Model-driven web engineering methods: a literature review

Métodos de ingeniería web dirigidos por modelos: una revisión de literatura

Jesús Andrés Hincapié Londoño[*]*, John Freddy Duitama*

Grupo de Investigación Ingeniería y Software. Universidad de Antioquia. Ciudad Universitaria. Bloque 21-316. A. A. 1226. Medellin, Colombia

(Recibido el 1 de septiembre de 2010. Aceptado el 23 de mayo de 2012)

Abstract

This paper presents some of the model-driven Web engineering methods that have been proposed, and discusses and analyzes the advantages and disadvantages of such methods regarding current tendencies and best practices on model-driven engineering. The idea is to present each approach and analyze the models they propose to represent Web applications, the architectural aspects in the transformations, and the use of current Web user interface technologies in the generated code. This is done in order to depict possible research lines for future works on the model-driven Web engineering area.

--------- *Keywords:* Web engineering, model-driven software development, model-driven web engineering

Resumen

Este artículo presenta algunos de los métodos de ingeniería Web dirigida por modelos que se han propuesto. En él se discuten y analizan las ventajas y desventajas de dichos métodos con relación a las tendencias actuales y las mejores prácticas en la ingeniería dirigida por modelos. La idea es presentar cada método y analizar los modelos que propone para representar aplicaciones Web, los aspectos arquitectónicos en las transformaciones y el uso de tecnologías actuales de interfaz de usuario Web en el código generado. Esto se hace con el fin de vislumbrar posibles líneas de investigación para trabajos futuros en el área de la ingeniería Web dirigida por modelos.

---------- *Palabras clave:* Ingeniería web, desarrollo de software dirigido por modelos, ingeniería web dirigida por modelos

[*] Autor de correspondencia: teléfono: + 57 + 4 + 260 30 18 ó 57 + 4 + 300 02 00 ext. 130, correo electrónico: jahlon@gmail.com (J. Hincapié)

Introduction

Model-Driven Engineering (MDE) technologies appeared as a promising approach to address the inability of third-generation languages [1] to alleviate the complexity of platforms and express domain concepts effectively [2]. Model-Driven Software Development (MDSD), which provides a highly agile software development process, has as one of its main priorities the production of software that can be validated by end users and stakeholders as early as possible. It includes technologies such as Domain-Specific Languages (DSL), model transformations, and code generation that contribute to the goal of making software development a model-centric process [3] instead of just using models as documentation.

When developing a Web application, it is necessary to specify structure, behavior, navigation, and presentation aspects. Traditional engineering methods failed to specify navigational and presentation issues, which is why some people have proposed specific approaches to tailor these two aspects of Web applications. Such approaches are called Web Engineering Methods. In recent years, there have been several attempts to promote web application development as a model-centric process. These attempts encourage software developers to focus on problem domain-specific modeling and analysis and structured software reuse, meanwhile code generation is left to an automatic model transformation process.

This paper presents some of the most relevant model-driven Web engineering methods that have been proposed, and discusses and analyzes the advantages and disadvantages of such methods regarding current tendencies and best practices in Web application development.

The structure of this paper is as follows: first, a brief conceptual framework describing MDSD is presented. Then, some concepts about Web engineering are introduced. After that, some Web engineering methods and several model-driven web engineering approaches are described. Then, a set of analysis criteria are defined in order to guide the discussion about the reviewed methods. Finally, some conclusions and further research lines are presented.

Model-Driven Software Development

The application of models to software development has been a tradition for a long time, and it has become more popular since the development of the Unified Modeling Language (UML) [4].

MDSD presents an approach in which models do not only constitute documentation, but are considered to be similar to code artifacts because their implementation is automated. Since those models are highly coupled to the domain of applications, MDSD aims at finding domain-specific abstractions that can be specified through formal modeling, providing models that can be understood by domain experts.

MDSD is a software paradigm with roots in Software Product Lines (SPL) [5] engineering, which is the discipline of designing and building families of applications for a specific purpose or market segment [6]. In order to apply the domain-specific model concept, there are certain requirements that need to be taken into account: DSLs that allow expressing the models, transformation languages to express the model-to-code transformations, and code generators to obtain executable code on several platforms. MDSD's idea is give models a central role.

Web Engineering

Based on scientific and engineering principles, Web Engineering aims at establishing systematic approaches to successfully develop, deploy and maintain high quality Web-based systems. It also incorporates well-known software engineering principles and practices [7] from diverse areas such as human-computer interaction (HCI), system analysis and design, requirements engineering, hypermedia engineering, data structures, testing and project management, as well as social sciences, arts, and graphic design.

When model-based initiatives such as MDSD grew in popularity within the software development community, several Web Engineering approaches began to change their notations and processes to be MDSD compliant. As stated in [8], this change implied a redesign in Web modeling languages; a reorganization of the set of models to be built in a modular and platform independent way; planning the development processes in terms of model transformations; and adopting standards such as UML, Meta-Object Facility (MOF) [9], XML Metadata Interchange (XMI), or Query/View/Transformation (QVT).

