25 Years of Model-Driven Web Engineering. What we achieved, What is missing

Gustavo Rossi[1], Matias Urbieta[1], Damiano Distante[2], Jose Matias Rivero[1], Sergio Firmenich[1]
[1]LIFIA, F. de Informática, UNLP and Conicet, Argentina
[gustavo, matias.urbieta, jose.matias.rivero, sergio.firmenich]@lifia.info.unlp.edu.ar
[2]Unitelma Sapienza University, Rome, Italy
damiano.distante@unitelma.it

Abstract
Model-Driven Web Engineering (MDWE) approaches aim to improve the Web applications development process by focusing on modeling instead of coding, and deriving the running application by transformations from conceptual models to code. The emergence of the Interaction Flow Modeling Language (IFML) has been an important milestone in the evolution of Web modeling languages, indicating not only the maturity of the field but also a final convergence of languages. In this paper we explain the evolution of modeling and design approaches since the early years (the 90's) detailing the forces which drove that evolution and discussing the strengths and weaknesses of some of those approaches. A brief presentation of IFML is accompanied with a thorough analysis of the most important achievements of the MDWE community as well as the problems and obstacles that hinder the dissemination of model-driven techniques in the Web engineering field.

1-Introduction and Motivation

The explosive growth of the Web as a platform for building software applications and as a means for communication, entertainment, education and commerce has dramatically changed the landscape of software development. "Conventional" (pre-Web) applications followed a more or less clear life-cycle in which maintenance and evolution were measured in years, end-users were well known by developers and were many times "captive", and interface and interaction issues were not decisive; meanwhile, the Web introduced a new set of challenges. Web application requirements vary at a very fast pace, we have to deal with thousands of unknown users which access millions of information items and who expect personalized contents and functionality. These users are seldom loyal and they expect easy to use and effective applications. At the same time, underlying technologies, frameworks, and programming languages have been continuously evolving. To make matters worse, the new generation of user devices, i.e., smart phones and tablets, pushed the set of challenges further. As a consequence, development teams suffer more stress since they need to deliver running applications with frequently unstable requirements in short time; moreover, these

applications must run in a huge variety of devices on top of different, and many times immature, frameworks. Very different to the "old" days, developers have become polyglot since they need to know many programming languages to build single high performance applications, for which even emerging write once, deploy anywhere frameworks fall short. Customers, meanwhile, have learned to use the Web and are each day more demanding, since they have access to applications which are similar to what they need and therefore can easily imagine new and many times more sophisticated features.

The first attempt to face the chaotic development of Web applications was the emergence of the Web Engineering field [23]. Web engineering has been defined as the application of systematic, disciplined and quantifiable approaches to development, operation, and maintenance of Web-based applications.

While it is not the purpose of this paper to discuss the interesting relationships between Software Engineering and Web Engineering, it is true that there has been debate about this. Clearly Web Engineering uses (must use indeed) the techniques and principles that Software Engineering developed during decades; at the same time, however, as said before, the Web brings new kinds of problems that did not exist previously. In this sense, Web Engineering uses techniques of information engineering, hypertext and hypermedia design, information retrieval and data mining, graphic and interaction design, etc. Both Software Engineering and Web Engineering involve software development and its life cycle (including maintenance and evolution [47]), and so they share an important theoretical and practical corpus; however, Web Engineering must go beyond this.

One of the most important common problems of both disciplines is the need to describe and build systems using models as a way to solve some of the previously discussed problems. In this direction, Model-Driven Software Engineering-MDSE [6] was defined as a software development approach that focuses on the creation of domain models, highlighting abstract representations of the knowledge and activities in an application domain rather than in the algorithmic concepts. Models in the MDSE can be used to generate running applications in general by means of incremental transformations of these models that end up with source code generation. In this paper we discuss the main results of one specific sub-area of MDSE, the so called Model-Driven Web Engineering (MDWE) [65].

MDWE (as well as its "father" discipline Model-Driven Software Engineering-MDSE) [9] deals with those approaches which aim to generate running Web applications by transforming conceptual models onto models which are understood by computers (e.g., programs). The most important examples of this paradigm are Web application models, i.e., models which describe different aspects of the Web application; therefore, one of the many facets of MDWE has been the development of Web modeling languages, those languages that allow building Web application models. These languages are usually accompanied by a set of guidelines (methods) for their use and the tooling that support the previously-mentioned transformations.

The most important contributions of MDWE approaches to the more general area have been mainly the identification of modeling concerns which are specific to the Web domain, such

as navigation and user interface/interaction. Specifically, as we will show in this paper the techniques for dealing with the latter concern (interface/interaction) while present in every interactive application can be considered in some way the most important contribution of MDWE approaches to the software engineering field. MDWE approaches have shown how a clear separation of these concerns from the most "conventional" ones' favor modularity and hence evolution. Additionally, MDWE approaches brought to the surface several concepts which had been "lost" after the emergence of the Web, particularly those related with the hypermedia paradigm (from which the Web was inspired).

Additionally, and as described elsewhere [65], the MDWE discipline had to deal with a myriad of new issues as well as the adaptation of some of the existing solutions in the software engineering and in the MDSE field. Examples of this are: the relationships among the concepts in the Model-Driven Architecture (MDA) approach and the architectural variants in the Web, the need of different meta-models for specific concerns in Web applications development, the emergence of service, social and cloud-computing and its impact in the definition of models, the impact of multidisciplinary aspects in the development of models, the emergence of new kinds of stakeholders for the definition of requirements, etc..

Figure 1 summarizes in graphical form the containment relationships between the different disciplines that we have mentioned so far.

