Development of CMS-Based Web-Applications Using a Model-Driven Approach

Conference Paper · September 2009

DOI: 10.1109/ICSEA.2009.79 · Source: DBLP

2 authors:

João Saraiva
Independent Researcher
24 PUBLICATIONS **189** CITATIONS
SEE PROFILE

Alberto Rodrigues Da Silva
University of Lisbon
271 PUBLICATIONS **2,345** CITATIONS
SEE PROFILE

Development of CMS-based Web-Applications Using a Model-Driven Approach

João de Sousa Saraiva, Alberto Rodrigues da Silva
INESC-ID / Instituto Superior Técnico,
Rua Alves Redol, 9, 1000-029 Lisboa, Portugal,
joao.saraiva@inesc-id.pt, alberto.silva@acm.org

***Abstract*—The emerging Model-Driven Engineering paradigm advocates the use of models as first-class citizens in the software development process, while artifacts such as documentation and source-code can be quickly produced from those models by using automated transformations. One of the many types of deployment platforms that can potentially benefit from such model-driven approaches are Content Management Systems, as these approaches can significantly accelerate the development of new web-applications and features, as well as simplify their maintenance. This work proposes the creation of a model-driven approach for the development of web-applications based on Content Management Systems. This approach is based on the creation of two modeling languages (which are situated at different levels of abstraction, and are used to both quickly model a web-application and provide a common ground for the creation of additional languages), and a mechanism for the processing of models specified using those languages. The current results of this work so far are the development of a Content Management System that effectively supports web-applications of medium complexity, and the creation of a reference case study that will be used to validate this work.**

Keywords: Software Design; Software Construction; Software Engineering Tools and Methods.

I. Introduction

The expansion of the Internet in the last years has made it a powerful platform for the deployment of a variety of artifacts and systems. This has led to the appearance of a myriad of frameworks and libraries[1] that attempt to harness the power of Internet-based technologies in order to accomplish various objectives. An (increasingly popular) example of this are the many web-oriented CMS (Content Management System) [1]–[5] and ECM (Enterprise Content Management) [6]–[9] systems that are available today, with the objective of facilitating the management and publication of digital contents on an intranet, or even on the Internet.

Although it is usually not their main goal, CMS systems can also be used as support platforms for web-applications to be used in the dynamic management of websites and their contents [10], [11]. These systems typically present aspects such as extensibility and modularity, independence between content and presentation, support for several types of contents, support for access management and user control, dynamic management of layout and visual appearance, or support for workflow definition and execution. On the other hand, ECM systems are typically regular web-applications oriented towards using Internet-based technologies and workflows to capture, manage, store, preserve, and deliver content and documents in the context of organizational processes [7]. Nevertheless, these two areas are not disjoint [6], and it is not unusual to find a CMS system acting as a repository for an organization's documents and contents, albeit at a very "primitive" level (e.g., no checking for duplicate information, no logical grouping of documents, and no support for providing metadata for each document). Also, a CMS platform can be used to perform the same tasks as an ECM platform, as long as the CMS provides the developer with adequate functionality and hooks [11].

Due to the different objectives of each framework (which, in turn, influences the nature of the hooks and functionality provided by the framework), they are often "accompanied" by approaches to the manual development of simple web-applications based on that framework. It is important to note that, although these CMS frameworks are themselves web-applications, they also provide a set of "high-level" concepts – such as user, role, or module – that are necessary to develop more complex web-applications (thus enabling the creation of web-applications that are themselves supported by other, lower-level, web-applications). Additionally, CMS-based web-applications are limited only by the underlying CMS itself (and its technology, of course). Of particular importance to this work are *model-driven approaches*, which are becoming increasingly popular due to the emergence of the Model-Driven Engineering (MDE) paradigm [12], [13]. MDE advocates the use of models as first-class citizens in the software development process, while artifacts such as documentation and source-code can be produced from those models by using automated transformations.