MDSD compliance turned into a discipline within Web Engineering called Model-Driven Web Engineering (MDWE). MDWE adopts some of the techniques proposed in MDSD in order to generate Web applications: (1) the construction of meta-models and models in the Web applications domain; (2) the definition and implementation of model-to-model transformations and model-to-text transformations with the purpose of obtaining some parts of the entire implementation; and (3) the adaptation or development of CASE tools to support the creation and transformation of models and the generation of code. In this way, MDWE aims at bridging the gap between the high level design models and the low-level Web implementation code [10].

Figure 1 depicts a general scheme followed by most of the MDWE methods, which propose a structure model to represent the data, a navigation model describing pages and the way to navigate among them, and the presentation model defining the human-computer interaction (HCI) elements. A user model is used in some approaches. Then, after a model transformation is performed, different layers of a Web application are generated, normally presentation, business logic, and persistence layers.

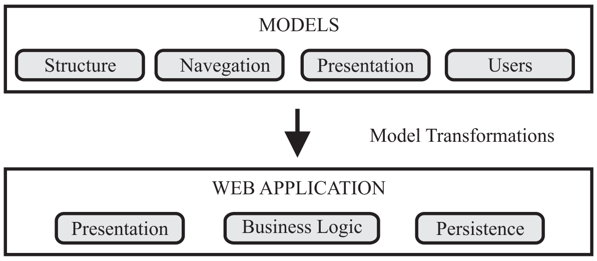

Figure 1 General scheme of a MDWE method

Literature review

Since most MDWE methods are based on traditional Web Engineering methods, we present first traditional methods, to proceed later to review MDWE approaches. At the end, we develop a discussion of each MDSD-based method presented based on a set of criteria previously selected.

Web engineering methods

Table 1 lists some of the most representative approaches of traditional Web engineering.

Table 1 List of Web engineering methods

#	*Name*
1	Object-Oriented Hypermedia Design Method (OOHDM) - 1995
2	Web Site Design Method (WSDM) - 1995
3	Web Modeling Language (WebML) - 2000
4	Object-Oriented Hypermedia (OO-H) - 2000
5	UML-Based Web Engineering (UWE) - 2002

The Object-Oriented Hypermedia Design Model (OOHDM) [11-14] is an approach that emphasizes separately the navigational design and the abstract interface design. As every object-oriented modeling proposal, it promotes the development of new applications reusing existing components. OOHDM is a four steps process. The first step is a domain analysis in which an application conceptual model is built. The second step consists of a navigational design describing the navigation structure of a hypermedia application in terms of navigational contexts. These contexts are inferred from navigation classes such as nodes, links, and guided tours. Nodes represent logic views of conceptual classes, and links are derived from conceptual relationships. The abstract interface design is the third step and proposes the construction of perceptible objects, such as an image or a map, in terms of interface classes. These perceptible objects are mapped to navigational objects in order to give the last ones a perceptible appearance. In the fourth step, interface objects are mapped to implementation objects involving architectural specifications.

Web Site Design Method (WSDM) is a user-centered approach where the starting point during the analysis is the set of potential Web application users. It consists of fourth phases: (1) user modeling, (2) conceptual design, (3) implementation design, and (4) the actual implementation.

Phase (1) classify users in function of the Web audience by looking at the organization or the process for which the Web site will be built. User classes are defined based on subsets of all potential users that are similar in terms of their information requirements. Next, user classes are analyzed focusing on the information requirements (what to present) and characteristics of intended audience. If users have different characteristics within one user class, this is divided into perspectives which represent different usability requirements. *Phase (2)* is projected to formally model the information requirements expressed in the user class descriptions by building conceptual models for the different user classes. During the sub-phase called navigational design, a conceptual navigation model -several navigation tracks- is constructed. A navigation track is described in terms of components and links that represent the way users of a particular perspective can navigate through the available information. Three types of components are described: An information component with the information that correspond to a specific perspective; a navigation component that consists of a group of links; and an external component which is a reference to a component in another site. In *phase (3)* the look and feel of the Web site is designed with the objective of creating a consistent, pleasing, and efficient look and feel for the conceptual design made in the previous phases. In order to achieve an implementation model, WSDM proposes some well-known guidelines like the ones described in [15]. *The final phase* is the actual realization of the Web site according to the design made in the previous phases. In order to store the information associated with the site, authors proposed using techniques for database backed web sites depicted in [16].

The Web Modeling Language (WebML) [17] is a notation to specify complex Web sites at conceptual level. It proposes four models: (1) a structural model defining the data of the application in terms of relevant entities and relationships, which is compatible with UML and Entity/Relationship (E/R) notations. (2) A hypertext model that describes hypertexts that can be published in the site. Each hypertext defines a site view and consists of a composition sub-model and a navigation sub-model. The first model specifies the pages that make up the hypertext and the content units that structure a page, and the second one describes how pages and content units are linked to form the hypertext. (3) A presentation model that states the distribution and graphical appearance of the pages, independently of the output device. And (4) a personalization model in which users and groups of users are modeled as predefined entities called *User* and *Group*.

