML[2], UML-based Web Engineering [5] and Object Oriented Web Solutions [8].

In [7] 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) [10]. We adapted the CREWS framework, analyzing the methods based on different views.

- **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.

Based on the three views, we conclude that the Activity Diagram (UWE) and the Business Process Model (OOWS) are the closest to our target model since they provide a way to define a form, have multiple abstraction levels and have a functional perspective. 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 actually does represent the form elements in a well-organized way. The UML Class Diagram is also interesting since it can define different form elements but has a rather generic application and will there for lack an end user acceptance. The Business Process Diagram from WebML has other options to model these web forms, but has a less compact notation to define the user interaction with the system. Taken all these remarks into account, we selected the Business Process Model (BPM), the User Interaction Diagram (UID) and the UML Class Diagram as the base models for our modeling language.

3 Defining a Graphical Model for the Web form Diagram

Now we now what needs to be modeled, how the end user would ideally model it and what method fragments we intend to reuse, we define our graphical model for the web form diagram. The web form diagram consists of a set of nodes with edges in between. Each node can be either one of the following types: step, form element, validator, handler, condition, webpage, and block. We detail these concepts in the following sections.

The web form consists of a set of **Steps** and a Flow between those steps. A Step is similar to an Activity (BPM), Interaction (UID) and Class (UML). A step is the foundation of a form. Each step can be detailed with multiple **Form Fields**. A Form Fields is similar to the Data Entry (UID) or Attribute (UML). It is a superclass of different formfield types such as textinput, single selection (radiobutton), multiple selection (checkbox), etc. As a result of the Domain Model, we know that each formfield is connected to a single step. The concepts of a Step and Form Fields are visualized in Figure 5: On the left side you see a single step which can be detailed with Form Fields as presented on the right side.

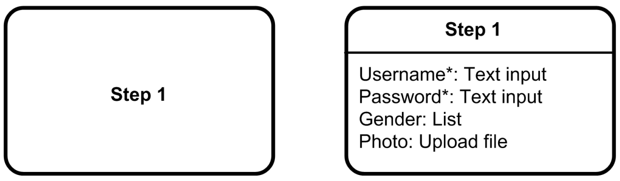

Fig. 5. A single step and a step with Form Fields

A **Validator** could be implemented in UML as an Operation which would check certain input and will return a boolean. The user analysis learned that a validator is ideally visualized as a gateway which will only allow the flow to continue if certain conditions are met as can be seen in Figure 4. As the result of the Domain Model, we identified two types of Validators: Field-Validators and Form-Validators. A Field-Validator validates the user input of a single Form Field (e.g. check if the field Username is not empty). A Form-Validator is used to check a complete Step (e.g. when signing in both the 'Username' and 'Password' should be valid or the Username in Figure 5 is 'Required'). In the graphical model we visualize the Validator as separate element between two steps and is visualized in Figure 6.

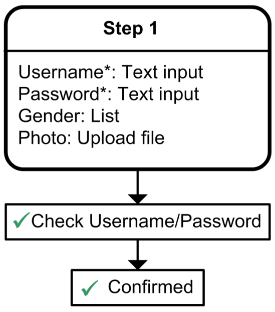

Fig. 6. Visualization of two Validators

Handlers can also be seen as Operations in UML. The execution of a Handler takes place between two steps – similar to Form-Validators. Handlers are therefore visualized as separate nodes in the form flow as illustrated in Figure 7. Within the CMS, there is a large set of Handlers available by default. The graphical notation consists of an icon with below the name of the Handler where the icon represents the type of Handler. In the example there are two Handlers: one reading from a database and one sending an e-mail. With the icon illustrating the type of Handler it is easier for business analysts to select the needed Handlers.

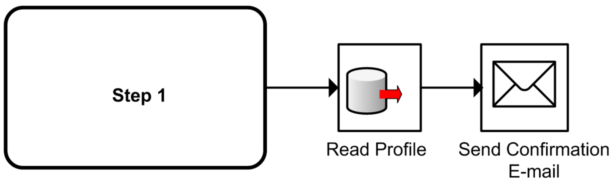

Fig. 7. Two Handler examples

A **Condition** is a decision point in a form flow and is similar to an Operation (UML) or a Gateway (BPM). A condition will determine forking and merging of paths depending on the conditions expressed and is visualized with a diamond containing the name of the condition as illustrated in Figure 8. In the figure, a Condition checks whether a user is logged on and will route the user to the next step depending on the outcome of the Condition. In this particular example the user will be routed to a **Page** on the website if the user is not logged on.