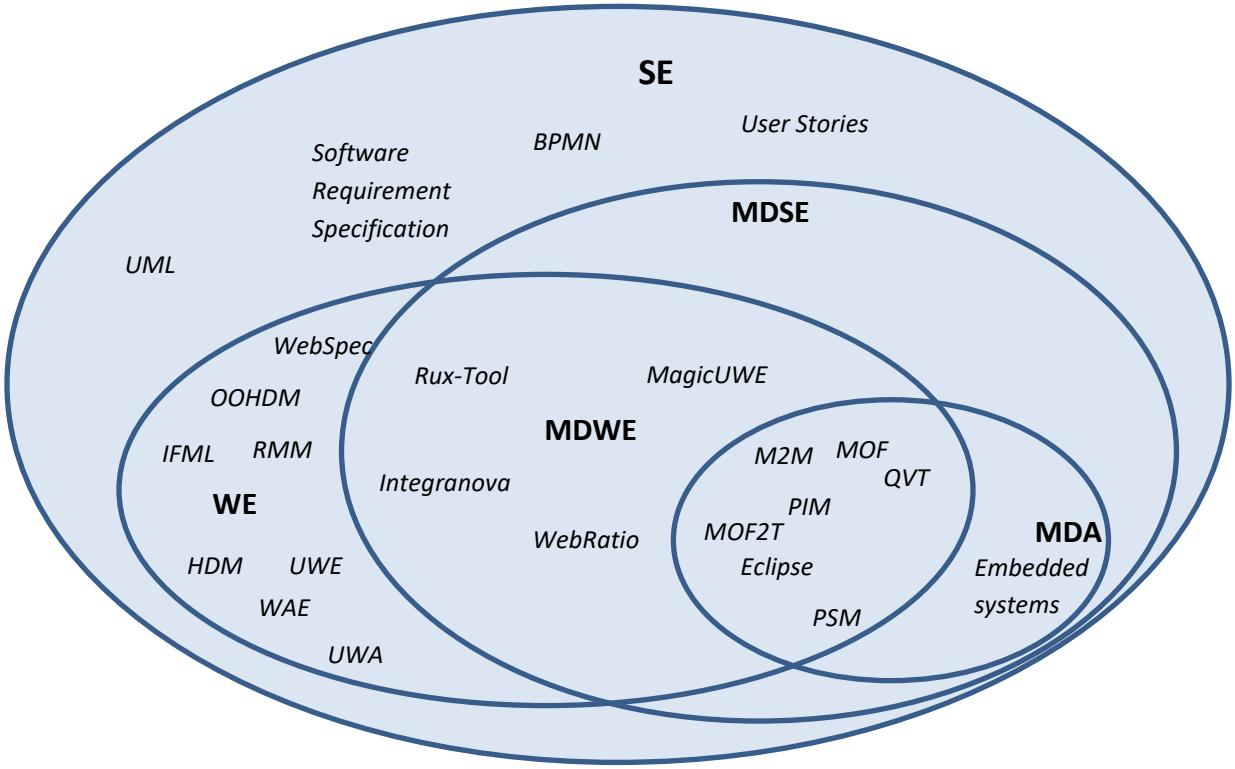

Figure 1: Relationships between Software Engineering (SE), Web Engineering (WE), Model-Driven Software Engineering (MDSE), Model-Driven Web Engineering (MDWE), and Model-Driven Architecture (MDA)

Since the early 90's many Web (at first Hypermedia) [37, 45, 94] modeling languages were developed and a lively community arose among those groups working in Web modeling issues. All these approaches tried to address the different features that Web applications support; to name a few: varying users' profiles, dynamic contents, rich user interaction, business processes, personalization and later mobility, social features, etc.

While the availability of a great number of different and heterogeneous approaches enriched the community understanding of the problem and helped to answer existing research questions, it also hindered the adoption of these technologies by developers. Fortunately in 2013, the Object Management Group (OMG) [128] adopted the Interaction Flow Modeling Language (IFML) [120] as the standard approach to describe the interaction features of Web (and other kind of interactive) applications, making it part of the set of software modeling standards like the Unified Modeling Language (UML) [134], the Business Process Modeling Notation (BPMN) [114], etc.

This paper describes the long path since the early modeling languages to the current state of the art, by focusing on which are the main achievements and what is still missing in the MDWE discipline. We briefly discuss the forces which drove this interesting evolution together with the most important milestones. The paper doesn't pretend to be a survey of existing methodologies and languages nor to describe any of those approaches with detail; rather than that, we try to explain the why's and how's of the different steps made by the community and the most outstanding products of the community knowledge. The reader can refer to the many excellent surveys on the field such as [107] or to the enormous amount of material describing specific approaches. A thorough literature review can be found in [32]. Also, and for the sake of conciseness we do not address quality issues which are of course important to the field. A detailed presentation of these issues can be found in [73]. There are other areas in the Web engineering field such as Web Services (specifically models for Web services) that have been addressed elsewhere [87] .

The paper is divided in two major parts. In the first part we briefly review the most important milestones to reach to the current state of the art. In Section 2 we detail the first modeling approaches in the field, and briefly discuss the taxonomy of forces that drove evolution. Section 3 discusses the first generation of specific Web modeling notations and methods. In Section 4 we dive in the language jungle of the first ten years of the new century. Section 5 shows how the community followed the road to a modeling standard. The second part is devoted to the achievements and problems to be solved. Section 6, lists the main achievements of the community beyond the main product (the IFML standard), while Section 7 discusses what is still missing in MDWE. We present our conclusions in Section 8.

2-Context and Forces

In the late 80's and early 90's we witnessed an explosion of research in hypermedia; the hypermedia paradigm, which gave birth to the Web, has been conceived many years before but it was the advent of the CD-ROM and multimedia in the 80´s which allowed some popularity of this way of accessing information.

Hypermedia (a multimedia extension to hypertext) has been defined as a paradigm for representing and accessing information [70]**.** Information is represented and stored in nodes, and nodes are related by links. The resulting graph can be then traversed by following these links from node to node. Nodes contain anchors for links which indicate those nodes' areas from where navigation can proceed. Nodes can also be the host of access structures such as indexes, which help to introduce hierarchical structures in the hypermedia graph.