We believe that CMS systems have the potential for becoming the next generation of web-application frameworks, as they provide most (if not all) of the functionality that can be found in regular frameworks, and provide facilities for addressing typical issues that must be repeatedly addressed by developers (e.g., the management of users and roles). We

[1]The main difference between a *framework* and a *library* is that a library is just code that the developer can (re)use in the application, while a framework typically provides: (1) a number of "hooks" to which a developer can/must provide functionality; and (2) generic functionality that the developer can choose to override or specialize. In the context of web-applications, the terms *framework* and *platform* are usually interchangeable.

also believe that CMS systems can benefit from MDE-based development approaches, as these approaches can accelerate the development of new features, the rapid deployment, as well as simplify their maintenance. It is the goal of this work to propose a MDE-based development approach for the development of web-applications on CMS platforms.

This paper is organized into six sections. Section 1 introduces the context of development approaches for CMS-based web-applications. Section 2 summarizes the current state-of-the-art regarding the area of this work. Section 3 presents the research objectives of this work, and our intended approach to achieve them. Section 4 describes our progress so far, and Section 5 presents our plan for the remainder of this work. Finally, Section 6 presents the main conclusion.

II. State-of-the-Art

This work addresses the field of development of web-applications based on CMS systems. Although some proposals exist regarding languages for *generic* web-application development (of which we highlight WebML, UWE, and XIS2), we have not found any related work regarding the use of CMS systems as platforms for web-applications.

The Web Modeling Language (WebML) [14]–[16] addresses the high-level, platform-independent graphical specification of web-applications (which can be supported by a CASE tool called WebRatio) and targets web sites that require such advanced features as the one-to-one personalization of content and the delivery of information on multiple devices (e.g., PCs, PDAs, WAP phones) [17]. The specification of a site in WebML consists of four perspectives [18]: (1) the Structural Model, which expresses the data content of the site, in terms of the relevant entities and relationships; (2) the Hypertext Model, describing the hypertext contents that can be published in the site, as well as the navigation between those different hypertext contents; (3) the Presentation Model, which expresses the layout and graphic appearance of pages, independently of the output device and of the rendition language, by means of an abstract XML syntax; and (4) the Personalization Model, in which users and user groups are explicitly modeled in the form of predefined entities called User and Group, whose features can be used for storing individual or group-specific content.

The UML-based Web Engineering (UWE) [14], [19] is a software engineering approach for development of applications in the web domain, based on OMG standards (e.g., UML, MDA, OCL, XMI), that focuses on models and model transformations, more specifically on systematization and automatic generation. The UWE notation is defined as a UML profile, tailored for an intuitive modeling of web-applications [20]; because of its compliance with standards, UWE can be used in existing UML tools or as plug-ins. Its main characteristic is the use of UML for all models, in particular [21]: (1) using "pure" UML whenever possible; and (2) for web-specific features, such as nodes and links of the hypertext structure, the UWE profile includes stereotypes, tagged values and constraints defined for the modeling elements. UWE comprises [21]: (1) a modeling language for graphically representing web-application models; (2) a method supporting semi-automatic generation; and (3) a process for the development life-cycle of web-applications.

The "eXtreme Modeling Interactive Systems" (XIS2, typically just called XIS for simplicity) language [22] is also defined as a UML profile, and it is oriented towards interactive systems for different platforms, such as desktop, web, or mobile platforms, instead of just web-based platforms. The XIS2 language defines six types of models: (1) the Domain View; (2) the Business-Entities View; (3) the Actors View; (4) the Use-Cases View; (5) the User-Interfaces View; and (6) the Navigation View. Although the Domain View, the User-Interfaces View, and the Navigation View are conceptually similar to what can be found in UWE, XIS explicitly addresses the behavioral aspect (through the capture of user-interface patterns), enabling interaction between users and computer applications.