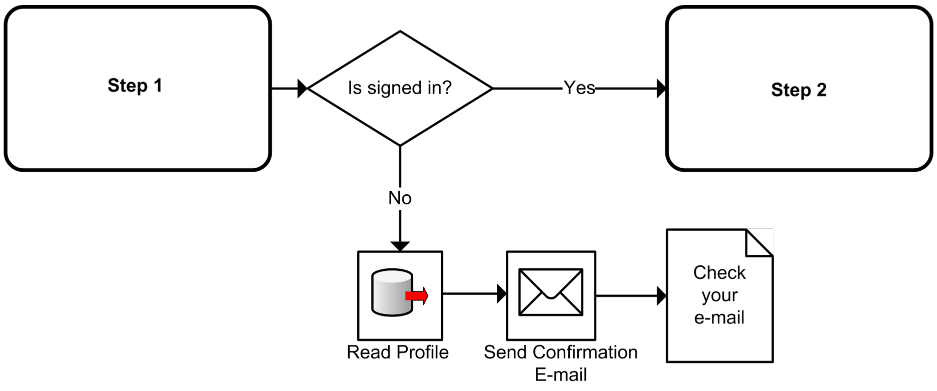

Fig. 8. A Condition and a routing to a Page

Business analysts do not always know all the steps and contents of the complete form in advance. One of the outcome of the user analysis was that business analysts should be able to define a temporary Step which they could specify in a second iteration. This however is not part of the Domain Model but was found as a necessity by the users. We therefore introduce a **Block** and will be visualized as a temporary node containing a description in natural language. An example is illustrated in 9. WebML has a similar construct in the Business Process Diagram.

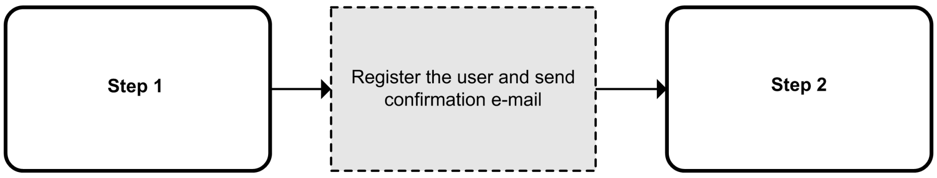

Fig. 9. A temporary Block between Step 1 and Step 2

To conclude, all the different elements are connected with directed **Edges**. An edge is comparable to a Directed Association (UML) and a Sequence Flow (BPM). Each Edge has a starting point (*source*) and an end point (*target*). Conditions have at least two outgoing Edges since there is at least one different routing of the flow. Moreover, Edges from a condition have a *case* which determines if the Edges is to be followed or not (depending of the outcome of the condition). All other nodes have a maximum of one outgoing Edge. A Step can be the last Step of the form and does not necessarily have any outgoing Edges. Other nodes (Form-Validators, Handlers, and Blocks) are never an end point (a website visitor will always see a Step or a WebPage) and they always have one outgoing edge. Edges are illustrated with arrows and can be seen in all previous examples.

Now we have defined all constructs which are needed to create a complete web form from the business analyst's perspective. A complete form consisting of all elements is displayed in Figure 11 which we will discuss in Section 4. To make the model logically sound we defined a concrete syntax of our model.

3.1 Transforming the Model to a CMS Configuration

The next step in our research is to make a CMS-configuration based on the web form diagram. Most of the defined concepts – such as Step, Validation, Handler, Page – are available in the CMS which we know because of the Domain Model. With some transformations, most concepts could therefore be applied directly to a CMS configuration. However, there are some concepts that are not the same as the configuration parameters of the CMS. For example, what end users defined as the 'flow' which became an Edge in our model can in fact become either one of two separate concepts in the CMS: a *sequential number* which determines the order in which the nodes are executed, or a *Router* which defines the next Step or Page. A router is a concept in the CMS which was not addressed explicitly end users. Also the definition of the router depends on Condition concepts of the web form diagram. Conditions are not available in the CMS as similar concepts but are parameters of the objects Handler or Router. Moreover the 'Required' attribute of a Form Field must be translated into a Validator in the CMS. And Blocks are not available at all in the CMS and cannot be configured in the CMS.

