International Journal of Innovative Computing, Information and Control
Volume **9**, Number **1**, January **2013**

ICIC International ©2013 ISSN 1349-4198
pp. **413–436**

AN ANALYSIS OF MODEL-DRIVEN WEB ENGINEERING METHODOLOGIES

Gustavo Aragón[1], Maria-Jose Escalona[1], Michael Lang[2] and Jose R. Hilera[3]

[1]IWT2 Group
University of Sevilla
Seville 41004, Spain
gustavo.aragon@iwt2.org; mjescalona@us.es

[2]School of Physics
National University of Ireland, Galway
Galway, Ireland
michael.lang@nuigalway.ie

[3]Computer Science Department
University of Alcalá
Alcalá de Henares 28801, Spain
jose.hilera@uah.es

Received November 2011; revised March 2012

Abstract. *In the late 1990's, there was substantial activity within the "Web engineering" research community and a multitude of new Web approaches were proposed. However, numerous studies have revealed major gaps in these approaches, including coverage and interoperability. In order to address these gaps, the Model-Driven Engineering (MDE) paradigm offers a new approach which has been demonstrated to achieve good results within applied research environments. This paper presents an analysis of a selection of Web development methodologies that are using the MDE paradigm in their development process and assesses whether MDE can provide an effective solution to address the aforementioned problems. This paper presents a critical review of previous studies of classical Web methodologies and makes a case for the potential of the MDWE paradigm as a means of addressing long-standing problems of Web development, for both research and enterprise. A selection of the main MDWE development approaches are analyzed and compared in accordance with criteria derived from the literature. The paper concludes that this new trend opens an interesting new way to develop Web systems within practical projects and argues that some classical gaps can be improved with MDWE.*
Keywords: Model-Driven Web Engineering, Web engineering, Web methodologies

1. **Introduction.** In the early 1990's, the research community started to work in a new area of software engineering oriented towards the special characteristics of the Web environment. Several approaches, such as HDM (Hypermedia Design Model) [1] and OOHDM (Object-Oriented Hypermedia Design Method) [2], offered new techniques, models and notations which initially dealt with "hypermedia" systems in general and later focused on Web-based systems in particular. The evolution of this line of research, Web engineering [3], is analyzed in several comparative studies and surveys [4-7]. Despite some doubts being expressed about the necessity for specialized Web development methods as opposed to "traditional" or "conventional" methods [8], Web engineering has since become an established branch of software engineering. However, these aforementioned studies and surveys also indicate a number of major gaps in the Web engineering body of knowledge, as later

summarized in Subsection 2.2. The motivation for this paper is therefore to analyze how the emerging paradigm of Model-Driven Web Engineering (MDWE) is being applied in order to address some of these gaps. The objectives of this paper are

1. To review the literature on the emerging MDWE paradigm and discuss how it may potentially address long-standing problems in Web engineering.
2. To explore how the MDWE paradigm is being applied in existing and emerging Web development approaches.

Our research approach was guided in the first instance by the general principles laid down in Brereton et al. [9] and Kitchenham et al. [10,11] for conducting systematic literature reviews, and the discursive aspect of our work was further informed by surveying the opinions of a number of international experts in the area of MDWE.

Thus, our paper offers a vision completely focused on MDWE and its application, which, as it is presented in this paper, is truly significant and novel contribution.

This study is structured as follows. We start in Section 2 by presenting an overview of a selection of the best-known Web development methods and discussing how Model-Driven Engineering (MDE) can potentially serve both, to rationalize and integrate these methods and also to solve several important gaps detected in Web engineering. In Section 3, we look specifically at a number of new and emerging Web development methodologies that are based on the MDWE paradigm. Section 4 then presents an analysis of these new MDWE methodologies, broken down into five different aspects: metamodel complexity, concepts, transformations, standards and compatibility, and tools and industry experiences. The paper then finishes with a brief overview of other related work in Section 5, and concludes by stating our views on current issues and problems in the field of MDWE, as well as outlining possible directions for future work in Section 6.

2. **Background to the Current State-of-Practice.**

2.1. **Web engineering: an overview of development methods.**

Over the past decade, several methods, approaches and techniques have been proposed in the academic and professional literature in order to deal with special aspects of Web development. Navigation, complex interfaces, difficult maintenance, security aspects and unknown remote users are amongst the critical challenges relevant to Web-based system development [12]. In an appendix to their study of the use of Web development methods in practice, Lang and Fitzgerald [13] present a comprehensive list of over fifty methods and approaches for Web/hypermedia systems development. A description and comparative analysis of the better known of these Web development approaches can be obtained in [6], from which is derived the chronological "map of the territory" shown in Figure 1. Although some of these approaches, such as HDM (Hypermedia Design Method), are no longer in use, they nevertheless continue to be relevant to the Web development community because of the underlying concepts and principles upon which they are based. A number of the early approaches such as HDM [1] and RMM (Relationship Management Methodology) [14], were based on Entity Relationship Modeling, but all of the subsequent methodologies included in Figure 1 are object-oriented.

An important departure was the influential publication of OOHDM (Object-Oriented Hypermedia Design Method) [15]. This methodology is based on both HDM and the object-oriented paradigm and offers a systematic approach for the design and implementation of hypermedia systems. The valuable contribution of OOHDM to the field of Web engineering research is generally acknowledged and many of its ideas have since become widely accepted. OOHDM proposed dividing hypermedia design into several models, each

of which represented a critical aspect of hypermedia systems: a conceptual model, a navigational model and an abstract interface model. This notion of separating the different aspects of hypermedia systems was novel at the time, but it is now followed by the Web engineering research community, enabling the complexity of a system to be broken down into separate layers. In OOHDM, a change in the navigational model affects only the navigational model, and the conceptual model needs no changes. Another important idea of OOHDM was to use class diagrams to model not only the conceptual model but also the navigational model via an extension of the basic class diagram. Additional aspects which could not be easily or fully explained using class diagrams, such as navigational context or abstract interface diagrams, could be modeled using a supplementary notation proposed by OOHDM.

Following OOHDM, more approaches were put forward, each of which offered new ideas, models, processes and techniques suited to the specific needs of interactive hypermedia systems and for the Web environment. Gradually, hypermedia systems evolved into fully-fledged Web-based information systems and these approaches were also modified to meet this new challenge; for example, HDM (Hypermedia Design Method) developed into HDM2, which later mutated into HDM2000/W2000 and eventually led towards WebML.

It is clearly evident, even from just a cursory glance at Figure 1, that there are a substantial number of different methods in existence. This leads to the obvious questions: Why are there so many approaches? Is there no standard? Each approach focuses on some specific aspects and proposes suitable models, techniques and vocabularies. For instance, WSDM (Web Site Design Method) [16] is mainly focused on the design of Web sites from a user-centered perspective. It proposes a specific way to deal with different audience classes and roles and, in this aspect, is one of the most interesting approaches. However, for its navigational and conceptual models, its approaches are quite similar to OOHDM and EORM (Enhanced Object-Oriented Relationship Methodology) [17], although it uses a different vocabulary and modeling notation.

The overwhelming number of approaches and vocabularies is one of the most criticized aspects of the Web methodology community. Approaches are often defined without connection and are not compatible with one another. Regrettably, a survey of the literature on Web development methods would lead one to conclude that the "not invented here" syndrome is rife, with numerous authors independently devising their own modeling notations to represent very similar concepts in quite different ways. This has led to a fragmentation rather than a coming together of the cumulative body of knowledge, reminiscent of the early years of the object-oriented paradigm. More recently, some approaches such as WebRE [18] have been developed in order to try to solve this problem of incompatibility. WebRE is a methodology which deals with Web requirements based on W2000 [19], NDT (Navigational Development Technique) [20], UWE (UML Web Engineering) [21] and OOHDM.

Another important observation that can be noticed from Figure 1 is the varied coverage by methods of the development phases. In the Figure, each approach is located in the phase where its main focus lies. Thus, although the UWA Project [22] or WebML (Web Model Language) [23] give some consideration to requirements definition and implementation, they mainly emphasize the analysis and design phase. As can be seen, the majority of Web development methods are concentrated within the analysis and design phase, with noticeably less focus on the other phases of the life cycle.