Although these languages and approaches do not address CMS-based development, we do not consider this to be a key problem because, until recently, CMS systems did not provide the necessary functionality to be considered adequate in the development of a web-application (in fact, currently only a few CMS systems do). However, our analysis of these languages does present some limitations (that, nevertheless, we believe are not exclusive to the languages mentioned in this section), the most relevant of which are that they either: (1) try to address multiple abstraction levels simultaneously [19], [22]; or (2) require additional work besides the specification of the web-application's model, to address low-level details [15]. For example, although these languages provide a relatively quick way of creating a web-based application, low-level details (e.g., the number of columns in a table, or whether the rows in a table should be colored in an alternate fashion) are typically considered as "implementation details" that should not be specified in a high-level language (although CSS-based – or similar – guidelines and techniques can be used to somewhat mitigate this) [15], a guideline with which we agree. On the other hand, when a language does provide a way to address such low-level details, it is usually by means of "property" mechanisms such as UML's tagged-values [19], [22], which leads, in practice, to the lowering of the language's abstraction level, because those properties can/should only be used by developers that are familiar with the low-level details that those properties represent.

Although it can be argued that, after generating the web-application, developers can directly edit the obtained source-code, one of the main objectives of MDE is precisely to avoid editing of the generated low-level artifacts (such as source-code), because this would force developers to (re)adopt the "traditional" development approaches at some

point in the web-application's development (if so, MDE-based approaches would present no real advantage over traditional development approaches).

Although our proposal also intends to deal both with "quick and easy" modeling and with low-level details, it does not intend to define a single language that solves these problems, as we believe that concentrating all these details into a single language is what actually triggers these problems in the first place. Instead, we propose to define a *set of languages* (situated at different levels of abstraction) and use a model-oriented variant of the well-known *compilation* process: a high-level modeling language (which is actually a set of mnemonics for a lower-level language) will be used to quickly specify a web-application, and a low-level language will be used to address details regarding CMS-based implementation. This proposal is further explained in the next section.

We have not found any related work regarding the use of CMS systems as platforms for web-applications (although some CMS systems are starting to evolve towards this end, such as Drupal [23] or WebComfort [11], [24]). However, we have noticed that proposals addressing parts of our intended work are beginning to surface. A good example can be found in [25], which presents the creation of a model interpreting web-application (designated "Integration Generator", conceptually similar to our own notion of "CMS Model Interpreter") that interacts with the Limestone CMS [26]. This application receives a XML file (which, in turn, results from the processing of a UWE model file – in XMI – with a XSLT style sheet), and is responsible for configuring the target CMS system, interacting with the backing data-store (such as a database server), and generating the necessary support files (e.g., ASP.NET pages and user controls).

III. Research Objectives and Approach

The research hypothesis that this thesis intends to address is that "development of CMS-based web-applications can be improved by means of a MDE-based approach". To address this thesis statement, we are to define an adequate approach, and evaluate whether web-application development using that approach presents concrete advantages over a traditional (source-code-based) development approach.

This approach should address at least the specification of the following issues (additional issues are also likely to be addressed during the course of this research work):

- Domain model, including constraints;
- Web-site structure, using CMS-oriented concepts (e.g., pages, page layouts);
- Navigation model and User-Interfaces model, using CMS-oriented concepts (e.g., tab, module), related to the web-site's structure;
- The web-application's behavior (by means of mechanisms similar to use-cases, interaction diagrams, and pseudo-code);
- CMS integration aspects (e.g., extensions provided by the modeled web-application).

This work is to be validated by means of a number of case-studies. One of those case-studies will be WebC-Docs [27], a Document Management System web-application that is based on the WebComfort CMS framework [11], [24].

We intend to achieve our goal by developing not one modeling language, but rather two such languages: CMS-IL (CMS Intermediate Language) and CMS-ML (CMS Modeling Language).

CMS-IL will be a relatively-low-level language that will nevertheless be independent of any CMS platform; because of this independence of specific CMS platforms, this language will likely be based on the CMS generic metamodel presented in [10]. It is important to note that the main objective of CMS-IL is to provide a "common ground" for the specification of CMS-based web-applications, and not to provide a way to produce web-application models in a simple and easy fashion. Depending on the capabilities of the target CMS platform, CMS-IL models are meant to be used as input to source-code generators or to a "CMS Model Interpreter" component located on the target CMS (this component will typically consist of a CMS module that will interpret the CMS-IL model).