To make a CMS configuration out of the concepts of the web form diagram, we need to transform the model. In de pseudocode below we convert the web form diagram into CMS web form diagram. In the pseudocode, the web form diagram is regarded as a directed graph. For each Edge the function ADDCHILD is invoked, which iterates recursively over all nodes and adding these nodes to a Step. When it runs into a Condition-node, the precondition is adjusted and added to the True-cases recursively. When an target-node of an Edge refers to a Step or Page, the Edge is converted to a Router. And the function 'createName(node, condition)' uses the type of target-node and a condition to return the correct label for the router.

Algorithm 3.1: CONVERT()

```
for each $(s \in Step)$
  do for each $(e \in \{x | sourceE(x) = s\})$
  do $addChild(s, e, o, null)$
```

Algorithm 3.2: ADDCHILD$(s, e, index, cond)$

```
$n = targetE(e)s$
if $(n \in C)$
  then
    $condT = cond \wedge xslCondition(n)$
    for each $(e_2 \in \{x|sourceE(n) = s \wedge caseE(n) = T)$
      do $index = addChild(s, e_2, index, condT)$
    for each $(e_3 \in \{x|sourceE(n) = s \wedge caseE(n) \neq T)$
      do $index = addChild(s, e_3, index, cond)$
  else if $(n \in S \| n \in P)$
    then
      $r \in R'$
      $r \in routersS'(s)$
      $precondition'(r) = cond$
      $targetR'(r) = n$
      $ordernr'(n) = index$
      $index + +$
      $typeO'(r) = r_t$
      $nameT'(r_t) = createName(n, cond)$
  else
    $ordernr'(n) = index$
    $index + +$
    $precondition'(n) = cond$
    if $(n \in FORMVALIDATOR)$
      then $n \in validatorsS'(s)$
    if $(n \in H)$
      then $n \in handlersS'(s)$
    for each $(e_2 \in \{x|sourceE(n) = s\})$
      do $index = addChild(s, e_2, index, cond)$
return $(index)$
```

4 CASE Application Supporting the Web Form Diagram

We have developed a tool to implement the web form diagram as a prototype. We used MetaEdit+ [4] to build our prototype of the web form Diagram. This application is both proven in building CASE tool as well as also provides computer aided support for method engineering [18].

The integration of MetaEdit+ and the CMS is facilitated by XML: the web form is exported from MetaEdit+ as an XML file. However, this XML file is MetaEdit+ specific and we transform this XML through XSLT towards our web form diagram compliant XML. We call this XML the GXML. Since it is a prototype, we have written the conversion in general-purpose languages Java and XSLT. The GXML need to be transformed as described in the pseudocode to make it in line with the Domain Model. This transformation is written in Java for speed of development purposes while upcoming model-to-model transformations seem promising [6]. The resulting XML is called GXML' and has all the concepts which are available in a CMS. Through an XSLT transformation the GXML' is then transformed into GX WebManager specific XML which can be imported into that specific CMS. Figure 10 gives an overview of the performed transformations. The transformation process works both ways: CMS form definitions can be exported to XML and can be transformed and imported into back into MetaEdit+ which provides business analysts with a useful tool to analyze implementations and make adjustments from a process perspective.

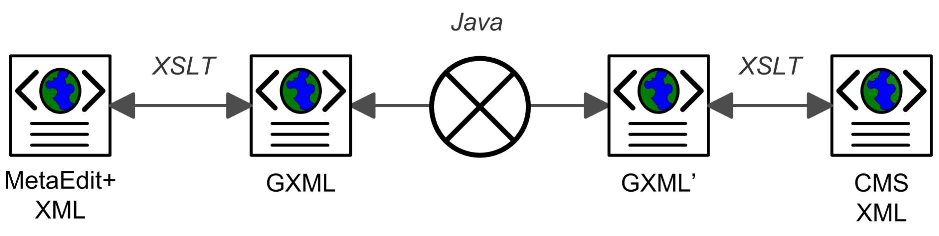

Fig. 10. transformation

The XSLT transformations are not necessary but it allows us to change modeling tool or the implementation of the CMS without having to change the GXML interface defining concepts of the web form diagram and the CMS concepts. The GXML-definition is described in an XML-Schema which contains the restrictions as the result of the web form diagram definition. All elements in the GXML (such as Steps, Form Fields, Validationrules, Handlers, Parameters, Pages, Conditions and Edges) have a unique identifier. This identifier is used for reference purposes: a Step for instance has multiple Form Fields which are referenced through this identifier. An excerpt of GXML is displayed in the excerpt below.