One particular aspect of Web engineering that remains problematic is the lack of integrated toolsets to support development methods and approaches, a long-standing difficulty alluded to some years ago in [8]. Because of the frequent changes in Web systems and the

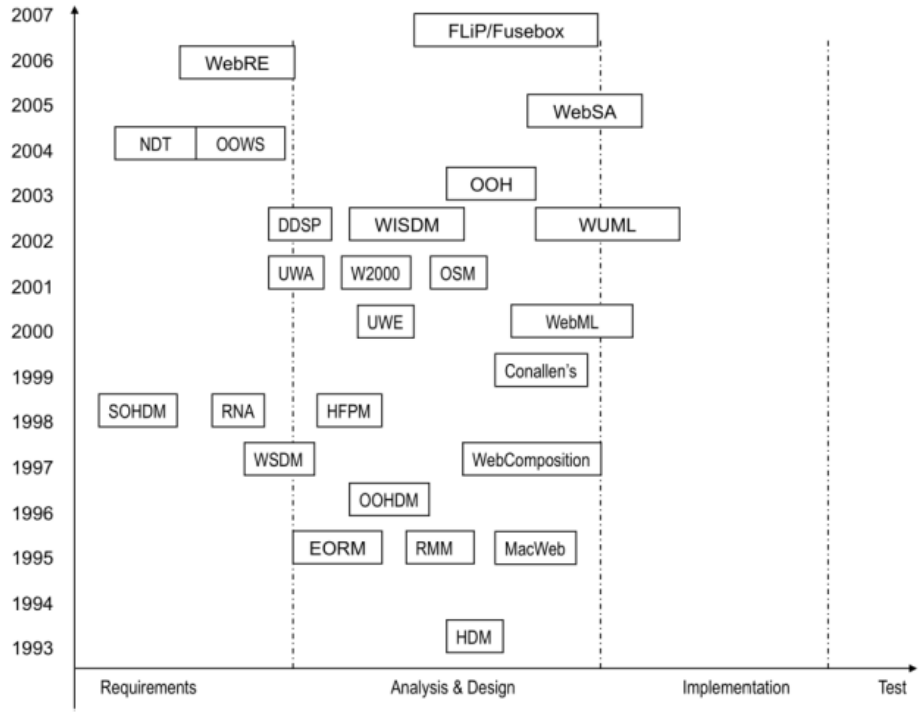

FIGURE 1. The evolution and coverage of the best-known Web development methodologies

imperative to release fully functional upgrades quickly and often, Web development methods must be highly agile. The use of CASE tools that provide automated processes and enable rapid development/re-factoring is therefore necessary. In recent years, approaches such as UWE, which offers a tool named MagicUWE [24], and WebML, which is supported by the WebRatio tool [25], have been greatly welcomed. Nevertheless, for CASE tools to be interoperable and interchangeable between and across Web development methods, it is essential that there must be a mechanism to facilitate the transformation and consistent integration of semantic metamodels. In this regard, MDWE offers much promise because it potentially enables Web developers to mix-and-match method fragments taken from different approaches and combine them into a tailored hybrid which is customized to the needs of a particular development project. This paper offers a critical view about this possibility by analyzing if approaches can be easily integrated or extended with new approaches.

2.2. **Model-Driven Web Engineering (MDWE).** Several comparative studies and surveys of Web development methodologies have drawn attention to areas where further research is needed to address a number of clearly identified gaps and shortcomings. Within the Web engineering community, a number of research groups are working towards suitable resolutions to these gaps, which as already outlined in the previous section can be broadly classified within three areas:

- There are a wide variety of Web development methodologies, using a multiplicity of different notations, models and techniques; this lack of homogeneity and standardization is unnecessarily confusing and counter-productive because, although the underlying concepts and principles of many of these methodologies are quite similar, the fact that they use their own way of doing things hinders interoperability.
- As can be seen in Figure 1, no single Web development approach provides coverage for the whole life cycle, and this absence of a single "all-in-one" solution means that

Web developers must mix-and-match aspects from different approaches, hence the need for methods that are compatible and interoperable.
- There still remains a lack of tool support for Web development methodologies, and conversely a lot of development tools lack methodical analysis/design components, so there is a bilateral disconnection between development tools and development methodologies, especially between analysis/design and implementation.

All of these issues can be addressed to some extent by adopting a model-driven development paradigm such as MDWE. This paper presents a novel contribution since it mainly focuses on analyzing approaches oriented to the model-driven paradigm. In MDWE, concepts have the greatest importance, independent of their representations. MDWE proposes the representation of concepts using metamodels which are platform-independent. The development process is supported by a set of transformations and relations among concepts that enables agile development and assures consistency between models.

The model-driven paradigm is being used with excellent results in some areas of software engineering and development. This suggests it could be also applied in Web engineering. For instance, in software products lines, MDE is offering a suitable way to assure traceability and products derivation [26,27].

It is also offering promising results in the area of programming languages. Thus, some important frameworks for Web system development based on MVC (Model View Controller) provide an easy way to build Web software. Struts [28], Django [29], and Ruby on Rails [30] are relevant examples. They are open source Web applications frameworks which use the MVC architecture and combine simplicity with the possibility of developing Web applications by writing so many codes as possible and using a simple configuration. In fact, the base of these frameworks is also in MDE.

MDE has also been recently used in the test phase too. TDD (Testing-Driven Development) is a relatively new direction of research which is providing important results. Both, the definition of metamodels to represent test aspects and the use of transformations to derive test cases are also interesting research areas [31-33]. Additionally, every day is more prevalent in university teaching for its multiple applications and utilities [34].

It can therefore be seen that the use of MDE in different areas of software development has increased considerably in recent years, embracing programming, architectures, software products lines, testing, SOA (Software Oriented Architecture) development, aspect programming, etc. This paradigm is being adopted in all these areas with relevant results, and has also been applied to Web engineering.

MDWE (Model-driven Web Engineering) refers to the use of the model-driven paradigm in Web development methodologies [35]. It helps to derive models in a specific point of the development process by using the knowledge acquired in the previous stages together with the models previously developed.

Such is the allure of MDWE that a number of the "classic" Web development approaches shown in Figure 1 are now evolving to embrace this new paradigm, as explained in Section 3. In order to analyze this evolutionary process, it is necessary to firstly clarify how MDWE can fill some of the aforementioned gaps in Web engineering listed at the beginning of this section.

Metamodels provide a solution for the multiplicity of vocabularies and approaches. A metamodel is an abstract representation of concepts. It does not focus on terminology or the way of expressing concepts. It only focuses on the concept itself. Thus, for instance, a storage requirement represents the necessity of the system to store information about content. In UWE, it is represented with a UML class and named Content. In NDT, it is called a Storage Information Requirement and it is described with a special pattern.

Nevertheless, the concept is the same. Hence, a common metamodel can be defined and some transformations from the common metamodel to the specific approach can then be declared.

By the definition of common or standard metamodels, Web development methodologies can become compatible and the differences in vocabulary together with the lack of connection among different approaches can be solved. A development team can use the most powerful idea of each approach and, through transformations, obtain advantages of other approaches.

As indicated in Figure 1, there is no single approach that covers the complete development life cycle in depth and each approach has its own particular strengths. Thus, a development team could be interested in applying the requirements approach of NDT with the aim of capturing the business knowledge, the analysis and design phases of UWE and the code generation of WebML. This can be possible if a suitable set of metamodels and transformations is defined. An example of this idea is illustrated in Figure 2. Starting with the requirements phase of NDT, after some transformations, it could be moved into the common model (a concrete instance of the metamodel in this project). After that, transformations could be applied to get UWE analysis and design and the process could be repeated to use the code generation of WebML. This hypothetical scenario could enable a developer to benefit from the advantages separately provided by each approach, and through the synergy achieved by combining different parts of different methods the problem of lack of full lifecycle coverage can be addressed.

Obviously, the quality of both the metamodel and transformations is fundamental in obtaining suitable results. To define a common metamodel is a hard task, and it is necessary to achieve a high degree of abstraction to define concepts and find common concepts. As mentioned in [36], there are some important studies that deal with the use of metamodels to fuse or make compatible different approaches.

If the use of tools is necessary in Web engineering, it is essential in MDWE. If Figure 2 is again analyzed, it is noticed that these ideas cannot be applied without tool support. Transformations must be carried out automatically and the development team should not have to apply them manually. Although tools to define metamodels and transformations are still in the early development stages, some important advances are being carried out. Thus, SmartQVT [37] and Moment [38] are two good examples. In particular, it is notable that these tools are methodology-independent because they are based on standards such

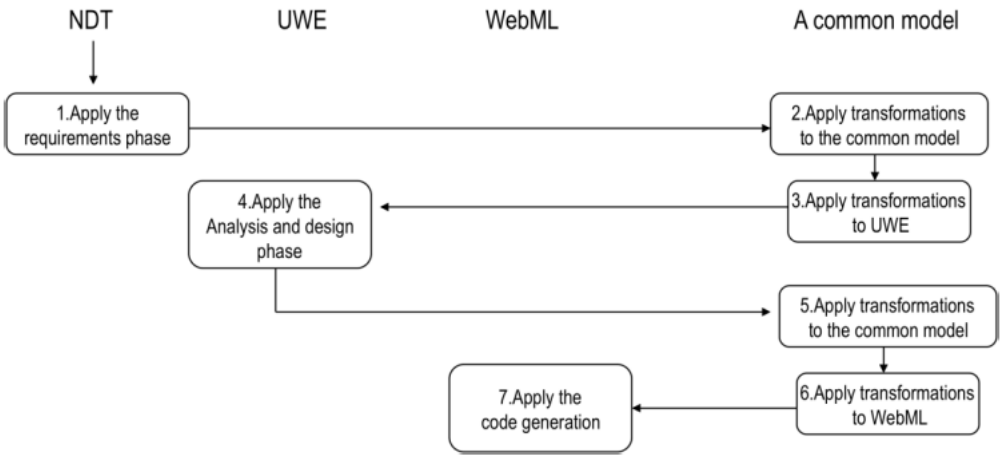

FIGURE 2. Use of common metamodels to make approaches compatible

as UML profiles [39] and QVT languages [40]. Therefore, if a metamodel of any Web development approach is defined using standards, any of these new tools is suitable.