On the other hand, CMS-ML will be a language at a higher level of abstraction than CMS-IL (it will provide a set of mnemonics that completely encapsulate the CMS-IL language). Unlike CMS-IL, the objective of CMS-ML is to produce web-application models in a way that is as simple and efficient as possible. The CMS-ML language will likely be inspired on already-existing languages such as WebML, UWE, XIS2, and UML [28].

Of course, the CMS-ML language would be irrelevant if there was no way to obtain CMS-IL models from CMS-ML models. Thus, this work also intends to produce a model-to-model transformation (not necessarily bidirectional, as such a requirement would likely not be relevant in practice [29]) that translates CMS-ML models to CMS-IL models with no information loss; we refer to this transformation as "ML2IL". Obviously, this presents the added requirement of CMS-IL being *at least* as expressive as CMS-ML.

Figure 1 provides a basic overview of the proposed approach, including the usage of the CMS-ML and CMS-IL modeling languages.

The rationale for these two languages is largely derived from our own previous experience in the development of the XIS2 language [22], where we learned that web-oriented modeling approaches (as well as modeling approaches in other domains) tend to concentrate too many details of too many abstraction levels in a single language, which makes the language either very complex and able to model most intended applications, or relatively simple and unable to model the more complex, "real-world" applications. We believe that one of the major contributions of this work will be

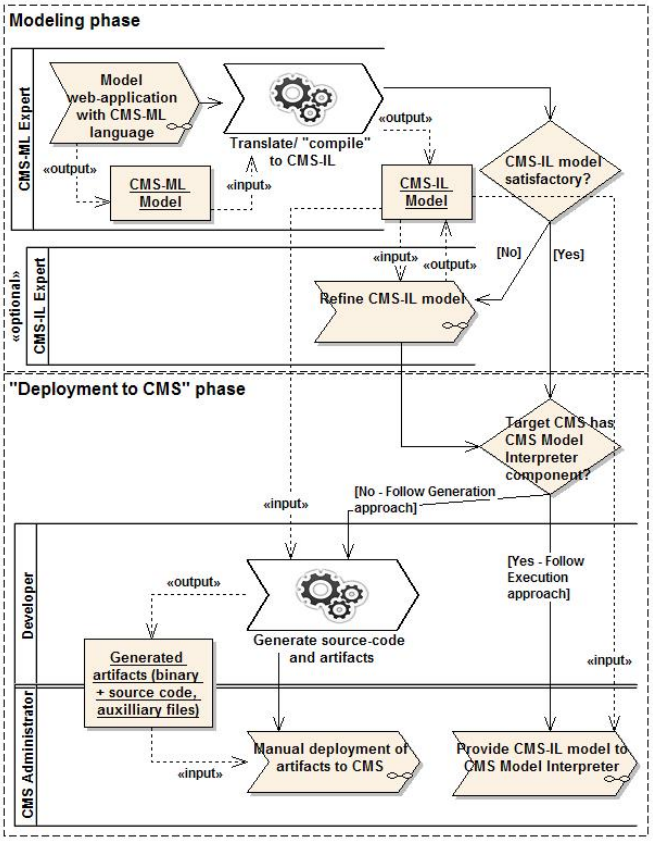

Figure 1. The proposed MDE-oriented approach.

the demonstration that this "separation of concerns" between two different languages can be of great help in defining a model-driven approach that can be used *in practice*.

A metaphor for the relationship between CMS-ML, CMS-IL, and the intended web-application itself, can be found in the Microsoft .NET Framework: (1) programs are written using a "high-level" source-code-based language such as C#; (2) the language's compiler converts the source-code to an intermediate language called MSIL; and (3) at runtime, the generated MSIL is converted into native binary code that is afterward executed. Considering this metaphor, CMS-ML would correspond to C#, CMS-IL would correspond to MSIL, and the native code would correspond to the intended web-application. It is important to note that this metaphor expresses only the relationship between these languages, and *not* their true level of abstraction (e.g., CMS-ML is not at C#'s level of abstraction, as C# is a low-level language). Another important item to highlight in this metaphor is that MSIL is not supposed to be modified by the developer (although it is possible to do it), while our approach does consider that CMS-IL can be modified.