Originally, hypermedia applications (i.e., specific hypermedia graphs in an application domain, e.g., a museum) were developed in a handcrafted way; however, it soon became apparent that this approach suffered the same kinds of problems that programs suffer (difficult to detect errors, complicated maintenance, etc.). To address these issues, similar to programming and database development, the hypermedia community developed several approaches to raise the level of abstraction in which these systems were built.

It is interesting to mention here that these early approaches were originated in groups with different backgrounds (e.g. databases, software engineering, human computer interaction and artificial intelligence). All of them recognized immediately the impact of interface issues, showing a first glance of the multidisciplinary aspects that were common a bit later in Web applications development. Additionally, these approaches emerged from groups that were used to the idea of models in their own disciplines and the differences between these approaches had to do with the nature of these models, e.g., entity/relationship, object-oriented, etc.

In this section, we briefly survey the main approaches for hypermedia design which were the roots of the first generation of Web design methods. We also describe the interleaving of forces which shaped all these methods.

2.1 Historical background: Early Hypermedia design approaches

Perhaps the first and most cited hypermedia design approach was the Hypermedia Design Model (HDM) published in 1993 [38], inspired by the E/R approach. The most important contribution of HDM was not the notation, which was not very expressive, but the ideas behind the approach, mainly the need to model content navigation using the primitives of node and link, and the need to separate design from implementation. The other relevant hypermedia design approach was the Relation Management Methodology (RMM) [45], which improved the ideas of HDM by enriching the entities with attributes and considering relationships as first-class citizens. Additionally, RMM introduced for the first time the idea that navigation and user interface were different concerns and that several interface issues

(e.g., how to limit the amount of information presented to the user in a screen) were critical and should be dealt with during design, in this case using the concept of slice of an entity. RMM was also the first approach to provide tool support for transforming a design model into a running hypermedia application [24]. The systematic analysis and design of relationships was also considered by other authors [110] which unfortunately was neither adopted by other approaches, nor considered in the design of Web applications in which links (the realization of relationships) are at the core of the applications' functionality.

2.2 Forces

There are different forces that have driven the evolution of Web engineering methods. They interleave in different ways and many times it is difficult to determine which of them caused a particular step in the evolution, e.g., a new method or notation, new primitives in a specific method or the identification of shortcomings in a particular approach. We can identify four main types forces (that may be surely sub-classified):

1. Device and network advances (e.g., mobile phones, touch screens, network connectivity).
2. Software technology evolution (e.g., new HTML versions, Web software frameworks like JSF[1] or AngularJS[2]).
3. User or market requirements (e.g., the need for personalization, support for business processes, volatile functionalities, or the creation of contents by end-users).
4. Better understanding of the design process and concerns (e.g., recognizing the need for separation of concerns, for more expressive notation, semantics, etc.).

If we observe how our field evolved in the last 20 years, it shows a similar pace regarding these forces than what happened in other fields (e.g., the object-oriented field). The Web was a research product which could exist because of the improvement of the internet and the needs of researchers to share documents; meanwhile software and methodological support for business processes was triggered by market and users' requirements. Sometimes, conceptual abstractions might have arisen even before the technology supported them reasonably: this can explain how many of the ideas of the mobile or context-aware Web (similar to the field of context-aware computing) such as [18, 33] which were previous to the smart phones, tablets or even Wi-Fi connectivity have not been fully implemented in concrete applications, as we will discuss later. However, similarly to what happened in other research areas (such as databases and programming), we can observe that, in general, methods and notations came after low level languages or tools support. And while in our field the first three forces continue to evolve, it seems that we reached a point of stability in the understanding of the design process.

[1] Java Server Faces, https://jcp.org/aboutJava/communityprocess/final/jsr344/index.html
[2] AngularJS, https://angularjs.org

3-First Generation of Model-Driven Web Engineering approaches

During the mid-90's hypermedia design approaches evolved into Web design approaches; this was caused by the rapid growth of the Web platform and by the understanding that Web applications were much more dynamic than the hypermedia applications they were trying to support, such as kiosks, or CD-ROM based systems.

The Object-Oriented Hypermedia Design Method (OOHDM) [93] formally introduced the idea that domain, navigation and interface models should be separated (although related) since they represented different concerns. Additionally, it introduced the idea that objects of the same domain model could be navigated differently (by the definition of different navigation models) and that different interaction and interface facilities could be built for the same navigation model, e.g., to support varied presentation devices. OOHDM also introduced a notation for navigational contexts [93] and considered access structures (such as menu and indexes) as first-class citizens at the same level of nodes and links early introduced by HDM. Interface and interaction issues beyond navigation were specified using Abstract Data Views (ADVs) and ADV charts [19]. Though originally based on the Object Management Technique (OMT) [88], it moved to UML when UML arose. However, the notation was in part proprietary.

Finally, together with the new notation, OOHDM comprised a set of process heuristics ranging from the typical waterfall to the iterative and incremental styles [52]. However, like other model-driven approaches, the process was mostly thought to be used in a waterfall way.

The main contribution of OOHDM was not its object-oriented nature (which of course served to cleanly describe relationships between modeling concerns) but the idea that a Web application was a (navigational) view built on top of a domain model. This idea was used subsequently by all Web design approaches. Further separation of interface and interaction issues (in different models) was an issue under debate for years in the community.

Another contribution of OOHDM was the introduction of the Web patterns concept [84] as a means to reuse navigation and interface designs. The idea that Web models comprise a domain, a navigational and an interaction/interface model, helped to understand that Web patterns might also exist in the three different design concerns. Therefore, it was a direct consequence of a modeling strategy. Many Web patterns were discovered by different groups [39, 67, 86]. Some of these patterns were later incorporated in different design notations either as primitives or model annotations.