The absence of practical applications is the only area that cannot be directly solved with MDWE. Nevertheless, in the very few practical applications that have been published, the results are promising [20].

However, as it can be deduced from this introduction, this paper offers an overall review of the situation and analyzes how MDE can solve the classical problems detected in Web development in the last years.

2.3. **Model-Driven Architecture (MDA).** MDA [41] is the standard Model-Driven Architecture defined by the Object Management Group (OMG) in 2001. It is oriented towards outlining a common architecture in the MDE environment. In MDA, four levels are proposed:

- CIM (Computer-Independent Model): This level defines concepts that capture the logic of the system. For instance, the business and the requirements models are included in this level.
- PIM (Platform-Independent Model): This level groups concepts that define the software system without any reference to the specific development platform. For instance, analysis artifacts are included in this level.
- PSM (Platform-Specific Model): In this level, computer-executable models that depend on the specific development platform are defined, such as models for Java or .NET.
- Code: This is the highest level and includes the implementation of the system.

In MDA, some transformations can be defined among these levels. Thus, CIM-to-PIM, PIM-to-PSM or PSM-to-code transformations can be defined. Furthermore, transformations on the same level, for instance PIM-to-PIM, can be defined in MDA. In Section 4 of this paper, MDA is used as a basic reference framework to compare and study a number of MDWE approaches. Most of these approaches define their metamodels and

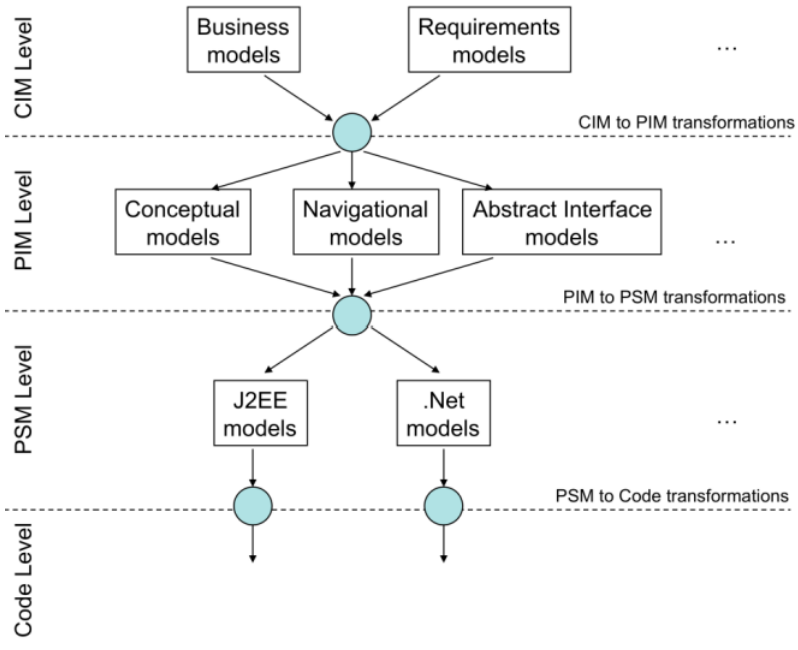

FIGURE 3. MDA structure for Web engineering (based on [42])

transformations based on the MDA standard, although they each focus on different levels of the MDA standard.

3. **Web Development Methods Based on the MDE Paradigm.** This section presents a number of Web development methods that are based on the model-driven paradigm, some of which are evolutions of classic approaches such as OOHDM and HDM. The main features of each approach are outlined, as well as a summary of advantages and disadvantages and references to where metamodels and transformations can be obtained.

3.1. **OOHDMDA.** OOHDM has been one of the most important methodologies in Web engineering. Originally proposed in 1995 [15], it presented important ideas such as separating the design of a Web system into three models: conceptual model, navigational model and abstract interface model. This idea was taken up by several later approaches. In its first incarnation OOHDM only covered design and implementation. However, it was later enriched with a specific technique, UID (User Interface Diagrams) [43], to deal with requirements.

OOHDMDA [44,45] is an MDE approach based on OOHDM. Starting as a PIM model designed with OOHDM, a servlet-based PSM is generated. OOHDMDA provides a Web application design with a UML-based design tool using the conceptual and the navigational model of OOHDM. With this base, the approach starts with the XMI file generated from the tool. Both models are enriched with behavioral semantics that are obtained from behavioral model classes incorporated in the approach. With this PIM XMI file, the approach defines some servlet-based transformations in order to obtain a PSM-XMI file with specific servlet technology. In this way, the approach offers some PIM-to-PSM transformations starting with OOHDM and ending up with servlet technology.

Although the approach is based on MDE, there is no specific PIM metamodel for OOHDMDA. Of course, MDE only implies that a development approach uses models and transformations without necessarily requiring the existence of a metamodel; transformations are not always from metamodel to metamodel, they may also be from model to model. As an extension of OOHDM, OOHDMDA naturally uses the OOHDM metamodel. OOHDM concepts are defined as stereotypes in a UML-based design tool and its transformations are generated with Java.

In the PSM level, OOHDMDA includes two specific metamodels [44]: servlet-based PSM for dynamic navigation and another one for advanced navigation. However, OOHDMDA mainly highlights the PSM level and although there is a departure from standards in the definition of the PIM metamodel and transformation, the approach taken is reasonable practical and illustrative examples can be found in [44,45]. Moreover, the use of tools in the OOHDMDA development approach offers a suitable environment for practical usage.

OOHDMDA is interesting because it shows how MDE can help to fuse separate approaches. In fact, OOHDMDA is an extension of OOHDM, which adds new concepts in the PSM level and uses the abstraction of metamodels for PSM generation. It defines new concerns to those already established by OOHDM and isolates the transformations of the implementation of the tool which supports the methodology.

3.2. **WebML.** As defined by its authors [23], WebML is a notation to specify the conceptual design of complex Web sites. Its development process starts with the conceptual modeling of the system, using a data model. In this phase, WebML does not define its own notation and instead proposes the use of standard modeling techniques such as Entity-Relation diagrams or UML class diagrams.

The process continues with the definition of a hypertext model. In this model, hypertexts that can be published on the Web are described. Each "hypertext" defines a view of the Web site. Hypertexts are described by means of two models: the composition model, which defines the pages and "content units" in the system, and the navigation model, which describes the navigation through these pages. The next step develops the presentation model, which defines the physical appearance of the Web pages. Finally, the personalization model underlines how the system has to be adapted to each user's role.

One of the most interesting contributions of WebML is that it offers a CASE tool named WebRatio [25] which enables the proposed techniques to be applied systematically.

Although WebML generates code from PIM models, there is no formal model-driven definition of the approach in the form of a metamodel or a set of formal transformations. In fact, two alternative metamodels can be found for WebML in the literature. The first, which is henceforth referenced as WebML_1 [46], is a MOF for the analysis and design WebML models. This metamodel is divided into four packages: CommonElements, DataView, HypertextView and PresentationView, one for each of the models that WebML considers. In each package, specific metaclasses, meta-associations and constraints represent each artifact of the methodology. Moreno et al. [46] use OCL (Object Constraint Language) in order to express these constraints. Transformations are not proposed in this approach as it mainly focuses on defining a metamodel for WebML.

The other metamodel for WebML, here called WebML_2, is the result of a study by Schauerhuber et al. [47] that attempts to ease the application of MDE techniques into Web modeling languages. They present a semi-automatic approach that allows the generation of MOF-based metamodels from DTD (Document Type Definition). These metamodels are also divided into packages that follow the initial definition of analysis and design of WebML metamodels: Hypertext Organization, Access Control, Hypertext, Content Management and Content. Some OCL constraints are also incorporated in order to represent restrictions in metaclasses and associations. In this approach some transformations are defined in order to obtain WebML metaconcepts from DTD ideas. These transformations offer suitable reusability of the solutions to cope with some disadvantages detected in the use of DTD. Transformations are defined in an informal way with a corresponding matrix included in a Metamodel Generator (MMG).

These two metamodels are quite similar despite the different ways in which they group WebML concepts. Although it is not an aim of this paper, their comparative study could prove very interesting. Some formal work has already been carried out for the translation of WebML models into a formal MDA environment [31].

3.3. **W2000.** As mentioned in a previous section, the W2000 approach [48] evolved out of HDM [1]. However, W2000 and HDM differ in two basic regards. Firstly, HDM is essentially an extended E-R metamodel, not a methodological proposal, whereas W2000 proposes a life cycle to develop Web systems. Secondly, W2000 is based on the object-oriented paradigm. However, despite these differences, the fundamental concepts of HDM have been inherited by W2000 and adapted to the object-oriented paradigm.