The Object Oriented Hypermedia (OO-H) [18, 19] Method is a generic model, integrated into OO-Method [20, 21], for the semantic structure of Web interfaces. It defines the abstract interaction model of the user interface, the information which each type of user can access, and the navigation paths from one information view to another.

Since OO-Method captures the statics of the system, OO-H addresses particularities associated with the design of web interfaces by adding several constructs for navigation and interface design to the OO-Method conceptual model. OO-H provides an interface execution model in order to determine the way of implementing the conceptual model in a given development environment.

In OO-H, the navigation model is represented by means of a Navigation Access Diagram (NAD). The NAD is built starting from the filtering and enriching of the information provided by the class diagram that is captured in the conceptual modeling phase of the OO-Method. As each type of user has a different system view and can activate different services, each one needs a corresponding NAD.

The main components of the NAD are navigation classes, navigation targets, navigation links, and collections.

Navigation classes, which are based on the classes identified during the conceptual modeling phase, contain the attributes relevant to the considered user and view, and the services capable of being invoked by the actual user of the NAD. A navigation target is a set of navigation classes which provide the user with a coherent view of the system. OO-H bases its navigation targets on user requirements, instead of on the physical presentation of the information. Navigation links are defined by a name, a source navigation class, a target navigation class, associated navigation patterns, and associated navigation filters. Navigation patterns [22] are a mechanism for a web user interface to share its knowledge about the way of showing the information objects to the user. Navigation filters restrict the order, the quantity or the quality of the target objects. Collections are structures, with a set of filters and a set of navigation patterns associated, which abstract some concepts regarding both external and internal navigations, and are useful limiting the interaction options between the user and the application.

Regarding the execution model for a target development environment, OO-H focuses on defining how to implement the interface information associated to web environments, since OO-Method has already defined an execution strategy.

UML-based Web Engineering (UWE) [23, 24] is a development process for Web applications which focuses on systematic design, personalization, and semi-automatic generation. Based on UML and the UML extension mechanism, it defines navigation and presentation models which are supplemented by other UML diagrams and UML modeling elements within an iterative and incremental approach based on the Unified Software Development Process [25]. The main modeling activities in UWE are the requirements analysis, conceptual, navigation and presentation design, supplemented with task and deployment modeling and visualization of Web scenarios. The task models and state charts of Web scenarios are included to model the dynamic aspects of the application.

In UWE, requirements of a Web application can be specified by using a use case model. The static view of the system, also known as conceptual model, is represented using a UML class diagram which is built based on the use cases and the detailed description of these use cases with activity diagrams (in a textual form).

The navigation model is represented as UML stereotyped class diagrams and consists of two components: the navigation space model and the navigation structure model. The former specifies which object can be visited by navigation, while the latter defines how these objects are reached. The modeling of the navigation is built following a set of guidelines defined in [26].

The presentation model is represented using a particular form of a class diagram that uses the UML composition notation for classes and also stereotyped classes. This model describes where and how navigation objects will be presented to the user. For the presentation model, UWE uses a set of stereotypes that consists of the stereotypes *text*, *button*, *image*, *audio*, *anchor*, *collection*, and *anchored collection*.

In UWE, state chart diagrams are used in order to visualize navigation scenarios that allow detailing parts of the navigation structure model by specifying the event that triggers the transitions, defining guard conditions, and including the actions to be performed. UWE also proposes the use of sequence diagrams to show presentation flows such as interaction between windows and frames. UWE uses activity diagrams for task modeling; here, a task represents one or more actions that a user may perform to achieve a goal [27]. It also extends the concept of task by including actions performed by the system. With this extension, the method defined road-maps of user interaction with the system.

MDWE approaches

Table 2 presents several MDWE approaches. Most of the approaches described are well-recognized and widely referenced within the Web engineering research community. They represent well the topics of interest in Web engineering during the last years.

Table 2 List of MDWE approaches

#	*Name*
1	Web Software Architecture (WebSA) – 2004
2	Hypertext Modeling Method of MIDAS (MIDAS HM^3) – 2006
3	Object Oriented Web Solutions (OOWS) – 2006
4	UML-Based Web Engineering (UWE) – 2007
5	Web Modeling Language and WebRatio (WebML and WebRatio) - 2008
6	DSL for the implementation of dynamic web applications (WebDSL) - 2009
7	DSL for generating Web application (MarTE) – 2009

The approaches are described in terms of the models they proposed that are common to Web Engineering and its diagramming notation; the consideration of architectural models; and model transformations

The Web Software Architecture (WebSA) [28, 29] is a model-driven approach that defines an instance of the MDA development process for the Web application domain. It groups the Web application model into three viewpoints: requirements, functional, and architectural viewpoints.

Regarding common models to Web engineering, WebSA uses models proposed in two approaches: UWE and OO-H. These models correspond to the requirements and functional viewpoints and consist of a structural model and a navigational model. The structural model is built using a UML class diagram, while the navigational model is built using a UML class diagram and a UML profile.