Listing 1.1. GXML Excerpt

```
<form id="_3_3706">
  <name>Sign in</name>
  <steps>
   <step id="_3_3740">
     <name>Stap 1</name>
         <field ref="_3_3728" />
         <handler ref="_3_3773" />
         <router ref="–2a327p0" />
  </step>
  <fields>
    <field id="_3_3716">
          <name>Username</name>
          <type>TextField</type>
        </field>
        <field id="_3_3728">
          <name>Password</name>
          <type>TextField</type>
        </field>
  </fields>
  <validators>
```

4.1 The Web Form Diagram Modeling Tool in Practice

We tested the application in practice at GX where we took a customer case implemented a web form from an existing website using the modeling tool. The resulting form of this case is displayed in Figure 11. The model shows a user registration scenario: 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 databases access are initialized. The complete implementation has been tested both ways which means we designing a web form diagram in MetaEdit+, imported the XML into the CMS; and the same process in the opposite order to visualize a defined web form in MetaEdit+. The import went without any problems and 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.

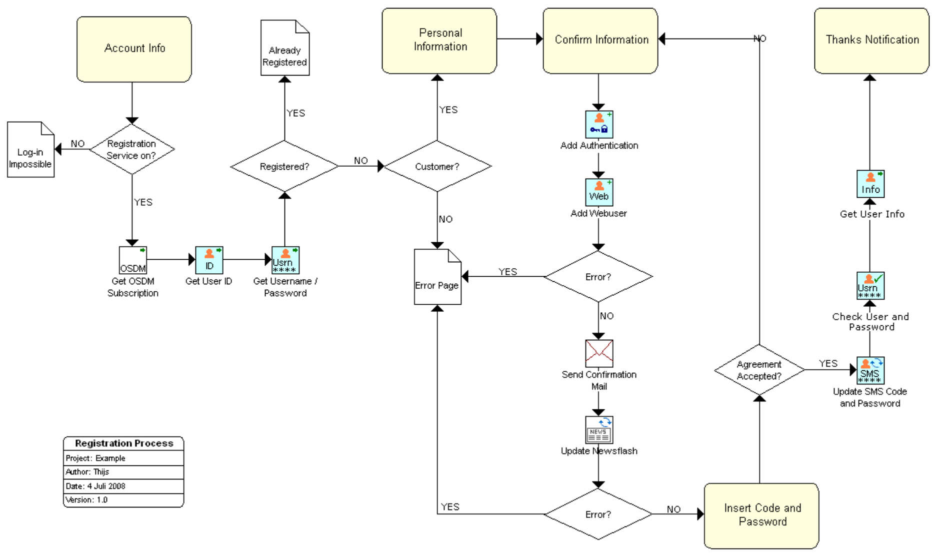

Fig. 11. Example of a business process modeled with the web form Diagram

4.2 Considerations

Based on the functional validation, some limitations concerning this implementation where identified preventing it from putting it directly into practice. We detail some considerations.

Within the CMS, there is an extensive library of existing handlers and routers. The case we implemented however had a few specific handlers which were not available by default. This resulted in a change in the web form diagram: a new placeholder handler during the modeling phase which needs to be configured or developed within the CMS 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 web form Diagram.

5 End user evaluation

The goal of this research is to allow end users create CMS-based web applications based on a requirements model. We have proven that we are able – with some limitations – to create CMS-based web applications based on the model. However, the end user evaluation will decide how useful the modeling tool is. To find out how useful the modeling tool is for CMS implementations, we interviewed eight experienced end users: four business analysts and four architects. Each user was confronted with the model as illustrated in 11 and were asked to rate (1 to 4) both the model and the modeling tool based on usability, transparency, suitability and applicability in projects. The users were then asked to substantiate their ratings during an interview.

The results show that the end users found the web form diagram easy to read and unambiguous. They would certainly want to use the model in projects since it would improve communication internally and externally. It would also improve the validity of requirements since customers would probably understand these diagrams. There was some discussion concerning the details of the web form model: what icons should be displayed to what extent. The general opinion was 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.

They also stated that they wanted to use the modeling tool if they had the proper tools supporting it. However, a strong remark was made concerning the fact that the current application as developed in MetaEdit+. Since MetaEdit+ is not web based in contrast with the CMS, and would therefore not be regarded as useful.