The W2000 life cycle starts with a requirements analysis phase, mainly based on use cases. By using the knowledge acquired from this requirements phase, the process goes on to the hypermedia design phase where two models are developed: the conceptual and the navigational model. To this end, W2000 modifies and extends some UML models such as the class diagram and the state diagram. The last phase is the functional design phase where the sequence diagram is used to express the functionality of the system.

In a more recent work [19], W2000 is presented as a MOF metamodel. In this metamodel, only concepts related to the analysis metamodel are presented. The metamodel is

structured into four packages which are related to each of the models defined by W2000 in the analysis phase: Information, Navigation, Presentation and Dynamic Behavior.

The abstract specification of the metamodel and the organization of metaclasses seem very relevant; however, this approach only covers the definition of metamodels and some constraints among concepts expressed with OCL. There are no transformations defined in the approach. Thus W2000 is just at the first stage of embracing the MDE paradigm.

3.4. **UWE.** UWE (UML Web Engineering) is one of the most cited techniques in Web engineering and one of the first techniques that evolved into the MDE paradigm. UWE is a Web approach that covers the complete life cycle although it is mainly focused on the analysis and design phase. One of its most important advantages is that all its models are formal extensions of UML. UWE uses a graphical notation that is entirely based on UML It enables the use of UML-based tools and reduces the learning time of Web developers who are already familiar with UML. Tool support for UWE is available in the form of MagicUWE, which offers a plug-in for MagicDraw with any artifact of UWE [49].

UWE follows the idea of model separation introduced by OOHDM, although it proposes the inclusion of some new characteristics, such as Adaptations and Presentation. Its MDE approach is perhaps one of the most complete since it offers a metamodel for each model of UWE: Requirements, Content, Navigation, Presentation and Process, together with the set of transformations to derive some models from others [50]. The content model is based on class diagrams of UML whereas the requirements model is based on WebRE [18]. Additionally, UWE has defined profiles in order to work with these metamodels. This profile definition is an efficient way to incorporate UWE metamodels in any UML-based design tool that included this possibility.

In regard to transformations, UWE defines them by using QVT as a standard language [51]. There have been some interesting experiences with the implementation of a part of its transformations by using ATL (from PIM-to-Code transformations). One such implementation is UWE4JSF, which consists of a plug-in tool defined with EMF that allows the generation of Web applications for the JSF (Java Server Faces) platform [52].

3.5. **NDT.** NDT (Navigational Development Techniques) [20] is an MDWE methodological approach mainly focused on requirements and analysis. NDT defines a set of CIM and PIM models and the set of transformations, using QVT, to derive PIM from CIM.

As occurs in other approaches, these metamodels are defined by using class diagrams. The requirements metamodel of NDT is an extension of WebRE that includes new concepts based on the WebRE approach. It also includes two metamodels, the content and the navigational, for the PIM level. The former is the UML metamodel for class diagrams and the latter is the UWE metamodel.

One of the most important advantages of this methodology is its tool support. A set of tools, called NDT-Suite, made up of four tools, supports the MDE development process of NDT (this toolset can be obtained at http://www.iwt2.org). Each metamodel of NDT has a specific profile that is implemented in Enterprise Architect [53]. The NDT methodology has adapted the interface of this tool with a set of tool boxes with direct access to each artifact of the methodology. This environment is called NDT-Profile. In addition, NDT-Suite includes six other tools:

1. NDT-Driver: A tool to execute transformations of NDT. NDT-Driver is a free Java-based tool that implements QVT Transformations of NDT and enables analysis models to be automatically obtained from the requirements models. Although transformations of NDT are completely defined using QVT, they are implemented in NDT-Driver with Java, which is very suitable for researchers working with companies on industry-based projects.
2. NDT-Quality: A tool that checks the quality of a project developed with NDT-Profile. It produces an objective evaluation of a project and assesses whether the methodology and MDE paradigm are used correctly. For this aim, NDT-Quality includes a test rule file that checks the use of QVT transformations in a NDT project.
3. NDT-Report: A tool that prepares formal documents that are validated by final users and clients. For instance, it provides the automatic generation of a Requirements Document according to the format determined by clients.
4. NDT-Prototypes: A tool that generates valuable prototypes from the NDT requirements. Because of the high level of tool support in NDT, with transformations capable of being executed automatically and assistance provided for all stages of the development life cycle, the NDT approach has been used in practice on several real projects [54].
5. NDT-Glossary implements an automated procedure that generates the first instance of the glossary of terms of a project developed by means of NDT-Profile tool.
6. NDT-Checker is the only tool in NDT-Suite that it is not based on the MDE paradigm. This tool includes a set of sheets, different for each product of NDT. These sheets give a set of check lists that should be reviewed manually with users in requirements reviews.

3.6. **OOWS.** OOWS [55] is a Web methodology which mainly focuses on the analysis phase. It is a Web extension for a previous methodology, OO-Method [56], which is based on the object-oriented paradigm and includes three models: a Structural Model, a Dynamic Model and a Functional Model. OOWS includes another two models specific to Web development: a Navigational Model and a Presentation Model.

OOWS is based on model-driven development and a recent paper [57] presents an approach for the transformation of a Web model into a set of prototypes. Firstly, this approach uses task metaphors to define requirements and these tasks are translated into an AGG graph. Using graph transformations, analysis models are then obtained. Graph grammars and graph transformations are a very mature approach for the generation, manipulation, recognition and evaluation of graphs [58], and most visual languages can be interpreted as a type of graph (directed, labeled, etc.). Thus graph transformations are a natural and intuitive way for transforming models. In contrast with other model transformation approaches, graph transformations are defined visually and are provided with a set of mature tools to define, execute and test transformations.

The OOWS approach is supported by a tool called OOWS Suite, which is a formal extension of a commercial tool named OlivaNova which supports the complete lifecycle of OO-Method. Valverde et al. [59] provide a detailed description of this tool (see http://www.care-t.com/products/).

The metamodel of OOWS is based on a MOF metamodel which is easily understood. However, its transformations are not based on OMG norms, so it is not fully compatible with other similar approaches.

TABLE 1. Web development approaches located within the MDA environment

	MDA levels			
	CIM	PIM	PSM	Code
OOHDMDA		X	X	
WebML_1		X	X	X
WebML_2		X	X	X
W2000		X	X	
UWE	X	X	X	
NDT	X	X		
OOWS	X	X		

4. **A Critical Analysis of MDWE Methodologies.** In this section, the MDWE approaches outlined in Section 3 are critically analyzed and, where it is possible and appropriate to do so, they are compared. Because the degree of definition of each metamodel or transformation is not the same in each of these MDWE approaches, in some cases there is not enough information available to compare them with the same criteria.

Before presenting the findings of our analysis, it should firstly be explained where each approach is located within the MDA framework. Table 1 represents each level of the MDA and an 'X' indicates if the MDWE approach works in this level; that is, if the approach defines metamodels and transformation oriented to the development of model in this abstract level of MDA. As previously seen in Figure 1, most of the "classic" Web development approaches were focused on analysis and design. Similarly, here again in Table 1 it is seen that most of the MDWE approaches are focused on the PIM level, which is equivalent to analysis and design within the MDA environment.

Notably, none of the approaches that we compare here covers the whole MDA. In theory, as is indicated in Figure 2, the use of common metamodels and transformations could facilitate a situation where developers choose to use models from a phase of one particular MDWE approach and transform them into models of other MDWE approaches to proceed on to the next phase of the development life cycle. Obviously, for such integration to work in practice, the authors of MDWE approaches must work together to define transformations so that approaches can be adapted for fusion. At present, interoperability of MDWE approaches is for the most part not easy if indeed feasible, but an example of how different approaches can be combined is provided in the work of Moreno et al. [35] where a common metamodel is defined to work with OOH, UWE and WebML. It has therefore been demonstrated that it is possible to overlap different MDWE approaches, thereby enabling Web developers to mix-and-match different approaches so that they can avail of the separate advantages of each approach as well as the combined benefit of integrating approaches which together support all levels of the MDA framework.

4.1. **Metamodel complexity.** The MDWE methodologies that were selected for analysis in this paper are considerably different as regards the aspects covered by their metamodels. As such, it is not possible to directly compare the metamodels of each methodology because of the variations in scope. We explored the possibility of comparing corresponding subsets of the methodologies, but this is not feasible because of the differences in the ways the metamodels are described. The purpose of this section of our analysis is therefore to provide an indication of the cognitive complexity of the metamodels of the methodologies. The rationale for looking at cognitive complexity is because previous research has shown this to be a relevant factor affecting the adoption of Web development methodologies in practice [4,13].