The architectural viewpoint, the main contribution of the approach, includes a logical architecture view and a physical architecture view. It is made up of three models: (1) the subsystem model, which determines the layers of the application, (2) the Web component configuration model, which represents each subsystem in terms of abstract components and abstract connectors, and (3) the web component integration model that allows the designer to determine the low level platform-independent component that make up the final application.

The MDA-based development process establishes four phases of the development life cycle: analysis; platform independent design, where a platform independent (PIM) model is built; platform specific design, where a platform specific model (PSM) is built; and implementation.

In the analysis phase, the Web application specification is divided into functional models and conceptual architecture models. The first ones reflect the functional analysis, and the second ones define the system architecture based on the concept of conceptual architecture [30].

In order to get to the platform independent design phase, a PIM-to-PIM transformation is performed providing a set of artifacts in which the conceptual elements of the analysis phase are mapped to concrete elements where the information about functionality and architecture is integrated. These models of the second phase are then transformed into Platform Specific Models (PSM) by means of several PIM-to-PSM transformations that generate the specification of the Web application for a given platform. In the final phase, a PSM-to-Code transformation, implemented by means of templates, is performed.

The Hypertext Modeling Method of MIDAS (HM^3) [31] is a methodological framework for agile development of Web information systems based on MDA. It proposes to model the system by specifying Computation Independent Models (CIMs), PIMs, PSMs, and the mapping rules between these models. It proposes to model the system according to three basic aspects: hypertext, content, and behavior. However, it does not propose any strategy for modeling architectural issues. All the models in MIDAS are made using UML as notation, as well as the use of UML profiles.

HM^3 defines a new UML profile to support the Hypertext modeling, and it uses this profile to specify the meta-models for the user services model, the extended user services model, the extended slices model, and the extended navigation model it proposes. Besides, the approach defines the transformation rules in a declarative style and then maps them to graph rules with the intention of automating these rules with existing facilities to automate graph transformations.

The Object-Oriented Web Solutions (OOWS) [32, 33] is based in OO-Method, which is a method that combines formal specifications with conventional object modeling techniques to specify information systems. OOWS integrates navigation and presentation designs into the object-oriented conceptual modeling provided by OO-Method. OOWS allows specifying functional, navigational, and presentational aspects of Web application requirements by using graphics schemes and high abstraction level primitives. Using conceptual schemes as input, a methodology is defined in order to bring the problem space to the solution space by defining matches between conceptual modeling abstractions and final software components.

The structural, dynamic, and functional models come from OO-Method; OOWS complements them with a navigational model and a presentation model. The structural model is defined by a class diagram. The dynamic model, a state charts diagrams, describes the different valid sequences of an object life-cycle for each system class, and it also represents interaction between objects by means of sequence diagrams. The functional model captures the semantics of state changes in order to define the effects of a service using a formal specification. Before the creation of the navigation and presentation models, OOWS defines a user diagram to describe the types of users that can interact with the system and the visibility they can have on the attributes and operations of the classes. Once users are identified, a system structured view is created for each class of the structure diagram in terms of attributes, operations, and relationships visibility, which forms the navigation diagram. The presentation model consists of several patterns associated to the primitives of navigational context (navigation classes, links): information paging, order criteria, and information organization. The last one is made of four patterns: record, table, master-detail, and tree.

In order to develop the application, OOWS takes as a basis the OO-Method structural and behavioral models and generates the persistence and application layers by using the OlivaNova model transformation engine [34]. The presentation layer is generated by an OOWS transformation process, and all the artifacts are generated to be deployed on .NET platforms.

As stated earlier, MDA approach of UWE proposes several models to represents structure, navigation, and presentation. All this models are based in several UML diagram –such as classes, state charts, and sequence diagrams– and the use of UML profiles. Nevertheless, it does not consider any architectural model.

In this model-driven version of UWE [35], the content and presentation models are translated into Java beans and Java Server Pages (JSPs) using model transformation rules implemented in the Atlas Transformation Language (ATL). In order to make models executable, UWE proposes a virtual machine built on top of the controller of the Spring Framework. This virtual machine executes the business processes integrated into the navigation structure. A detailed description of CIMs, PIMs, CIM-to-PIM transformations, and PIM-to-PIM transformations expressed as ATL transformation rules is included in the UWE extension presented in [36].

The WebML [37] characteristics regarding the models proposed by the method were presented earlier. However, in this new version of WebML, the language has been extended to include new constructs that allow specifying applications where page content and navigation can be adapted to build context-aware Web applications [38]. Next, we describe the commercial tool that assists the development process with WebML and its model-driven features.

WebRatio presents a design layer that allows data and hypertext design, producing an internal representation in XML of the models. It also allows creating XSL style sheets from XHTML mock-ups, which are prototypes of the presentation layer based on XHTML. These style sheets are associated with WebML pages in order to define a presentation style of the application.

A code generator, which connects the design layer with a J2EE-based runtime layer, exploits XSL transformations to translate models represented as XML into application code. The XSL translators produce a set of dynamic page templates and XML files that express the dependencies of a WebML unit from the data layer. These XML files are called unit descriptors.