Finally, one could argue that, if the CMS-ML language is sufficiently expressive, there will be no need for CMS-IL. Although that would be true, it is important to reiterate that CMS-IL's main objective is to provide a "common ground" for the specification of CMS-based web-applications; in turn, this common ground will be important for the definition of future CMS-oriented languages (which are not necessarily based on CMS-ML). Considering the "Microsoft .NET Framework" metaphor again, the importance of CMS-IL would be equivalent to the importance of MSIL in relation to languages such as C# or Visual Basic.NET.

IV. Current Work and Preliminary Results

We are currently defining the CMS-IL and CMS-ML languages. To this end, we are analyzing existing web-application modeling languages (e.g., WebML, UWE) and approaches, and determining what concepts and patterns are relevant to our work. The generic CMS metamodel defined in [10] also plays an important role in this process, as it is the first step towards a language that is independent of any specific CMS.

The CMS-IL language is to be defined first, as it is our lowest-level language. This language is supposed to exhibit a high degree of complexity, due to its main requirements: (1) ensure CMS independence; and (2) address all aspects necessary to specify a web-application of high complexity.

After the specification of CMS-IL is finalized, we can concretely define the CMS-ML language, as it will be a higher-level language (in the sense that it provides a set of mnemonics that "hides" the low-level CMS-IL language) with the objective of accelerating web-application development (namely by providing patterns that can be found in typical web-applications).

This work has already produced the following results: (1) the WebComfort CMS [24] has been evolving to become a practical web-application framework [11] (in fact, it is being used in some projects of our research group), and (2) one of our validation case-studies, WebC-Docs [27], has been developed by means of a traditional development approach.

The WebComfort framework has been evolving to support the deployment and execution of web-applications [11]; WebComfort is already being used for some medium-complexity web-applications such as "Portal eArte" [30], the WebSNARE project [31], and some instances of WebC-Docs (all of these examples are currently being used by "real" end-users). This is relevant to our work, because the WebComfort framework will be the basis for some of our validation case-studies.

Additionally, one of this work's validation case-studies, the WebC-Docs system [27], has been developed as a toolkit for the WebComfort framework, by means of a traditional development approach. WebC-Docs consists of a web-based Document Management System, with a medium-to-high degree of complexity, that provides flexible mechanisms for typical document management activities, such as indexing and search, metadata specification, and document storage. It is important to note that this manual development effort was

important to: (1) gain some insight regarding the typical set of patterns to be followed by CMS-based web-applications (e.g., the specification of "external" configuration modules, the definition of an API that can be used by third-party components or web-applications); and (2) determine the set of functionalities that are necessary to effectively support such relatively complex web-applications (e.g., workflow specification, user and role management).

V. Work Plan and Implications

As we have mentioned in the previous section, we are currently in the process of defining the CMS-IL and CMS-ML modeling languages. The "ML2IL" model-to-model transformation (that will be used to "compile" CMS-ML to CMS-IL) will also be defined in conjunction with the definition of the CMS-ML language itself, as this transformation will consist of translating a CMS-ML model to the corresponding CMS-IL model.

After the CMS-IL and CMS-ML languages are defined, we will model the WebC-Docs system (in its current state) using those languages. This effort will be the first validation of our work, and will be undertaken on a purely internal/personal scale. The objective of this effort will be to determine: (1) whether a relatively complex web-application such as WebC-Docs can be entirely specified using the CMS-IL and CMS-ML languages; and (2) the correctness of the ML2IL transformation (i.e., if the obtained CMS-IL model correctly reflects the web-application that is specified in the CMS-ML model).

We will also create a "CMS Model Interpreter" component as a set of WebComfort modules to allow the runtime interpretation of CMS-IL models. We believe that this component will allow us to simplify considerably the deployment of modeled web-applications, as this kind of deployment will typically just require the upload of a CMS-IL model to the target CMS instance.

After the first validation with the WebC-Docs system, we will also validate the CMS-ML and CMS-IL languages (and the development approach defined by this work) in additional case-studies, namely in the context of some of our research group's MSc projects as well as other academic projects (e.g., projects in curricular disciplines). At the same time, we will evolve the WebC-Docs system (according to business requirements that are identified in the meantime) by altering the previously-obtained models; this effort will be used to determine the validity of CMS-ML and CMS-IL for the development of web-applications addressing requirements other than those that were identified before these languages were defined.