Cognitive complexity is a subjective notion, but it is related to structural complexity, which can be assessed using appropriate metrics [60]. To guide our analysis, a review of the literature on metamodel metrics was conducted following the general principles laid down by Kitchenham et al. [10,11]. In the methodologies analyzed, metamodels are introduced as class diagrams. For this reason, normal class diagram metrics are appropriate. Because we are interested only in the static elements of the metamodel, only class diagram metrics relating to structure were selected and others regarding behavior and functionality were omitted from our analysis. After analyzing several metric approaches [61-64], we chose to include a number of classic class diagram metrics as the basis of our analysis: number of classes; maximum number of attributes per class; maximum inheritance depth; average number of child classes inherited; and the number of new concepts presented. A detailed definition of these metrics and their general meaning in object-oriented models can be found in Pressman [64].

We must qualify our analysis by acknowledging that such metrics do not necessarily give a true and fair view of the degree of complexity of a methodology, because richer methodologies may be seen to be of greater size simply because they have broader scope and therefore have more extensive metamodels. It would be better to provide some indication of the "accidental" complexity of a metamodel (i.e., the amount of unnecessary complexity) but because this is a very difficult thing to measure we instead chose to use the aforementioned metrics of structural complexity as a proxy for overall cognitive complexity. Only those metamodels specific to each approach were considered:

- In OOHDMDA only the servlet-based PSM for dynamic navigation and for advanced navigation are included. Although OOHDMDA uses the OODHM metamodel, it is not proposed by the approach itself.
- In WebML_1, all four packages were considered: CommonElements, DataView, HypertextView and PresentationView.
- In WebML_2, five packages were considered: Hypertext Organization, Access Control, Hypertext, Content Management, Content.
- In W2000, its four packages were included in the survey: Information, Navigation, Presentation and Dynamic Behavior.
- For UWE, only four packages are considered: Requirements, Navigation, Presentation and Process. The Content package is not included because it is based on the UML metaclass for class diagrams.
- In NDT, only the requirements metamodel is included. This approach also uses the UML content metamodel and the UWE navigation metamodel.
- Finally, for OOWS, only Navigational and Presentation metamodels are considered since it inherits the rest of the metamodels from OO-Method.

The results of our analysis are presented in Table 2. We wish to emphasize that the purpose of Table 2 is not to directly compare methodologies, because it is not possible to do so on this basis. Nor should it be inferred that methodologies of greater dimension are of lesser usefulness, because the various methodologies have different scope. Our intention here is to provide some indication of the overall size of the metamodels contained within each of the methodologies, which we interpret as a proxy for overall cognitive complexity. Obviously, if an approach deals with a higher number of metaclasses or concepts than another, it does not mean that it is worse. However, authors should be conscious of the importance of recommending metamodels that are easily understandable, and Metamodel complexity is essential in this regard.

The number of classes defines the number of classes specific to a metamodel. It does not include classes imported from other packages. A high number of classes could reduce the

TABLE 2. Metamodel metrics for each MDWE approach

	OOHDMDA	$WebML_1$	$WebML_2$	W2000	UWE	NDT	OOWS
Number of classes	14	51	53	21	38	10	21
Number of new concepts presented	13	53	53	24	38	12	21
Maximum number of attributes per class	6	3	3	0	1	0	5
Average number of methods	1.5	–	–	–	–	–	–
Maximum inheritance depth	2	3	4	3	3	1	1
Average number of child classes inherited	1.25	2.4	2.3	1.75	2.5	1	2

readability of the metamodel. The number of new concepts measures concepts introduced by the approach in its metamodels. The number of concepts is closely related with the number of classes. In fact, concepts are normally presented as classes in metamodels. In some approaches, associations introduce new concepts. Thus, the complexity of an MDWE metamodel can be measured with the number of classes. If a metamodel has a high number of metaclasses, heritage, or associations, it will be difficult to understand. The authors of MDWE approaches should consider these two metrics because the readability of metamodels is affected by size [60]. As can be seen in Table 2, there are differences between $WebML_1$ and $WebML_2$. They offer different metamodels for the same approach, but the number of concepts is the same. In fact, both the number of concepts and the number of classes must be quite similar because each new concept must be defined in the metamodel either as a class or as a special association. If a new concept is not included in the metamodel because it is represented as a UML class, it is not considered a new concept but a UML concept. The maximum number of attributes per class is another measure of metamodel complexity. In Table 2, only those attributes presented in the metamodel diagram are included. Approaches that define transformations in a formal way, such as QVT and XML, have other attributes that, for the sake of simplicity, are not considered here.

The next two metrics are oriented towards class heritage. Heritage is one of the most relevant artifacts of class diagrams and this metric is applicable to the metamodels of the approaches included in Table 2. In classic class metrics, the maximum inheritance depth must not surpass three levels. This high number of levels makes it too complicated to understand class models. Similarity in metamodels, a large inheritance depth causes complexity in the metamodel and it is therefore difficult to follow concept definition. The average number of child classes inherited is considered another important metric. Classic metric approaches propose that the number of child classes remains small since it is also a measure of complexity.

Each author, even with the same approach, as can be observed with WebML, expresses concepts and their relations according to experience. The fact that UWE has fewer concepts than WebML does not mean that the metamodel expresses fewer semantics. In fact, UWE, NDT and OOHDMDA extend and use a high number of concepts from UML, but these are not included in Table 2.

Some interesting conclusions about metamodels in general can be drawn from Table 2. Firstly, metamodels seldom include methods since they normally express concepts and their relations and do not include information about functionality. Only OOHDMDA

includes some methods since this approach is close to model generation and these methods express the possibility of this generation.

Secondly, the use of heritage is present in all of the MDWE approaches that we compare in this study. Heritage is an important artifact to express relations and extensions of concepts. The number of child classes or the maximum inheritance depth changes in each approach, although it never reaches a high number. In fact, metamodels express concepts and the relations and constraints among them. Consequently, the authors should reduce the complexity of their approaches. The main aim of a metamodel is to present the approach as simply as possible. As can be concluded from Table 2, this tendency is followed by the approaches under study.

Finally, although it was not included in the table as a metric, an important advantage for MDWE is a profile definition. The standard definition of a profile for metamodels, based on UML, is a powerful artifact for each MDWE methodology. In a profile, each concept in the metamodel is defined as a formal extension of a UML class thereby two important advantages are assumed: the first one is related to Figure 2. If a methodology, such as UWE or WebML, defines a concept as an extension of UML activity for instance, it is easier to find a connection between these two approaches and to find similar concepts in the two metamodels.

Furthermore, as for NDT with NDT-Suite or UWE with MagicUWE, the use of a profile facilitates the use of UML-based tools. With a simple extension of UML, any commercial UML-based tool could be a suitable tool support for the methodology. Only UWE, NDT and WebML offer specific profiles for metamodels, although other approaches such as OOHDMDA use them in their approach, and as aforementioned OOHDMDA uses a profile for OOHDM.

4.2. **Metamodel concepts.** Concepts are the basic aspects that are handled in MDWE. Each of the MDWE approaches analyses in this study defines its own concepts. In some cases, these approaches coincide by using the same name for the same concept. However, in other cases, the same name is used for different concepts or various names are used for the same concept.

The lack of a standard terminology in Web engineering is a well-known and lamented problem [4-7], and indeed it caused some difficulties when conducting this analytical study of MDWE approaches.

Nevertheless, there are a number of concepts which commonly appear in most Web engineering approaches. Based on a review of the literature, including previous comparative analyses of MDWE approaches [5], Table 3 presents an overview of the scope of the MDWE approaches analyzed in this study. An 'X' indicates that the approach defines concepts that are included in its metamodel, and a shaded cell indicates that the particular approach does not cover the MDA level. It should be noted that, because each approach uses its own terminology, the row labels in Table 3 may therefore not be the same as the actual name given by each approach to the corresponding concept in its metamodel; however, the essential meaning of the concept is the same.

In the upper section of the table, models treated in the requirements phase (CIM Level) are listed, based on a classification obtained from [5].

In the lower section of the table (the PIM level), a classification of models mainly based on UWE notation is presented. Neither NDT nor OOWS cover this level directly by themselves, although NDT uses some UWE metamodels and OOWS uses the OO-Method to deal with these aspects. For that reason, NDT and OOWS are shown in the table as not covering the PIM level. Similarly, although the WebML methodology deals

with adaptation, the WebML_1 and WebML_2 metamodels do not consider this aspect, as indicated in Table 3.

TABLE 3. Models covered by each MDWE approach

		OOHDMDA	WebML_1	WebML_2	W2000	UWE	NDT	OOWS
CIM Level	Data requirements model					X	X	
	User interface requirements model					X	X	
	Navigational requirements model					X	X	X
	Adaptive requirements model					X	X	X
	Transactional requirements model					X	X	
	Non-Functional requirements model					X	X	
PIM Level	Content model	X	X	X	X	X		
	Navigational model	X	X	X	X	X		
	Presentation model	X	X	X	X	X		
	Adaptive model					X		
	Process model					X		

As was previously mentioned in Section 2, in the overview of "classical" Web engineering approaches, the conclusion can be drawn that the main characteristics studied in MDWE by these approaches are again:

- Static aspects, represented by a content model or content requirements
- Navigational aspects, represented by navigational models or navigational requirements
- Presentation aspects, represented by abstract interfaces models, users' adaptation in requirements, etc.