In this new version of WebML, the language has been extended to include new constructs that allow specifying application where Web services can be invoke, the navigation can be driven by process models, and page content and navigation can be adapted to build context-aware Web applications [38].

The next work is a DSL for the implementation of dynamic web application called WebDSL [39, 40]. It consists of sub-languages for the specification of data models and for the definition of pages for viewing and editing objects in the data model. WebDSL uses entity definitions syntax in order to describe the data model of a Web application. It also proposes textual constructs for page definitions specifying a presentation of a Web page and its associated entities. The navigation between pages is defined by means of navigational elements that specify linked pages.

WebDSL also proposes higher abstraction level constructs for access control and workflow. The access control is governed by rules that determine access to the application components. It also allows representing users in order to generate authentication components. The workflow abstraction, based on WebWorkFlow [41], defines activities between different actors which result in task pages, task lists, status pages, and navigation between them.

The model transformations in WebDSL are implemented using Stratego/XL which is a rewriting system that integrates model-to-model, model-to-code, and code-to-code transformations. The WebDSL generator consists of a set of rules that rewrite extensions of the WebDSL core language to more primitive language constructs by means of a technique of compilation by normalization [42].

MarTE [43] uses a DSL to generate web application from UML domain models [44]. It describes the language's semantics, abstract syntax, and concrete syntax and frames it within a MDSD based transformation tool to generate web applications. The semantics of the DSL is referred as the meaning of web application elements that allow providing a well fit human-computer interaction [45] to generated applications. They describe concepts like web forms, web list, master-detail, lookup, defined selection, and primary key, claiming that with these artifacts it is possible to generate fully functional web applications that perform data manipulation operations (Create, Retrieve, Update, Delete, Exists and List) and that are ready for deployment.

The abstract syntax is supported by a meta-model called Web Application Meta-Model (WAMM). It describes all the global components needed to generate a complete web application in an object oriented programming language. Authors aim at WAMM as a generic web application platform, which any web application could be defined in. WAMM is divided in two parts: a structure part that contains the structure of the domain objects and the relationships between them; and an application part containing the relations between the domain objects and the web user interface.

The concrete syntax consists of a UML profile called WebApp Profile. It offers a mechanism to mark the UML domain model in order to provide a good human-computer interaction to the generated application. Such profile is made of several stereotypes and tagged values that are use to mark classes, attributes, and relationships in the domain model. These stereotypes are: Form, List, Master-Detail, Lookup, Defined Selection, and Primary Key. Each stereotype contains several tagged values which define specific characteristics of the user interface elements derived from the stereotypes through a transformation process. As an example, the Form stereotype, which applies to classes, is used when there is the need of manipulating in a web form the information of a single record based on the marked class. The transformation process for generating web applications is based on ATL and JET, and uses some technologies in order to integrate the DSL into the Eclipse Platform [46].

Analysis criteria

Table 3 presents a list with the criteria that will be used to analyze, MDWE approaches presented previously.

Table 3 List of analysis criteria

#	*Criterion*
1	Definition of common models for Web engineering methods
2	Ease of use of the diagramming notation
3	Independence of the target architecture from the transformations
4	Use of current Web interface technologies in the generated presentation layers.

The idea behind these criteria is to determine if the methods described in the previous section include elements of current research in MDWE. Also, it is important to analyze the ease of use when the methods serve as a basis for new MDWE developments. The first criterion refers to the discussion of whether the methods proposed or not some of the models depicted in figure 1, and how those models are implemented and coupled to the whole process. The criterion number two takes to analyze the graphics constructs that each method proposes. This implies discussing about how easy it is for a developer to understand and use the notation, and how well it represents the characteristics of Web applications. The third criterion refers to determine if the target architecture of the generated Web application is included in the method transformation engine, or if there is a mechanism to detach the target architecture from the method, opening the opportunity to generate Web applications for different architectures. Finally, the last criterion is intended to determine if the generated Web applications are based on technologies that are used nowadays to build Web applications, or if the concepts of Rich Internet Applications (RIA) [47] are taken into account in the generation process.

RIA approaches bring all the benefits of desktop applications to increase the responsiveness and usability of the user interfaces in the generated Web application. This is achieved by allowing the use of client's memory, powering both the client and the server to carry out complex operations, and improving the presentation and user interaction by avoiding unnecessary refreshments of the whole page.

Discussion

Here, an analysis of the approaches described previously is presented, regarding the criteria listed in table 3.

All the approaches coincide on defining a structure model, a navigation model, and a presentation model. However, there are some special cases in which extra models are defined, or in which the common models are defined implicitly. It is the case of MarTE, where the navigation and presentation characteristics of a Web application are represented through tagged values in the UML profile used to mark the structural model. Although this feature of MarTE may be simple and easy for the user to use, it restraints the user to the navigation and presentation characteristics defined by the authors.

The use of common models in different approaches facilitates the understanding of Web application designs, and creates a set of standard concepts for Web application modeling. Furthermore, it could lead to define Web application models serialization standards that allow interchanging models between different tools, as it is done nowadays with standards like XMI.