Evaluation of the approach will focus on practical aspects such as: (1) whether the intended web-application can be specified entirely using the CMS-IL and CMS-ML languages; (2) whether CMS-ML really accelerates the modeling of a web-application; (3) which deployment alternative presents the most advantages (a CMS Model Interpreter, or the *a-priori* generation of CMS-specific artifacts), from a practical perspective; and (4) whether the development of web-applications really benefit from this approach, by considering issues like reduced development time or how many times a model needs to be changed to be able to address a given set of requirements.

If the opportunity arises, we also hope to address practical issues regarding the CMS Model Interpreter component, such as the possible advantage of this component generating (and compiling) CMS-specific artifacts when a CMS-IL model is provided, instead of the (potentially slow) model interpretation at runtime.

VI. Conclusion

The increasingly-popular MDE paradigm advocates the use of models as first-class citizens in the software development process, while artifacts such as documentation and source-code can be automatically produced from those models by means of model transformations. CMS systems have the potential for becoming the next wave of web-application frameworks; they can also benefit from the advantages provided by this paradigm, as MDE approaches can significantly accelerate the development and deployment of these web-applications and features, and simplify their maintenance.

This paper presented our proposal for a MDE-based approach for the development of web-applications based on CMS systems. This approach is based on two CMS-oriented languages, CMS-ML and CMS-IL, that are situated at different levels of abstraction. CMS-ML addresses modeling in a quick and simple fashion; on the other hand, CMS-IL uses low-level concepts (but not specific towards a particular CMS), but provides a common ground for the building of higher-level languages (such as CMS-ML). Developers can create web-applications using a high-level language (CMS-ML), "compile it" to a lower-level language (CMS-IL) by means of an automatic model-to-model transformation ("ML2IL"), and finally, either generate source-code or supply the created models as input to a "CMS Model Interpreter" component (that will handle the runtime execution of the modeled application).

References

[1] J. Robertson, "So, what is a content management system?" June 2003, Retrieved Tuesday 17th March, 2009 from http://www.steptwo.com.au/papers/kmc_what/index.html.

[2] P. Suh, D. Addey, D. Thiemecke, and J. Ellis, *Content Management Systems (Tools of the Trade)*. Glasshaus, October 2003.

[3] B. Boiko, *Content Management Bible*. Hoboken, New Jersey, U.S.A.: John Wiley & Sons, December 2001.

[4] CMSMatrix, "The CMS Matrix," Retrieved Tuesday 17th March, 2009 from http://www.cmsmatrix.org, 2009.

[5] OpenSourceCMS, "OpenSourceCMS," Retrieved Tuesday 17th March, 2009 from http://www.opensourcecms.com, 2009.

[6] U. Kampffmeyer, "ECM – Enterprise Content Management," 2006, Retrieved Tuesday 17th March, 2009 from http://www.project-consult.net/Files/ECM_WhitePaper_kff_2006.pdf.

[7] AIIM, "Association for Information and Image Management," Retrieved Tuesday 17th March, 2009 from http://www.aiim.org, 2009.

[8] A. Rockley, *Managing Enterprise Content: A Unified Content Strategy (VOICES)*. New Riders Press, October 2002.

[9] T. Jenkins, *Enterprise Content Management Technology: What You Need to Know*. Open Text Corporation, October 2004.

[10] J. L. V. d. Carmo, "Web Content Management Systems: Experiences and Evaluations with the WebComfort Framework," Master's thesis, Instituto Superior Técnico, Portugal, December 2006.

[11] J. d. S. Saraiva and A. R. d. Silva, "The WebComfort Framework: An Extensible Platform for the Development of Web Applications," in *Proceedings of the 34th EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO 2008)*, IEEE Computer Society, Ed., September 2008, pp. 19–26.