6 Conclusion

In this paper we have discussed a pragmatic Model Driven Engineering approach for the creation of CMS-based web applications. Based on end user analysis, a Domain Model and existing method fragments, we detailed the web form model. We have shown that the web form diagram works in practice with a prototype of the model in MetaEdit+

and were indeed able to automatically configure the CMS, although there were some limitations. An end user evaluation learned that the model itself is regarded as quite useful and has a lot of potential. However, the modeling tool not being web enabled will stop the end users from putting it into practice. Another aspect to note is that we validated the web form diagram in one single CMS. Since the web form diagram is defined conform W3C standards, we do believe that the modeling tool could be used to configure other CMS systems but it would probably require a different transformation. Our transformation mechanism however takes this possibility into account. Future research includes further development of the WEM Framework and refinement of the modeling tool to support the automated configuration of CMS.

References

1. M. Brambilla, J. C. Preciado, M. Linaje, and F. Sanchez-Figueroa. Business process-based conceptual design of rich internet applications. In *ICWE '08: Proceedings of the 2008 Eighth International Conference on Web Engineering*, pages 155–161, Washington, DC, USA, 2008. IEEE Computer Society.
2. S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, and M. Matera. *Designing Data Intensive Web Applications*. Morgan Kaufmann, 2003.
3. O. M. Group. Unified modeling language: Infrastructure, version 2.2. *http://www.omg.org/docs/formal/09-02-02.pdf*, 2009.
4. S. Kelly, K. Lyytinen, and M. Rossi. Metaedit+: A fully configurable multi-user and multi-tool case and came environment. In *CAiSE '96: Proceedings of the 8th International Conference on Advances Information System Engineering*, pages 1–21, London, UK, 1996. Springer-Verlag.
5. N. Koch. A comparative study of methods for hypermedia development. Technical Report 9905, Institut fr Informatik der LMU, 1999.
6. N. Koch, S. Meliá, N. Moreno, V. Pelechano, F. Sanchez, and J. M. Vara. Model-driven web engineering. *Upgrade-Novtica Journal (English and Spanish), Council of European Professional Informatics Societies (CEPIS)*, IX(2):40–45, 2008.
7. L. Luinenburg, S. Jansen, J. Souer, I. van de Weerd, and S. Brinkkemper. Designing web content management systems using the method association approach. In *Proceedings of the 4th International Workshop on Model-Driven Web Engineering (MDWE 2008)*, pages 106–120, 2008.
8. O. Pastor, J. Fons, V. Pelechano, and S. Abrahao. Conceptual modelling of web applications: The oows approach. *E. Mendes and N. Mosley (eds.) Web Engineering: Theory and Practice of Metrics and Measurement for Web Development*, 2006.
9. J. Ralyté, S. Brinkkemper, and B. Henderson-Sellers. Situational method engineering: Fundamentals and experiences. *Proceedings of the IFIP WG 8.1 Working Conference*, 38(4):XII + 368, 2007.
10. C. Rolland, C. B. Achour, C. Cauvet, J. Ralyté, A. Sutcliffe, N. Maiden, M. Jarke, P. Haumer, K. Pohl, E. Dubois, and P. Heymans. A proposal for a scenario classification framework. *Requir. Eng.*, 3(1):23–47, 1998.
11. D. C. Schmidt. Guest editor's introduction: Model-driven engineering. *Computer*, 39(2):25–31, 2006.
12. D. Schwabe and G. Rossi. The object-oriented hypermedia design model. *Commun. ACM*, 38(8):45–46, 1995.

13. H. Smith. Business process management–the third wave: business process modelling language (bpml) and its pi-calculus foundations. *Information & Software Technology*, 45(15):1065–1069, 2003.
14. S. S. Som. Supporting use case based requirements engineering. *Information & Software Technology*, 48:2006, 2006.
15. J. Souer, P. Honders, J. Versendaal, and S. Brinkkemper. A framework for web content management system operation and maintenance. *Journal of Digital Information Management (JDIM)*, pages 324 – 331, 2008.
16. J. Souer, I. van de Weerd, J. Versendaal, and S. Brinkkemper. Situational requirements engineering for the development of content management system-based web applications. *Int. J. Web Eng. Technol. (IJWET)*, 3(4):420–440, 2007.
17. A. G. Sutcliffe, N. A. Maiden, S. Minocha, and D. Manuel. Supporting scenario-based requirements engineering. *IEEE Transactions on Software Engineering*, 24(12):1072–1088, 1998.
18. J.-P. Tolvanen and M. Rossi. Metaedit+: defining and using domain-specific modeling languages and code generators. In *OOPSLA '03: Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pages 92–93, New York, NY, USA, 2003. ACM.