All approaches, except WebML and WebDSL, use UML along with UML profiles to define the modeling notation. This is very convenient since UML is a widely accepted standard for modeling software, and learning a UML-based Web modeling language, for a developer used to UML, may be easier than learning a totally unknown language. However, there are notations with graphic constructs that represent better the elements of the Web application domain. It is the case of WebML which has a proprietary notation (DSL) to define each model it proposes. The notation of WebML is easy to understand since its abstractions are very similar to actual objects of the Web domain, thus it provides a more intuitive way for developers to model Web applications. This type of notations may demand more learning time than a known notation like UML, but it compensates this time with its intuitiveness. Moreover, using technologies such as Eclipse Modeling Framework (EMF) or Graphic Modeling Framework (GMF), it is relatively easy to build tools that support modeling Web application with these notations.

The case of WebDSL is different since the modeling notation is not based on commonly software development notations. Besides, it lacks of the intuitiveness of the graphic constructs because of its textual form. This fact makes the process of learning and using this notation harder than using a graphic one whether it is UML-based or no.

Most of the approaches do not consider a mechanism to keep the specification of the target architecture separate from the transformations. In most of them, authors decide first the architecture of the generated application and the platform in which the application will be executed, and then they define the method transformations according to the decision they made. This leads the target architecture to depend on the method, and restricts the use of the method because, for instance, if the method generates application for the J2EE platform, it could not be used in a project that requires to be implemented for a .NET platform.

In the case of WebSA, authors claim that their approach can provide a way for transformation and platform modularization to support different architectures. They define two transformations: (1) a transformation in which functional models and architectural models are merged, and (2) a transformation that converts the integration model to different platform specific models. The second transformation uses the OMG standard MOFScript [48], and defines the rules to transform a PIM to code corresponding to J2EE and .NET platforms.

Despite WebSA already predefines different transformations for specific platforms, it is a very promising approach since there are similarities in these transformations that could be common to any platform. In this way, a generic mechanism could be defined in order to include more transformation rules that allow the generation of Web application to other platforms like PHP.

Finally, the definition of an architectural model could allow the integration of this approach with different Web engineering methods.

On the topic of the last criterion, most of the methods do not present any evidence of the use of RIA technologies within the transformations that generate the presentation layers. At most, they support presentation patterns catalogs for Web applications, or XML techniques for the implementation of the navigation and presentation aspects. These patterns or techniques lack of current presentation characteristics of the so-called Web 2.0 applications such as the use of AJAX or Abode AIR technologies. Including this type of technologies within the Web engineering methods could improve and make more attractive to final users the generated applications. The only method of the ones described in this paper, which presents works regarding the use of current Web interface technologies, is WebML. It proposes an approach [49] to apply the RUX-Method [50] presentation model to obtain a RIA.

The downside of using the RUX-Method is that it implies an increase in the complexity of the transformations of the method since it has to take into account the constructs for the new RIA-based presentation model. Nevertheless, it is something that can be done, and whose benefits outweigh the disadvantages.

Conclusions

In this paper we presented a literature review of Web Engineering methods. First, some traditional non model-driven methods were introduced to establish the origins of Web engineering. Then, several model-driven approaches were portrayed and analyzed regarding the definition of common models in Web engineering, the ease of use of the diagramming notation, the independence of the target architecture from the transformations, and the use of current Web interface technologies in the generated application.

One conclusion that comes out of the analysis performed is that the use of common models is recommended since it promotes standard concepts in Web application modeling. Furthermore, the notation for building such models should be intuitive and easy to use, and it should have a serialization mechanism that allows integrating the models with other methods.

Another conclusion is that the use of current Web interface technologies in the generated applications and the separation of the target architecture from the transformations are not common among the methods analyzed. This leads to possible research lines in which mechanisms to represent architectures are studied in order to find a way to separate that representation from the transformations of a MDSD-based Web engineering method. It also suggest research lines that consider approaches to model RIAs within a Web engineering method in order to generate application that take advantage of modern Web technologies.

Acknowledgments

This work was developed within the *ALIANZA REGIONAL EN TIC APLICADAS (ARTICA)*, funded by Colciencias and the Ministry of Information and Communication Technologies.

References

1. M. Nasir. *A Journey Through Programming Language Generations*. Disponible en: http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol2/mjbn/article2.html. Consultado en Octubre 5 de 2009

2. D. Schmidt. "Guest Editor's Introduction: Model-Driven Engineering." *Computer.* Vol. 39. 2006. pp. 25-31.

3. A. Forward, T. Lethbridge. *Problems and opportunities for model-centric versus code-centric software development: a survey of software professionals*. Proceedings of the 2008 international workshop on Models in software engineering. ACM, Leipzig (Germany). 2008. pp. 27-32.

4. OMG. *OMG Unified Modeling Language (UML), Superstructure*. 2009. Disponible en: http://www.omg.org/spec/UML/2.2/Superstructure/PDF/. Consultado en Octubre 5 de 2009.

5. SEI. *Software Product Lines | Overview*. Disponible en: http://www.sei.cmu.edu/productlines/. Consutado en Mayo 20 de 2010.