In this sense, MDWE follows the same line as that of "classical" Web engineering. The use of the new paradigm only offers a new way of carrying out development. As is concluded in this study, this new paradigm offers solutions for various problems such as compatibility between approaches, the use of UML-based tools, although no new concepts are introduced which are different to those of classical Web engineering ones.

4.3. **Transformations.** If metamodels are the base of MDE, transformations are its most important advantage. Transformations make model derivation easier and help maintain traceability among these models. Transformations look for connections between a previous model to another one and enables the translation of knowledge from one phase to the next more readily. Thus, for instance, if in the requirements phase the development team detects the necessity of storage data about users in the system, the CIM-to-PIM transformation will create a class in the analysis model to store users' data and the PIM-to-PSM transformation will define a persistent Java class to store this information.

Special metrics to measure the quality of a transformation were not found in the literature. The reason for this gap in the literature can be attributed to a number of factors: transformations are a new way of building software, the standards for definition (e.g., QVT) have only recently been defined, and each MDWE approach uses either a different way to express transformations or different transformations languages. Furthermore, each of the MDWE approaches that we analyzed has a different degree of development in its transformations. Notwithstanding these difficulties in forming meaningful comparisons,

TABLE 4. MDA transformations dealt with by MDWE approaches

	OOHDMDA	$WebML_1$	$WebML_2$	W2000	UWE	NDT	OOWS
Transformation of CIM to PIM						X	X
Transformation of PIM to PSM			X		X		
Transformation of PSM to Code	X				X		
Language used for transformations	Java		XSLT		QVT ATL	QVT	Graph trans.

Table 4 presents an outline of the set of transformations dealt with by each of the MDWE approaches under consideration in this study. An 'X' indicates that the approach supports the specified transformation.

It is important to point out that some of these approaches have as yet only defined a metamodel, but do not incorporate transformations. Furthermore, the definition of transformations is still a relatively unexplored area in the model-driven paradigm. OMG has defined a standard, known as QVT, which is still in its early stage and, although there are some tools that support this language, such as SmartQVT or Moment, insufficient development means that research groups cannot provide the transformations yet.

An important approach in this area is OOWS, which uses a set of graph transformations to translate from a CIM model into a PIM model. The use of graph transformations and AGG graphs addresses problems of incompatibility. Since this is the only approach under study that works with this technology, it is difficult to compare its results with the others. However, AGG graphs and their transformations represent a robust and well-studied environment. There are suitable tools that support these transformations; therefore, OOWS has implemented its translations and offers a suitable tool environment for its use.

On the other hand, NDT has defined its transformations in a theoretical way with QVT. Nevertheless, it has translated these transformations into Java and offers the derivation of models in its tool called NDT-Driver. This is a suitable solution for use in practice, but it is not the principal aim of the MDE paradigm. The ideal environment is that the MDE community could use a general and standard tool that permits the metamodel definition to use a standard language, for instance, a class metamodel. The tool should also offer a suitable environment for the definition of transformations and standard such as QVT, to define these transformations. As explained in Subsection 4.5, this is currently one of the most important areas lacking research in the MDE environment.

4.4. **Standards and compatibility.** One of the most important advantages of the MDWE paradigm is the possibility of making various approaches compatible. MDWE is focused on concepts and the way to deal with and represent these concepts is unimportant. However, if a metamodel or a concept is defined freely without reference to a common standard, the multiplicity of concepts can surface again as a problem, just as it originally did in the Web engineering approaches of the 1990's. If a metamodel or some transformations were defined using a common language, the connection among approaches could be easily facilitated.

To this end, the use of UML profiles offers very interesting results. A UML profile is an extension mechanism offered by UML to extend the basic concepts of an MDWE

approach. Thus, if an approach defines its own metamodel using a class diagram and later defines a UML profile, then it offers a standard definition of its concepts that can be understood by other researchers and groups. As examples of UML profiles, NDT provides the concept of Storage requirements which is an extension of the UML class, while UWE defines the Content concept, which is also an extension of the UML class. If both are analyzed in each approach, we can conclude that they represent the same idea, although they are named differently in each methodology. Extensions which are based on the same UML concept give rise to opportunities for forward compatibility, thereby representing an important step towards a common metamodel for Web modeling [47].

4.5. **Tools and industry experiences.** Despite the fact that Web engineering is a very active area within the research community and some very good results have been achieved, the application of many of its ideas, models and techniques within industry has not yet been realized. As one example of the low rate of knowledge transfer into practice, a recent study conducted in Spain, which interviewed more than 50 project managers and 70 analysts from a sample of 30 software companies representing local, national and international organisations, found that just 25% of medium- to large-sized companies (i.e., more than 50 employees) knew anything about "Web engineering", while only 10% of small companies had heard about it. Overall, only 1% of companies had applied Web engineering in their projects [65]. Similar experiences were found in comparable studies conducted in Ireland [4,13]. These results indicate that although Web engineering methods could be very useful, few of them are currently in use.

In MDWE the situation has not changed, there so far being very little application of academic research experiences to real projects. Of the MDWE approaches described in this paper, only a few published accounts of "real world" practical applications are known to exist. In [20], NDT shows how its initial definition evolved because of feedback from practice, and these experiences show the good results that can be obtained. Two relevant advantages offered by the MDWE paradigm are the reduction of the development time by using transformations and the concordance among models in different phases. WebML, and principally its tool, WebRatio, have also been applied with successful results within real enterprises.

The need for translational research to move research findings from academic research laboratories into the world of practice is widely accepted as a firm prerogative within applied disciplines such as Web engineering. However, practical application is not possible without a set of suitable tools that facilitate the application of models, techniques, transformations and the maintenance of model coherence.

Despite this fact, an important advantage attaches to the approaches examined in this paper. Through metamodels, standards, profiles and basic tools under the MDE paradigm, the evolution of tools is better in MDWE than in Web engineering. The lack of suitable tools has always been a disadvantage of the "classic" Web engineering methods. In this survey of MDWE approaches, all the studied approaches offer suitable tool environments for its application.

The use of profiles to facilitate tool support is potentially very interesting. With profile definition, UML-based tools can provide a suitable solution for any MDWE and it reduces the cost of learning curve because they are quite friendly for development teams, which make easier the application of these approaches in enterprise environment. In fact, if the definition of MDE standard languages evolves in the next few years, then the use of general UML-based tools for MDWE should become a reality. This idea is being followed by OOHDMDA, NDT and UWE and they are offering suitable and adaptable results. However, in MDE in general, there is an important gap in tools that offer the possibility

of defining a metamodel and transformations which can be executed into concrete models in real projects.

In addition, the research community needs tools to support the MDE process successfully. To implement an MDWE approach, it is necessary in the first instance to have a tool which can represent metamodels and transformations written, for example, in QVT. In this sense, the EMF or ATL environments offer promising results, although MDE also needs a defined concrete syntax to represent its metamodels. For instance, in the case of NDT, metamodels are not used by development teams in practice; they use a set of tools, defined in NDT-Suite, that represent each artifact of the approach as a UML artifact extension.

As yet, UML-based tools do not offer the possibility of writing transformations in a standard language. One solution that researchers are proposing as a resolution of this issue consists in writing transformations in a standard language and later implementing them with programming languages. For instance, WebRatio uses XML or NDT uses Struts. However, this suggestion does not seem suitable enough because, in fact, a change in the transformations implies a manual change in the code for executing these transformations. As yet, the existing limitation of tools means that no other possibilities are available. Some new developments, like Moment or SmartQVT, or the inclusion of MDA transformation languages in UML-based tools, like the case of Enterprise Architect, offer promising solutions.

5. **Related Work.** Although the analysis in the previous section focused on the main MDWE methodologies, there is also some other interesting work going on within this research area. In comparative studies on Web approaches, a general conclusion is that similar concepts are used or represented with a different number of models, techniques or artifacts. Thus, for instance, navigational classes are presented with different elements in UWE, OOHDM, NDT and W2000. Escalona and Koch [18] show how a metamodel can represent a concept independently from its representation or notation; only concepts are important. A metamodel for Web requirements called WebRE, which represents requirements models of W2000, NDT, OOHDM and UWE, is presented. In [21], their work goes on by using QVT to obtain analysis models from this metamodel. These papers are interesting since they are completely based on UML and on QVT, standards defined by OMG, although the work can be considered to be excessively theoretical.

This tendency to use metamodels and transformations to make different approaches compatible is applied in a recent work under the name of MDWEnet [37] which is an initiative carried out by a representative group of MDWE researchers in an effort to find a common approach which allows various approaches to be represented and handled.

Fernández and Mozón [66] present the possibilities of working with metamodels and tools, and show how a requirements metamodel can easily be defined in IRqA (Integral Requisite Analyzer), which is a commercial tool that helps in the definition of metamodels for requirements [67]. In this way, this paper reveals the power of the tools supporting metamodels as they are suitable for any approach defined by using metamodels. This work is very practical in fact, although it is not an approach for the Web. Metamodels do not offer specific artifacts to deal with the Web environment since it only offers an approach for classic requirements treatment [68].