One of the main drawbacks of OOHDM was that it had poor support for automatic (or semi-automatic) generation of running Web applications, although some support was later built in the context of the Model-Driven Architecture (MDA) style [91].

The Web Applications Extension (WAE) [16] extended UML with some stereotypes to represent client and server pages, forms, client script objects, links, etc. While WAE recognized the software engineering issues behind Web applications, it did not address the major concerns of these applications clearly. One of the main reasons of this drawback might

have been the fact that WAE was thought with implementation technologies (such as ASP or JSP) in mind. As a consequence, the impact of WAE in the literature was minor and there was no movement on this approach after its second version (WAE 2) in 2002 [17] .

Different to WAE, the Unified Web Engineering (UWE) [42] approach represented a major step forward. It was also object-oriented but its notation was fully based on UML. While still dividing the design process in three main activities: domain or application modeling, navigation design, and presentation design, UWE extended UML in a conservative way (not modifying the meta-model but using stereotypes) and used the whole UML notation starting from requirements, with an extension to the use cases notation. The next major contribution of UWE was its support for deriving a Web application from design models via model-to-model and model-to-code transformations [62]. Additionally, UWE introduced the idea of modeling the user in order to support different strategies of application adaptation [50].

Though originally published in 2000, we consider the Web Modeling Language (WebML) [14] as a first generation language since it also helped to shape the subsequent approaches. It also separated data modeling from navigation design (called hypertext design in WebML) although interface and interaction models were not considered in its first version. WebML privileged entity-relationship modeling and later moved to UML models. While its hypertext notation was proprietary, it was simple and in many ways it was more expressive than OOHDM and UWE, incorporating some features that the previous approaches relegated to presentation models. WebML's tool support, WebRatio [138] (which was also the name of the spin-off company created to support the language) was easy to use, powerful and well-advertised. Additionally, the WebML support group was very active and therefore many extensions were built in the next few years (See Section 4). Moreover, as we will discuss later, WebML evolved seamlessly into the IFML standard.

Summarizing, the first group of modeling approaches shaped the modeling space of Web applications by defining the main concerns and the concepts that needed to be modeled in each one. These approaches also showed that model-driven development was feasible and provided tools (conceptual and technological) to support this process. Figure 2 shows a timeline of these contributions.

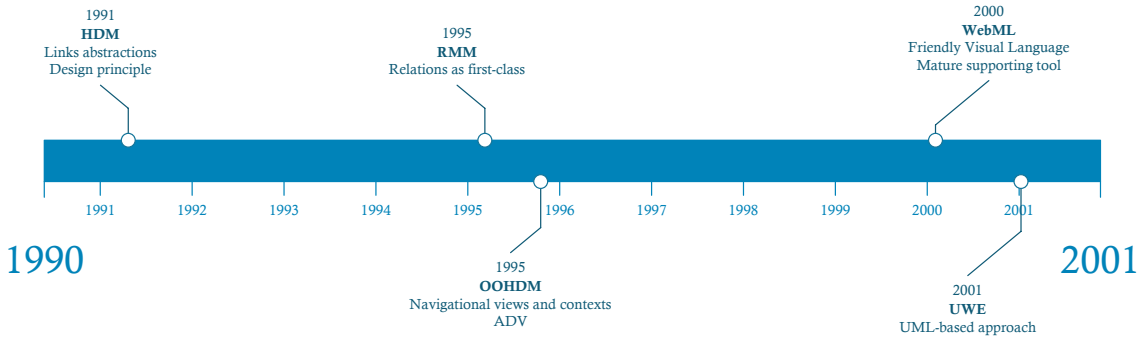

Figure 2: Contributions of the first generation of MDWE approaches

4-Modeling Languages in the New Century

From 2000 we witnessed an explosion of methods and modeling approaches for Web application development as well as extensions to the abovementioned approaches. Many of them were motivated by some combination of the forces presented in Section 2; others were just different notations with the same features of the methods presented in the 90's. Here, we just enumerate some of them together with their motivations. A very thorough analysis of the complete literature can be found in [107].
For the sake of clarity, we separate the methods in two groups: those which introduced new features motivated by either new users' requirements or technological advances (e.g., personalization, support for semantic Web applications, business processes, etc.), and those which introduced better support for the life cycle (e.g., requirements specification, better separation of concerns, improved meta-modeling and transformations, etc.). Of course it might happen that some methods appear in the two sub-sections. Again, the intention is to give meaningful examples rather than to mention all approaches.

4.1 Methods introducing new features

In the early 2000's several methods introduced features for dealing with personalization and some kind of context-awareness. OOHDM [83], WebML [13], and UWE [4] were slightly improved in that direction. From the new methods, an outstanding role was played by the Ubiquitous Web Applications approach (UWA) [33] since it claimed to consider ubiquity in all design dimensions. The work on UWA was remarkable and many interesting extensions were developed, such as UWA+ [6] and UWAT+ [26, 28]. As mentioned in Section 2.2, this is an interesting case where computing abstractions and understanding of a technology came much before the time in which this technology was widely available. This mismatch might be the cause why many of the UWA concepts were not applied immediately or even re-invented later. A complete and thorough survey on methods' support for ubiquity can be found in [95].

Another interesting achievement for methods was the introduction of Semantic Web features (such as an extensive use of ontologies and the use of RDF[3] to represent models). The more representative approaches were Hera [105], the Semantic Hypermedia Design Method (SHDM, an evolution of OOHDM) [96] and the Web Semantics Design Method (WSDM, an evolution of the Web Site Design Method) [97].

The problem of designing complex workflows (typical in business processes) was discussed very early in the MDWE community and therefore support for representing business

[3] Resource Description Framework (RDF), https://www.w3.org/RDF/

processes was soon available in UWE [48], OOHDM [92] and WebML [10]; UWA+ [6] and UWAT+ [6, 28] extended the UWA approach with better support for business processes.