6. T. Stahl, M. Voelter. *Model-Driven Software Development: Technology, Engineering, Management*. 1st ed. Ed. Wiley. 2006. pp. 20-35.

7. S. Murugesan, Y. Deshpande, S. Hansen, A. Ginige. “Web Engineering: a New Discipline for Development of Web-Based Systems.” *Web Engineering*. Vol. 2016. 2001. pp. 3-13.

8. N. Koch, S. Meliá-Beigbeder, N. Moreno-Vergara, V. Pelechano-Ferragud, F. Sánchez-Figueroa, J. Vara-Mesa. “Model-driven web engineering.” *Upgrade-Novática Journal (English and Spanish), Council of European Professional Informatics Societies (CEPIS) IX*. Vol. 2. 2008. pp. 40-45.

9. OMG. *Meta Object Facility (MOF) Core Specification*. 2006. Disponible en: http://www.omg.org/spec/MOF/2.0/. Consultado en Octubre 5 de 2009.

10. H. Gellersen, M. Gaedke. “Object-Oriented Web Application Development.” *IEEE Internet Computing*. Vol. 3. 1999. pp. 60-68.

11. D. Schwabe, G. Rossi. “The object-oriented hypermedia design model.” *Commun. ACM*. Vol. 38. 1995. pp. 45-46.

12. D. Schwabe, G. Rossi. *Building hypermedia applications as navigational views of information models*. Hawaii International Conference on System Sciences. IEEE Computer Society. Los Alamitos, Californis (USA). 1995. pp. 231.

13. D. Schwabe, R. Guimarães, G. Rossi. “Cohesive Design of Personalized Web Applications.” *IEEE Internet Computing*. Vol. 6. 2002. pp. 34-43.

14. D. Schwabe, G. Rossi. “An object oriented approach to Web-based applications design.” *Theory and practice of object systems*. Vol. 4. 1998. pp. 207-225.

15. J. December, M. Ginsburg. *Html and Cgi Unleashed/Book and Cd-Rom*. 1st ed. Ed. Pearson Education Ltd. 1995. pp.194-258.

16. P. Greenspun. *Database Backed Web Sites: The Thinking Person’s Guide to Web Publishing*. 1st ed. Ed. Ziff-Davis Press. 1997. pp.214-251.

17. S. Ceri, P. Fraternali, A. Bongio. “Web Modeling Language (WebML): a modeling language for designing Web sites.” *Computer Networks*. Vol. 33. 2000. pp. 137-157.

18. J. Gómez, C. Cachero, O. Pastor. *Extending a Conceptual Modelling Approach to Web Application Design*. Proceedings of the 12th International Conference on Advanced Information Systems Engineering. Ed. Springer-Verlag. London, UK. 2000. pp. 79-93.

19. C. Cachero, J. Gómez, *Advanced conceptual modeling of Web applications: Embedding operation interfaces in navigation design*. 21th International Conference on Conceptual Modeling (JISBD). Madrid (Spain). 2002. pp. 235-248.

20. O. Pastor, E. Insfran, V. Pelechano, J. Romero, J. Merseguer. *De Sistemes Informàtics. OO-METHOD: An OO Software Production Environment Combining Conventional and Formal Methods*. IN CAISE ’97. International conference on advanced information systems. Barcelona (Spain). 1997. pp. 145-158.

21. O. Pastor, J. Gómez, E. Insfrán, V. Pelechano. “The OO-Method approach for information systems modeling: from object-oriented conceptual modeling to automated programming.” *Inf. Syst*. Vol. 26. 2001. pp. 507-534.

22. M. Bernstein. *Patterns of hypertext*. Proceedings of the ninth ACM conference on Hypertext and hypermedia: links, objects, time and space---structure in hypermedia systems: links, objects, time and space---structure in hypermedia systems. ACM, Pittsburgh (USA). 1998. pp. 21-29.

23. N. Koch, M. Wirsing. *Software engineering for adaptive hypermedia applications*. PhD. Thesis. Reihe Softwaretechnik 12. 2001. pp. 145-289.

24. N. Koch, A. Kraus. *The expressive power of uml-based web engineering*. Second International Workshop on Web-oriented Software Technology (IWWOST02). Málaga (Spain). 2002. pp. 105-120.

25. I. Jacobson, G. Booch, J. Rumbaugh. *The Unified Software Development Process*. 1st ed. Ed. Addison-Wesley Professional. 1999. pp. 1-512.

26. R. Hennicker, N. Koch. *A UML-based methodology for hypermedia design*. Proceedings of the 3rd international conference on The unified modeling language: advancing the standard., Ed. Springer-Verlag. York (UK). 2000. pp. 410-424.

27. M. Harmelen. “Interactive system design using Oo&hci methods” *Object modeling and user interface design: designing interactive systems*. Ed. Addison-Wesley Longman Publishing Co. Inc. 2001. pp. 365-427.

28. S. Beigbeder, C. Castro. “An MDA Approach for the Development of Web Applications.” *Web Engineering*. Vol. 3140. 2004. pp. 769.