[12] D. C. Schmidt, "Guest Editor's Introduction: Model-Driven Engineering," *Computer*, vol. 39, no. 2, pp. 25–31, February 2006, Retrieved Wednesday 1st April, 2009 from http://doi.ieeecomputersociety.org/10.1109/MC.2006.58.

[13] Model transformation at Inria / Introduction to Model-Driven Engineering, Retrieved Wednesday 1st April, 2009 from http://modelware.inria.fr/article65.html.

[14] G. Rossi, O. Pastor, D. Schwabe, and L. Olsina, Eds., *Web Engineering: Modelling and Implementing Web Applications*. Springer-Verlag, 2008.

[15] WebML.org, Retrieved Wednesday 18th March, 2009 from http://www.webml.org.

[16] S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, and M. Matera, *Designing Data-Intensive Web Applications*. Morgan Kaufmann, 2003.

[17] N. Moreno, P. Fraternalli, and A. Vallecillo, "A UML 2.0 profile for WebML modeling," in *ICWE '06: Workshop proceedings of the sixth international conference on Web engineering*. New York, NY, USA: ACM, July 2006, Retrieved Wednesday 18th March, 2009 from http://doi.acm.org/10.1145/1149993.1149998.

[18] ——, "WebML modelling in UML," *IET Software*, vol. 1, no. 3, pp. 67–80, June 2007.

[19] UWE – UML-based Web Engineering, Retrieved Wednesday 18th March, 2009 from http://www.pst.ifi.lmu.de/projekte/uwe.

[20] N. Koch and A. Kraus, "The Expressive Power of UML-based Web Engineering," in *Proceedings of the Second International Workshop on Web-Oriented Software Technology (IWWOST'2002)*, June 2002, Retrieved Wednesday 18th March, 2009 from http://www.pst.informatik.uni-muenchen.de/personen/kochn/IWWOST02-koch-kraus.PDF.

[21] N. Koch, A. Kraus, and R. Hennicker, "The Authoring Process of the UML-based Web Engineering Approach," in *Proceedings of the First International Workshop on Web-Oriented Software Technology (IWWOST'2001)*, June 2001, Retrieved Wednesday 18th March, 2009 from http://www.dsic.upv.es/~west/iwwost01/files/contributions/NoraKoch/Uwe.pdf.

[22] A. R. d. Silva, J. d. S. Saraiva, R. Silva, and C. Martins, "XIS – UML Profile for eXtreme Modeling Interactive Systems," in *Fourth International Workshop on Model-based Methodologies for Pervasive and Embedded Software (MOMPES 2007)*. Los Alamitos, CA, USA: IEEE Computer Society, March 2007, pp. 55–66, Retrieved Thursday 26th March, 2009 from http://doi.ieeecomputersociety.org/10.1109/MOMPES.2007.19.

[23] Drupal CMS, Retrieved Wednesday 8th April, 2009 from http://drupal.org.

[24] WebComfortOrg, "WebComfort.org," Retrieved Monday 8th June, 2009 from http://www.webcomfort.org, 2009.

[25] L. Schou, "Creating a model-driven Web application framework," Master's thesis, Technical University of Denmark, Denmark, March 2008, Retrieved Wednesday 18th March, 2009 from http://www.imm.dtu.dk/English/Research/Software_Engineering/Publications.aspx?lg=showcommon&id=213586.

[26] Limestone Solutions – Content Management, Retrieved Tuesday 24th March, 2009 from http://www.limestoneweb.co.uk/Solutions/ContentManagement.

[27] WebC-Docs, "WebComfort.org – WebC-Docs," Retrieved Wednesday 25th March, 2009 from http://www.webcomfort.org/WebCDocs, 2009.

[28] OMG, "Object Management Group – UML," Retrieved Monday 13th April, 2009 from http://www.uml.org, 2009.

[29] T. Stahl and M. Voelter, *Model-Driven Software Development: Technology, Engineering, Management*. Hoboken, New Jersey, U.S.A.: John Wiley & Sons, May 2005.

[30] eArte - Portal de Arte e Cultura, Retrieved Thursday 16th April, 2009 from http://www.portal-earte.com.

[31] WebSnare, Retrieved Thursday 16th April, 2009 from http://snare.inesc-id.pt.