To illustrate the expressive power of MDWE approaches and the maturity of the community, almost each new technological possibility was soon supported by modeling approaches; we illustrate here the case of RIA [29] and Mashups [22].

Many Web modeling approaches such as OOHDM [99], WebML [7] and OOH [43] improved their modelling primitives to specify interaction features by embracing Rich Internet Applications (RIA) features, those Web applications with advanced interaction features. The RUX method [80] provided support for improving both WebML and UWE with RIA features.

In the case of Mashups, almost when the first programming approaches (such as Yahoo's) for Mashups appeared, researchers in the MDWE field devised new modeling approaches [21, 55, 81]. A thorough analysis of this topic is presented in [22]. Figure 3, summarizes the contributions of the previously mentioned approaches.

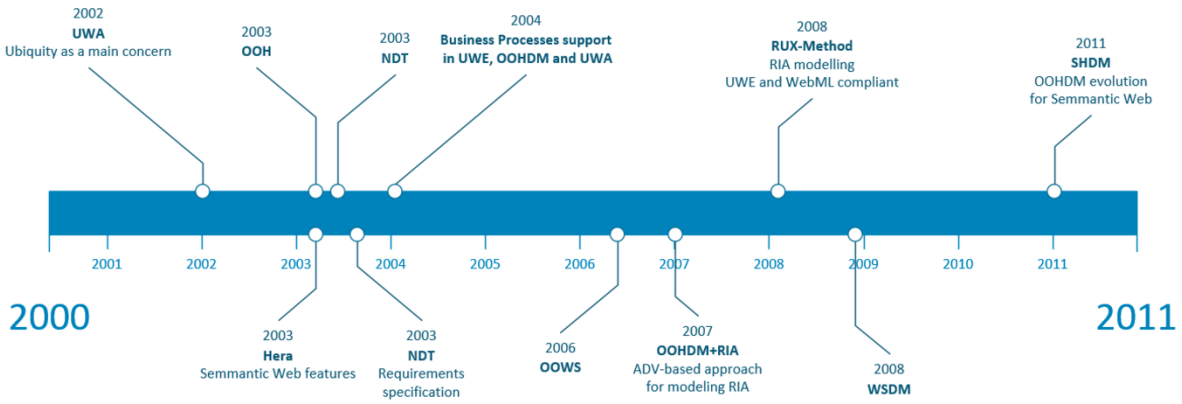

Figure 3: Timeline describing contributions in the new century

4.2 Improving modeling and design issues

During the last decade, the MDWE community gained in understanding of the methods' weaknesses regarding life cycle coverage. Many methods also improved their formal basis, by incorporating concepts and techniques developed in the MDSE field.

4.2.1 Life cycle coverage

While most methods supported the modeling of the different Web application concerns and the automatic (or semi-automatic) generation of running applications, they didn't usually cover requirement elicitation and specification. In some cases, the method assumed that requirements were captured and represented using some existing technique (e.g., use cases or user stories) but soon it became clear that certain requirements (e.g., navigation or interface) could not be easily represented in this way.
As previously commented, UWE defined a specific UML profile [136] for improving use case specifications. OOHDM incorporated User Interaction Diagrams (UIDs) [106], a state machine like notation for navigation requirements. OOWS defined a model-driven approach to map Web requirements onto design models [101]. Navigational Development Technique (NDT) [30], meanwhile, was originally devised to deal with Web applications requirements, and it later evolved into a full-fledged approach. WebSpec [58] provided support for specifying navigation and interaction requirements and for mapping them onto WebML models. Nice surveys of approaches for specifying requirements in Web applications can be found in [32, 103].

In [63] the authors presented WebSA (Web Software Architecture), a novel approach to address architectural issues in the model-driven process, "merging" them with the application's functional models, and applying the corresponding transformations to generate the running application.

Testing (including early testing) and maintenance were not addressed in the first generation of MDWE proposals but were later discussed in several approaches [35, 57, 69]; a thorough survey can be found in [20].

Additionally, many MDWE approaches attacked different aspects of the realization of Web applications; a remarkable aspect that was introduced during this new century is the specification of Web Services (WS) in the context of MDWE (beyond the specification of WS, their choreographies, etc.). WebML improved its notation to support Web Services [8]; similarly, OOWS incorporated WS in its design repertoire.

4.2.2 Improving conceptual support for MDWE

The continuous advance on model-driven engineering technology contributed with the improvement of MDWE approaches. Perhaps the most important of these advances was the

emergence of the Model Driven Architecture (MDA) standard [123] defined in 2001 by the OMG. In MDA, system functionality is defined using a Platform Independent Model (PIM) described using an appropriate Domain-Specific Language (DSL), for example UML. The PIM is then transformed into a Platform Specific Model (PSM) by means of model transformations to finally generate the source code of the application. The PSM might be different for different implementation platforms (e.g., Java, .Net, etc.). PIMs and PSMs are defined using modeling languages that are described using a corresponding meta-model and transformations are (or can be) described also using a transformation meta-model.

MDA is supported by a set of additional OMG standards, including the Meta-Object Facility (MOF) standard [129] for metamodels definition, the MOF Query/View/Transformation (QVT) language [131] for model-to-model (M2M) transformations, and the MOF Model to Text Transformation (MOFM2T or MOF2T) language [125] for model-to-text transformations (i.e., code generation).

Though the MDA approach was never considered mainstream in the industry, its contributions to the understanding of the model-driven software engineering process were outstanding.

UWE was perhaps the first proposal to describe the whole approach, including the process, with a metamodel [136] using MOF, and to base the generation of running applications by using model transformations [49], using a "pure" translationist approach; this allowed to constantly improve application generation adding new target platforms.

OO-H [40] and OOWS [75] are two new evolutions of the formal object-oriented method (OO-Method [76]) which arises to support automatic generation of Web applications. OOWS introduced as specific contribution the implementation of a PIM compliant with the MDA standard. The PIM allows OOWS models to be transformed into running code and the process is supported by the commercial tool OlivaNova (provided by Integranova [119]).