29. S. Beigbeder. *WebSA: un método de desarrollo dirigido por modelos de arquitectura para aplicaciones web.* PhD. Thesis. Universidad de Alicante. Departamento de Lenguajes y Sistemas Informáticos. Alicante (España). 2007. pp. 85-223.

30. P. Nowack. *Structures And Interactions - Characterizing Object-Oriented Software Architecture*. PhD Thesis. The Maersk Mc-Kinney Moeller Institute for Production Technology. University of Southern Denmark. Odense, Denmark. 2000. pp. 41-45.

31. P. Cáceres, V. Castro, J. Vara, E. Marcos. *Model transformations for hypertext modeling on web information systems*. Proceedings of the 2006 ACM symposium on Applied computing. ACM. Dijon (France). 2006. pp. 1232-1239.

32. O. Pastor, J. Fons, V. Pelechano, S. Abrahão. "Conceptual Modelling of Web Applications: The OOWS Approach." *Web Engineering*. Vol. 4143. 2006. pp. 277-302.

33. F. Valverde, P. Valderas, J. Fons, O. Pastor. *A MDA-Based Environment for Web Applications Development: From Conceptual Models to Code*. 6th International Workshop on Web-Oriented Software Technologies. Bucharest (Romania). 2007. pp. 164-178

34. Care Technologies. *CARE Technologies. OlivaNova Model Transformation Engines*. Disponible en: http://www.care-t.com/index.asp. Consultado en Abril 13 de 2010.

35. A. Kraus, A. Knapp, N. Koch. *Model-driven generation of web applications in UWE*. Proceedings of the International Workshop on Model-Driven Web Engineering. Como (Italy). 2007. pp. 23-38.

36. A. Kraus. *Model Driven Software Engineering for Web Applications*. PhD. Thesis. Ludwig-Maximilians-Universität München. 2007. pp. 73-114.

37. M. Brambilla, S. Comai, P. Fraternali, M. Matera. "Designing Web Applications with Webml and Webratio." *Web Engineering: Modelling and Implementing Web Applications*. Vol. 4823. 2008. pp. 221-261.

38. S. Ceri, F. Daniel, M. Matera, F. Facca. "Model-driven development of context-aware Web applications." *ACM Trans. Internet Technol*. Vol. 7. 2007. pp. 30-63.

39. E. Visser. "WebDSL: A Case Study in Domain-Specific Language Engineering." *Generative and Transformational Techniques in Software Engineering II*. Vol. 5235. 2008. pp. 291-373.

40. D. Groenewegen, Z. Hemel, L. Kats, E. Visser. *WebDSL: a domain-specific language for dynamic web applications*. Companion to the 23rd ACM SIGPLAN conference on Object-oriented programming systems languages and applications. ACM, Nashville. TN (USA). 2008. pp. 779-780.

41. Z. Hemel, R. Verhaaf, E. Visser. "WebWorkFlow: An Object-Oriented Workflow Modeling Language for Web Applications." *Model Driven Engineering Languages and Systems*. Vol. LNCS 5301. 2009. pp. 113-127.

42. L. Kats, M. Bravenboer, E. Visser. *Mixing source and bytecode: a case for compilation by normalization.* Proceedings of the 23rd ACM SIGPLAN conference on Object-oriented programming systems languages and applications. ACM, Nashville. TN (USA). 2008. pp. 91-108.

43. J. Cadavid, D. Lopez, J. Hincapié, J. Quintero. *A Domain Specific Language to Generate Web Applications*. Memorias de la XII Conferencia Iberoamericana de Software Engineering (CIbSE 2009). Medellin (Colombia). 2009. pp. 139-144.

44. B. Selic. *A Systematic Approach to Domain-Specific Language Design Using UML.* 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'07). Santorini island (Greece). 2007. pp. 2-9.

45. P. Molina. *Especificación de interfaz de usuario: De los requisitos a la generación automática*. PhD. Thesis. Universidad Politécnica de Valencia. Valencia (España). 2003. pp.113-225

46. Eclipse. *Eclipse Foundation: Eclipse*. Disponible en: http://www.eclipse.org/. Consultado el Octubre 5 de 2009.

47. P. Fraternali, G. Rossi, F. Sánchez. "Rich Internet Applications." *IEEE Internet Computing*. Vol. 14. 2010. pp. 9-12.

48. OMG. *MOF Model to Text Transformation Language 1.0. Jan*. 2008. Disponible en: http://www.omg.org/spec/MOFM2T/1.0/. Consultado en Junio 24 de 2010.

49. M. Brambilla, J. Preciado, M. Linaje, F. Sanchez. *Business Process-Based Conceptual Design of Rich Internet Applications.* Web Engineering, International Conference on. IEEE Computer Society. Los Alamitos, California (USA). 2008. pp. 155-161.

50. M. Linaje, J. Preciado, F. Sánchez. "Engineering Rich Internet Application User Interfaces over Legacy Web Models." *IEEE Internet Computing.* Vol. 11. 2007. pp. 5 3-59.