In [69], Meliá and Gómez analyze an approach called WebSA (Web Software Architecture) which provides the designer with a set of architectural and transformation models used to specify a Web application. Although these models only work in the design phase, this approach is very relevant since MDA and QVT are applied in a very exhaustive way.

To conclude, the use of metamodels and MDE are areas of software engineering that are becoming widely accepted as a solution for classic problems in Web engineering.

6. **Conclusions and Future Work.** This paper presents an overview of how classic Web engineering methodologies have evolved to embrace the model-driven paradigm. A brief review of some of the most relevant Web approaches working on the model-driven paradigm was given, and the findings of an analysis of model-driven Web development methodologies were set out.

Although Web engineering is now an established branch of software engineering, this paper argues that there are a number of long-standing problems to be covered that could potentially be addressed by using MDWE. One of these gaps is the multiplicity of methodologies which, given the lack of standards, means that Web developers cannot interoperably mix-and-match the products of different phases of different methodologies. As can be seen in Figure 1, there are many Web development methodologies, all the more evident from the compendium of over fifty methods and approaches compiled by Lang and Fitzgerald [13]. Previous studies conclude that many of these methodologies have their own particular strong points above the others [4-7]. Furthermore other more recent approaches for Web application development, as proposals derived from other engineering areas (for example, the idea from Dynamic Interactive Systems (DIS), of modeling and synthesizing fully functional Web-based interactive applications using the incremental, component wise, correct-by-construction approach named Equivalent Transformation (ET)).

We are not arguing that there should be a standard universal Web development approach, but it is important that whatever method or methods chosen by a development team for any given project may be capable of integration. The model-driven paradigm can offer a suitable solution for this problem. As shown in Figure 2, the use of MDWE can help to fuse approaches, thereby benefiting from the respective advantages of each individual method. Work referred to in Section 5, such as common metamodels and WebRE, is offering interesting results along these lines.

Another problem in Web engineering is the lack of tools that offer suitable support for the development environment. As can be deduced from the comparative analysis of OOHDMDA, UWE, OOWS and NDT presented in this paper, MDWE shows a good solution to this problem by means of using metamodels and profiles. Hence, it is not necessary to define a specific tool for each approach. If a metamodel and a suitable profile are defined, UML-based tools can be used in the approach. Thus, with only the profile definition for instance, for NDT, then Enterprise Architect, IBM Rational Rose, ArgoUML and StarUML can be used for the application of the methodology. The use of suitable tools for the application of these approaches is one of the most important issues in the enterprise application of this kind of solutions. In fact, MDWE approaches offer more empirical experiences that Web engineering, although there is too much work to do in this line.

However, although results in this area are encouraging, further work is necessary. Profile and metamodel definitions are well-supported by these tools, but transformations are not. Some tools, such as Enterprise Architect, define their own MDA language. They must be based on standards in order to offer flexibility. Thus, if a methodology defines a suitable profile and transformations using these standards, any tool could be used to support the development with this methodology. Although there are some solutions, such as the use of ATL, the implementation of transformations with Java or the use of Graph transformations, UML-based tools must evolve along this line.

The availability of tools and the possibility of fusing several approaches are placing MDWE closer to the being used in industry-strength "real world" projects with suitable results. MDWE offers important advantages for companies. For instance, transformations and systematic model generation can reduce the development time, especially if a tool is used. With MDWE, the knowledge reached in one phase of the life cycle is carried over to the next phase by means of using transformations. Furthermore, with metamodel constraints, traceability can be checked systematically, thereby solving major errors, inconsistencies and mistakes in the first phases of the life cycle.

Despite the fact that MDWE offers suitable results, there are still some important areas that must be considered in this survey. The first one is the "feedback" in the life cycle. For instance, if the requirements phase is completed and the analysis is generated from requirements results, then, how can future changes in requirements be incorporated into the analysis? The MDWE approaches that we analyzed are working in this direction. Thus, NDT, for instance, includes a specific method of generation in NDT-Driver to solve this problem. However, in general, it is future work for approaches included in this survey.

Another line of future work is research oriented towards practical application. Although MDWE offers suitable aspects to be applied within industry, there have been very few practical applications up to date. Research groups must work together towards a common aim of extending MDWE research from academic laboratories into practical settings, trying to apply in the future this approach for designing Web sites and to complement it with other quality website design techniques. Importantly, this should include guidance on the practical limitations of applying MDWE methodologies, such as experience reports on which methodologies work best in different circumstances.

As a final conclusion, MDE is a relatively new paradigm suitable for Web engineering and offers a productive research line for the Web community. However, it is still in its development stages and needs further research to offer more attractive solutions for its application in practice.

Acknowledgements. This research has been supported by the Tempros project (TIN20 10-20057-C03-02) and the National Network Quality Assurance in Practise. CaSA (TIN20 10-12312-E) of the Ministry of Education and Science, Spain and by the project NDTQ-Framework (TIC-5789) of the Junta de Andalucia, Spain.

REFERENCES

[1] F. Garzotto, D. Schwabe and P. Paolini, HDM – A model-based approach to hypermedia application design, *ACM Trans. on Information Systems*, vol.11, no.1, pp.1-26, 1993.

[2] G. Rossi and D. Schwabe, Modelling and implementing web applications with OOHDM, in *Web Engineering: Modelling and Implementing Web Applications*, G. Rossi, O. Pastor, D. Schwabe and L. Olsina (eds.), Springer, 2008.

[3] Y. Deshpande, S. Marugesan, A. Ginige, D. Schwabe, M. Gaedke and B. White, Web engineering, *Journal of Web Engineering*, vol.1, no.1, pp.3-17, 2002.

[4] C. Barry and M. Lang, A survey of multimedia and web development techniques and methodology usage, *IEEE Multimedia*, vol.8, no.3, pp.52-61, 2001.

[5] M. J. Escalona and N. Koch, Requirements engineering for web applications: A comparative study, *Journal of Web Engineering*, vol.2, no.3, pp.193-212, 2004.

[6] M. J. Escalona, J. Torres, M. Mejías, J. J. Gutiérrez and D. Villadiego, The treatment of navigation in web engineering, *Advances in Engineering Software*, vol.38, no.4, pp.267-282, 2007.

[7] W. Schwinger, W. Retschitzegger, A. Schauerhuber et al., A survey on Web modeling approaches for ubiquitous Web applications, *International Journal of Web Information Systems*, vol.4, no.3, pp.234-305, 2008.

[8] M. Lang, Hypermedia system development: Do we really need new methods? *Proc. of the Informing Science + IT Education Conference*, Cork, Ireland, pp.883-891, 2002.