Many research groups proposed metamodels for WebML [90] which meant a step forward that was later profited to create the IFML standard.

The UWA approach also assisted to evolutions towards MDA and automatic generation of application prototypes [5, 27] . In particular, the UWA-based MDWE approach presented in [5] adopts MOF for the definition of the PIM and PSM metamodels used by the method, the ATLAS Transformation Language (ATL) [113] for model-to-model transformations, and Xpand [124] for code generation from models.

NDT defined a metamodel for requirements [31], which was interoperable with the UWE approach. All these efforts contributed together to pave the way to a standard language.

Regarding the design process itself, some approaches recognized the need to improve separation of concerns in their notations to make designs more evolvable. While separation of design concerns had been implicitly recognized by all design approaches and also by component-based approaches like the WebComposition Process Model [36], the addressed concerns remained the "canonic" ones: content, navigation, interface, and interaction. In some cases, additional concerns were business processes and personalization. However,

functional applications concerns were poorly considered by MDE, nor there was support for this in MDWE approaches.
To address the lack of design support, OOHDM used advanced separation of concerns to deal with volatile requirements [100]. Hera [12] used aspect-oriented concepts to deal with crosscutting concerns. An aspect-oriented WebML was also presented in [89]. Unfortunately, the momentum of aspect-oriented technology ended some years ago and these ideas were not further explored beyond these academic works.

4.3 Towards a standard modeling language

The negative aspect of having a myriad of design approaches and notations was very early recognized by the community. While it was natural that different actors proposed their own approaches according to their understanding of the problem, the problems this caused to potential users, educators and students learning MDWE became clear soon (e.g., how to choose the design language for a project). Although the concepts in most approaches were more or less similar, the differences in notation made difficult to organize learning materials. The MDWE community organized a series of workshops in which this problem was discussed: the International Workshop on Web Oriented Software Techniques (IWWOST) [112], the Workshop on Model-Driven Web Engineering [111] and later a network of institutions called MDWEnet [104] were created to, at least in part, discuss this issue.
There were mainly three ways to solve this problem. The first way was to achieve an agreement among different methods and come out with a new one with the "best" of each of them. This was the approach followed by the "three amigos" (Booch, Jacobson and Rumbaugh) to create UML in the 90´s [46]. This approach failed, perhaps because there were much more than three modeling approaches and because each involved scientist had a different research target (neither Booch nor Jacobson and Rumbaugh were scientists). The second way was to make all approaches interoperate in a way or another, allowing easy portability from one design model to another or to bridge them easily. Many researchers tried to define "common" meta-models and there were very interesting results. The most thorough and systematic approach was the Web Engineering Interoperability (WEI) initiative which aimed to provide easy exchange of models among existing tools but also to address the incorporation of new concerns to existing methods. A very complete analysis of this problems and the proposed solution can be found in [66].
Finally, the third possible way was to create a standard, either by the emergence of a brand new approach or through the evolution of one of the existing ones. In 2013 when the OMG announced that it has adopted IFML as the standard language for describing interaction aspects of Web (and other kinds of) applications, we reached a new milestone, not necessarily the end of the road, but definitively a step forward. In the next section we discuss our achievements as a community, emphasizing on the IFML and its impact.

5-MDSE and MDWE Impact on Industry

MDWE approaches aim at providing tools and techniques for simplifying applications design, development and evolution. Tools play an important role in MDWE adoption since traditionally software engineers are used to rely on the use of CASE tools for boosting their productivity.

Diverse works have studied the impact of MDSE in the industry highlighting the pros and cons of its usage. Model Driven Engineering (MDE) has been widely adopted in diverse industry domains. According to [109], software companies that successfully adopted MDSE based their development on domain-specific models rather than on general purpose languages such as UML. Surprisingly, the real benefit gained from code generation is hindered by the training effort on MDSE techniques. However, the real benefit that companies find in MDSE usage is the clear definition of the software architecture. The process and the method behind MDSE adoption in a company are not always properly supported by a tool; indeed, the immaturity, complexity, and usability are the major tool's barriers [108].

MDE for Web applications has been supported by both proprietary and open source solutions. The Eclipse Modeling Tools suite is the leading open source technology for implementing MDE used by IBM and Oracle. Other proprietary technologies are available with the same purpose such as JetBrains Meta Programming System [126].

Web applications development can be performed using tools which rely on proprietary approaches and languages such as Mendix [116] and OutSystem [130]. These solutions support agile development processes comprising common phases such as requirement gathering, requirement modelling, application generation (access to models transformation tools are often not available to the designer), deployment and monitoring. There are also UML-based tools such as Visual Paradigm [137] and IBM Rational Suite [117] that exhaustively support the UML standards (i.e., Use Case, Class Diagrams, and Deployment Diagrams) and provide code generation from models.

From the academic research effort, a lot of tools have been released in the last decades; many of them are still active projects: NDT Suite [127], Integranova [119], MagicUWE [135], and WebRatio [138]. To our knowledge there is still no survey presenting a state of art of these tools.

The impact of MDWE approaches on the industry has not been studied properly so far. Although some researches have thrown light on advantages and disadvantages of MDE implementations [3, 64, 108], there is not rigorous evidence about the benefits of MDWE approaches usage in real development contexts.

6-Achievements

In the last 25 years our community has got an enormous number of scientific and practical results, including the dissemination of MDWE technologies in dozens of universities and

thousands of students and professionals of the Web field. Even though MDWE technology has not been widely adopted yet, its advances have influenced many other fields. We next describe the main achievements beginning with the IFML.

6.1 The IFML standard

The evolution from WebML to IFML is very interesting. The group in Politecnico di Milano and the WebRatio spin-off identified that the problem of formal specification of user interaction and interface was shared by every interactive application (not just Web) and specifically those using the Model-View-Controller (MVC) architectural design pattern [53, 56]. For this reason, they decided to generalize WebML in such a way that it was not Web specific but targeted to a broader set of applications (e.g. Desktop, Web and Mobile ones). Additionally, and since UML (and BPMN) covered practically all aspects of software development except the front-end aspects, they decided to narrow the scope of WebML in such a way that IFML "only" focuses on these aspects, while the rest of the approaches in some way pretend to cover the whole life-cycle. Instead of trying to adapt or extend UML to the user interaction realm (as it has been done by UWE as described previously in this paper), the IFML team decided to reuse the proprietary style of WebML (which had been successful) and began the hard road to (technically and surely politically) convince the OMG that this was a better approach.

Beyond the outstanding fact that the OMG adopted IFML as a standard, the language itself has important merits (without delving into the specific notation itself which will surely evolve and improve with its use). IFML inherits the simplicity and expressivity of its "parent" WebML, improves it by removing notations which do not belong to the interaction concerns, such as business process issues, and adds user interface events as first-class citizens in the notation. Additionally, and with the aim of making IFML fit into the UML universe, the language supports the same kind of extension mechanisms that UML already possesses (e.g., stereotypes, tagged values, etc.) and it is itself described by a clearly defined meta-model. Of course, the impact of these features will be seen in practice, but the IFML team has already shown some examples to illustrate the extensibility features.

However, from our point of view, the most relevant merit of IFML is the fact that it brings to light a shocking fact, which had not been made explicit in this way before. After almost 30 years since the MVC appeared as a formal concept [51], developers building interactive applications using this architectural pattern only modeled the Model features, while the View and Controller components were not considered for its implementation-independent design (except of course for MDWE approaches) and had to be addressed at the code level. Now, the OMG offers a notation which allows specifying View and Controller features (and their relationships between each other and with the Model) using a simple notation, while UML continues to be the standard to model the Model component. Standards for specifying lay-out or lower level detail presentation aspects are still to come.

6.2 A community and collective knowledge

The MDWE community evolved rapidly. After the first workshop on Web Engineering in 1998 [132] held in the context of the WWW conference [122], the IWWOST series [112], held since 2001, targeted more specifically model-based Web development techniques. Most researchers in this area gathered in the International Conference on Web Engineering (ICWE) [118] held since 2002. The more specific MDWE workshop was held since 2005 [139] and the MDWENet initiative [104] was launched soon after that in 2007.

These collective efforts involved dozens of institutions and researchers both from academia and industry. As a result, the community acquired an important understanding of the problems involved in the model-driven development of Web applications and a formidable corpus of knowledge was produced and published in different venues, including journals and conferences. As mentioned previously, this knowledge was disseminated in courses, seminars and books [11, 14, 22, 85] and provides an excellent background both for new scientists and practitioners, not only those who work using model-driven techniques, but also for those developing Web applications in the "traditional" way.

6.3 Technical recognition

The prevalence of the Web as a development and deployment platform had also impact on the research topics of the software engineering community. Before the Web, it was rare to find many papers on software usability or accessibility, which of course were already important problems, as well as on other topics, such as information retrieval, which have become increasingly relevant today. The MDWE community not only brought to the scene many new problems such as navigation and user interaction modeling, but also those related with quality issues. Several researchers had already pointed out the importance of dealing with these problems in a high-level, implementation independent way [54]. However, the important number of new problems posed by the Web made this research increasingly critical since the number of users exploded in orders of magnitude. This fact not only made these subjects more popular, but also motivated new research. Many researchers in the MDWE community proposed modifications to existing user interaction modeling approaches to face the new problems [102]. Similarly, researchers in the user interaction field modified or upgraded their proposals to solve these issues [77, 78].

Nowadays, not only the OMG recognizes the importance of modeling user interaction and interface, which can be read in its presentation of the IFML (read for example the overview and benefits in [120]), but most software engineering related conferences and journals have incorporated these issues into the list of topics of interest. Dozens of workshops related with MDWE research topics have been held in important conferences such as International Conference on Software Engineering (ICSE) [121], International Conference on Model Driven Engineering Languages and Systems (Models) [133], and Conference on Advanced Information Systems Engineering (CAiSE) [115], and papers on the previously mentioned

approaches have been published in all the top journals of the field. Additionally, most of the seminal research papers of the MDWE community have had an important impact in research performed by other communities, such as the User interaction or User interface communities, the Multimedia community, the Business Process community, among others. As an example of this impact, it is worth mentioning the many research works that has been done and the several approaches that have been proposed for the reverse engineering and the evolution of existing Web applications adopting MDWE methods, models, and techniques. A survey on this research can be found in [47].

7-What Is Missing

While the technical achievements of the MDWE community have been outstanding, there are still many issues to be solved before MDWE becomes mainstream as a strategy for building real Web applications. Many of these issues are shared with the large MDE community. For the sake of comprehension, we explain these shared problems detailing the specificities of the Web engineering field.

7.1. Adoption

The problem of the adoption of model-driven development (MDD) has been widely discussed in the general software domain; specifically, many studies show that the problem is more than technical and involves also cultural, organizational, and other kind of issues [109]. In the Web engineering field, the situation is much worse indeed. While UML has already more than 20 years of life and has been practically the unique object-oriented modeling language since its birth, the myriad of Web modeling languages (briefly presented in this paper) have in some way hindered adoption. Despite the fact that WebML has been the most popular MDWE approach being used in the industry, IFML is far from being mainstream. Indeed, this standard still needs to gain a supporting community. Additionally, since the Web development activity is much more interdisciplinary than general software development, there are a huge number of developers who barely program, and of course do not model. Despite the empirical evidence presented in several works about how software quality and team productivity improves when using MDWE approaches than code-based ones [44, 61, 74], developers ignore the benefits of adopting an MDWE approach. Also, the huge variety of programming languages and frameworks make the adoption of MDWE approaches difficult since developers tend to think about modeling techniques as to an obstacle for creativity, while in fact it is the contrary. One can reasonably argue that the lack of good tool support and the absence of user communities have an important place in this problem: some of the problems listed in the next sub-sections have also slowed down the adoption of MDWE techniques.