[9] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner and M. Khail, Lessons from applying the systematic literature review process within the software engineering domain, *Journal of Systems and Software*, vol.80, no.4, pp.571-583, 2007.
[10] B. Kitchenham and S. Charters, Guidelines for performing systematic literature reviews in software engineering, Version 2.3, *Technical Report EBSE-2007-01*, Software Engineering Group (SEG), School of Computer Science and Mathematics, Keele University and Department of Computer Science, University of Durham, UK, 2007.
[11] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey and S. Linkman, Systematic literature reviews in software engineering – A systematic literature review, *Information and Software Technology*, vol.51, no.1, pp.7-15, 2009.
[12] M. Lang, A critical review of challenges in hypermedia systems development, in *Information Systems Development: Advances in Theory, Practice and Education*, O. Vasilecas, A. Caplinskas, W. Wojtkowski et al. (eds.), Heidelberg, Springer, 2005.
[13] M. Lang and B. Fitzgerald, New branches, old roots: A study of methods and techniques in Web/Hypermedia systems design, *Information Systems Management*, vol.23, no.3, pp.62-74, 2006.
[14] T. Isakowitz, E. Stohr and P. Balasubramanian, RMM: A methodology for the design of structured hypermedia applications, *Communications of the ACM*, vol.38, no.8, pp.34-44, 1995.
[15] D. Schwabe and G. Rossi, The object-oriented hypermedia design model, *Communications of the ACM*, vol.38, no.8, pp.45-46, 1995.
[16] O. de Troyer and C. Leune, WSDM: A user-centered design method for web sites, *Computer Networks and ISDN Systems*, vol.30, no.1-7, 85-94, 1998.
[17] D. Lange, An object-oriented design approach for developing hypermedia information systems, *Journal of Organizational Computing and Electronic Commerce*, vol.6, no.3, pp.269-293, 1996.
[18] M. J. Escalona and N. Koch, Metamodelling the requirements of web systems, in *Web Information Systems and Technologies*, J. Filipe, J. Cordeiro and V. Pedrosa (eds.), Springer, 2007.
[19] L. Baresi, F. Garzotto and M. Maritati, W2000 as a MOF metamodel, *The 6th World Multi-Conference on Systemics, Cybernetics and Informatics*, Orlando, Florida, USA, 2002.
[20] M. J. Escalona and G. Aragón, NDT: A model-driven approach for web requirements, *IEEE Trans. on Software Engineering*, vol.34, no.3, pp.370-390, 2008.
[21] N. Koch, A. Knapp, G. Zhang and H. Baumeister, UML-based web engineering, in *Web Engineering: Modelling and Implementing Web Applications*, G. Rossi, O. Pastor, D. Schwabe and L. Olsina (eds.), Springer, 2008.
[22] UWA Consortium, *Requirements Elicitation: Model, Notation, and Tool Architecture*, Ubiquitous Web Applications Consortium (Deliverable D6), 2001.
[23] S. Ceri, P. Fraternali and A. Bongio, Web modelling language (WebML): A modelling language for designing web sites, *Computer Networks*, vol.33, no.1-6, pp.137-157, 2000.
[24] *MagicUWE*, http://uwe.pst.ifi.lmu.de/toolMagicUWE.html, 2011.
[25] *WebRatio*, http://www.webratio.com, 2011.
[26] B. Pérez, M. Polo and M. Piatini, Software product line testing – A systematic review, *The 4th International Conference on Software and Data Technologies*, Sofia, Bulgaria, 2009.
[27] S. B. Abid, Resolving traceability issues in product derivation for software product lines, *The 4th International Conference on Software and Data Technologies*, Sofia, Bulgaria, 2009.
[28] *Struts*, http://struts.apache.org, 2011.
[29] *Django*, http://www.djangoproject.com, 2011.
[30] *Ruby on Rails*, http://rubyonrails.org/, 2011.
[31] M. Brambilla, P. Fraternali and M. Tisi, A transformation framework to bridge domain specific languages to MDEA, in *Models in Software Engineering*, M. R. V. Chaudron (ed.), Springer, 2009.
[32] E. Robles Luna, J. Grigera and G. Rossi, Bridging test and model-driven approaches in web engineering, in *Web Engineering*, M. Gaedke, M. Grossniklaus and O. Díaz (eds.), Springer, 2009.
[33] J. J. Gutiérrez, C. Nebut, M. J. Escalona, M. Mejías and I. Ramos, Visualization of use cases through automatically generated activity diagrams, *Proc. of the 11th International Conference on Model Driven Engineering Languages and Systems, LNCS*, Toulouse, France, vol.5301, pp.83-96, 2008.
[34] W. B. Lin, S. J. Wang, H. K. Chiang and Y. Y. Lu, The implementation and design of web-based e-learning platform for class B skill computer maintenance, *ICIC Express Letters*, vol.5, no.9(B). pp.3367-3374. 2011.

[35] N. Moreno, J. R. Romero and A. Vallecillo, An overview of model-driven web engineering and the MDA, in *Web Engineering: Modelling and Implementing Web Applications*, G. Rossi, O. Pastor, D. Schwabe and L. Olsina (eds.), Springer, 2008.
[36] A. Vallecillo, N. Koch, C. Cachero et al., MDWEnet: A practical approach to achieving interoperatiblity of model-driven Web engineering methods, *The 3rd Workshop on Model-Driven Web Engineering*, pp.246-254, 2007.
[37] *SmartQVT*, http://sourceforge.net/projects/smartqvt, 2011.
[38] P. Queralt, L. Hoyos, A. Boronat, J. A. Carsí and I. Ramos, Un motor de transformación de modelos con soporte para el lenguaje QVT relations, *Proc. of CEUR Workshop*, Spain, 2006.
[39] OMG, *Unified Modeling Language 2.1.1 Superstructure Specification (OMG doc. formal/07-02-05)*, http://www.omg.org/cgi-bin/doc?formal/07-02-05, 2007.
[40] OMG, *MOF QVT Final Adopted Specification*, Object Management Group (OMG doc. ptc/05-11-01), 2005.
[41] OMG, *MDA Guide*, Version 1.0.1, http://www.omg.org/docs/omg/03-06-01.pdf, 2003.
[42] N. Koch, G. Zhang and M. J. Escalona, Model transformations from requirements to Web system design, *Proc. of the 6th International Conference on Web Engineering*, Palo Alto, California, USA, pp.281-288, 2006.
[43] P. Vilain, D. Schwabe and C. S. de Souza, A diagrammatic tool for representing user interaction in UML, *Proc. of the 3rd International Conference on the Unified Modeling Language: Advancing the Standard, LNCS*, York, UK, vol.1939, pp.133-147, 2000.
[44] H. A. Schmid, Model-driven architecture with OOHDM, in *Engineering Advanced Web Applications*, M. Matera and S. Comai (eds.), Munich, Germany, Rinton Press, 2004.
[45] H. A. Schmid and O. Donnerhak, OOHDMDA-An MDA approach for OOHDM, *Proc. of the 5th ICWE, LNCS*, Sydney, Australia, vol.3579, pp.569-574, 2005.
[46] N. Moreno, P. Fraternalli and A. Vallecillo, A UML 2.0 profile for WebML modelling, *Proc. of the 6th International Conference on Web Engineering*, Palo Alto, California, USA, 2006.
[47] A. Schauerhuber, M. Wimmer and E. Kapsammer, Bridging existing Web modeling languages to model-driven engineering: A metamodel for WebML, *Proc. of the 2nd International Workshop on Model-Driven Web Engineering*, Palo Alto, California, USA, 2006.
[48] L. Baresi, F. Garzotto and P. Paolini, Extending UML for modelling Web applications, *Proc. of the 34th Hawaii International Conference on System Sciences*, USA, pp.1285-1294, 2001.
[49] *UWE*, http://uwe.pst.ifi.lmu.de/toolMagicUWE.html, 2011.
[50] N. Koch and A. Krauss, Towards a common metamodel for the development of Web applications, *Proc. of the 3rd International Conference on Web Engineering, LNCS*, Oviedo, Spain, vol.2722, pp.419-422, 2003.
[51] N. Koch, Transformation techniques in the model-driven development process of UWE, *Proc. of the 6th International Conference on Web Engineering*, Palo Alto, California, USA, 2006.
[52] *UWE4JSF*, http://uwe.pst.ifi.lmu.de/toolUWE4JSF.html, 2011.
[53] *Enterprise Architect*, http://www.sparx.org, 2011.
[54] M. J. Escalona, G. Aragón, J. J. Gutierrez, J. A. Ortega and I. Ramos, NDT & metrica v3: An approach for public organizations based on model driven engineering, *Proc. of International Conference on Web Information Systems and Technologies*, Funchal, Madeira, Portugal, pp.224-227, 2008.
[55] J. Fons, V. Pelechano, M. Albert and O. Pastor, Development of web applications from web enhanced conceptual schemas, *Proc. of the 22nd International Conference on Conceptual Modeling, LNCS*, Chicago, IL, USA, vol.2813, pp.232-245, 2003.
[56] O. Pastor, J. Gómez, E. Insfran and V. Pelechano, The OO-method approach for information systems modeling: From object-oriented conceptual modeling to automated programming, *Information Systems*, vol.26, no.7, pp.507-534, 2001.
[57] P. Valderas, V. Pelechano and O. Pastor, A transformational approach to produce web application prototypes from a web requirements model, *International Journal of Web Engineering and Technology*, vol.3, no.1, pp.4-42, 2007.
[58] G. Rozenberg, *Handbook of Graph Grammars and Computing by Graph Transformations*, vol.1, World Scientific, New Jersey, 1997.
[59] F. Valverde, P. Valderas and J. Fons, OOWS suite: Un entorno de desarrollo para aplicaciones web basado en MDA, *X Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes Software*, Isla de Margarita, Venezuela, pp.253-266, 2007.

[60] J. Erickson and K. Siau, Theoretical and practical complexity of unified modeling language: Delphi study and metrics analyses, *International Conference on Information Systems*, Washington, DC, USA, 2004.
[61] *SDMetrics*, http://www.sdmetrics.com, 2011.
[62] S. R. Chidamber and C. F. Kemerer, A metrics suite for object-oriented design, *IEEE Trans. on Software Engineering*, vol.20, no.6, pp.476-493, 1994.
[63] M. Lorenz and J. Kidd, *Object-Oriented Software Metrics*, Prentice-Hall, 1996.
[64] R. S. Pressman, *Software Engineering: A Practitioner's Approach*, McGraw Hill, 2007.
[65] IWT2, *Web Engineering and Early Testing research Group (IWT2)*, Department of Computer Languages and Systems, University of Sevilla, Spain, http://www.iwt2.org, 2005.
[66] J. L. Fernández and A. Monzón, A metamodel and a tool for software requirements management (poster), *Reliable Software Technologies – Ada-Europe*, Potsdam, Germany, 2000.
[67] *IRqA (Integral Requisite Analyzer)*, http://www.irqaonline.com, 2011.
[68] J. Duan and Q. Zhu, A requirement-driven approach to enterprise application evolution, *ICIC Express Letters, Part B: Applications*, vol.2, no.2, pp.313-318, 2011
[69] S. Meliá and J. Gómez, The WebSA approach: Applying model-driven engineering to web applications, *Journal of Web Engineering*, vol.5, no.2, pp.121-149, 2006.