7.2 Better support for non-functional requirements

While there has been considerable research on modeling functional requirements and transforming them into design models and running applications, the transformation of Non-Functional Requirements (NFRs) to code has been less explored in the literature. These kind of requirements are particularly critical in Web applications, and some of them (like usability and accessibility) appear once and again as weaknesses of even small Web sites. In a recent research paper [2] the authors have clearly analyzed this deficit of model-driven approaches in the general software field. The example they use (scalability) and the different solutions (basically replication) they propose apply perfectly to show the problems in our field.

Most MDWE approaches are rigid regarding the details or specificities of the architecture which will support the running application. At most, these details might be specified as after thoughts during the last code generation process.

It should not be surprising that the two non-functional requirements that were most systematically addressed in MDWE approaches are usability and accessibility, precisely because they express themselves during user interaction. Accessibility was dealt with during model construction in WSDM [79] and usability was also considered in OOWS [68]. IFML itself provides some hint for improving usability, but specific usability or accessibility features cannot be yet specified.

7.3 Agility

The very nature of Web applications, with very often ill-defined requirements that are themselves very volatile, make agile approaches [59] the optimal choice to support the development process. Agile methods have quickly developed in the last 15 years and today one might say that they are pretty mature, well used in industry and with reasonable productivity.

Meanwhile, model-driven approaches, in general on the Web field, tend to be “monolithic” and thought more for a waterfall rather than agile, iterative style. Even though practically all of them (from the first version of OOHDM to IFML) claim to support some kind of iterative, incremental or agile styles, there is another “cultural” problem: agile approaches usually assume (and so do developers) that the development process is more code-based than model-based. Perhaps this is one of the biggest obstacles in the adoption of MDWE approaches. Developers are more used to write small components to address some requirement (usually expressed informally in a user story) and then iterate to the next requirement. Code refactoring allows keeping this code more or less “clean”.

It is absolutely obvious that the same approach (e.g., in a Scrum [15] context) could be performed using MDWE approaches (and of course UML in the general case), but this is not the state of practice today.

We have been working to demonstrate different ways to introduce agility in a MDWE schema with rather good results. In [57] we presented a test-driven approach (TDD) which combined interaction tests (in the style of Selenium) with a model-driven schema to combine the agile

TDD style with the use of models instead of code. In [82] we show how to use partial mockups to derive Web design models (in WebML). In this case, mockups were derived from user stories, and each development cycle dealt with a single requirement as in most agile methods but relying on automatic program derivation. In both cases the experiments with users (developers) show acceptance and good performance.
However, being this topic more cultural or organizational than purely technical, the way to solve it from the MDWE field might be tricky but need to be addressed soon if we expect some kind of success.

7.4 Support for end-users

A very interesting and thorough survey on modern Web development practices [71] shows that an important portion of Web sites are or have been developed by end-users using tools such as Content Management Systems (CMSs) such as Drupal[4] or WordPress[5]. These users are grouped in communities sharing plugins, themes, designs, etc. At the same time, many other users communities have emerged in the context of the so called Web augmentation (WA) metaphor [25], where final users adapt their preferred applications by developing scripts (usually JavaScript ones) that run on the browser. CMSs or WA communities are very active and end-users crowdsource their projects to the community, benefiting from the results. A very simple but effective reuse and sharing schema of page designs and augmentation scripts has emerged in these communities.
However, and as it is cleverly indicated in [71], the MDWE community has ignored this phenomenon. End-users often do not have the technical skills to apply the concepts underlying the MDWE approaches; they might not even have the resources to use any of them. And certainly this divorce has an important impact both in the spreading of MDWE methods and also because many of the techniques that have been used in the context of the MDWE community could be useful for end-users and at the same time end-users could enrich the MDWE community.
We can mention two proposals coming from the research community to bridge this gap which fall in the end-user development style; they both use extensively the same concepts that have been developed by the MDWE community such as metamodels. The authors of [72] propose a strategy to improve design by example in WordPress by allowing users to reuse WordPress themes with a finer granularity style, by combining parts of them into new ones. CrowdMock [34] allows end-users to specify their needs to the WA community by sketching a mockup with the intended adaptation or augmentation of the intended site.

[4] Drupal - https://www.drupal.org/
[5] WordPress - https://en.wordpress.com/

7.5 Better coverage of Web engineering technologies

While, as shown in Section 4, most MDWE methodologies adapted and improved their modeling primitives to the different technologies that arose in the century, all extensions were in some ways "ad-hoc" and the impulse towards standardization did not cover all of these technologies. To make some pending topics, we should mention that Web Engineering conferences and workshops [112, 118] promote the innovation in new research lines such as Web of things [41], Semantic Web [98, 105], and Social Web applications [1] (including of course social networks). IFML is still in its infancy and it has not been demonstrated if these kinds of Web software can be developed using the MDWE approach. Further research on this topic is needed.

7.6 Improvement of architectural and implementation issues

The integration of different kinds of (heterogeneous) models has not received the needed attention yet. Though this is not only a problem of MDWE approaches (since the same happens in most OMG models), there is still space for improvement regarding different ways to deal with models uniformly. This implies expressing this integration at the meta-model level (e.g., BPMN and IFML), but also at the concrete application level where different points of view of the same domain might be expressed. At the same time, and as mention in Section 7.1 and 7.2, adoption is hindered by the immaturity of tools. Specifically, most tools (not only WebRatio) are very rigid in the transformation to code phase, leaving no space for expressing the different variations that different implementation settings might have. Architectural variations are also ignored, with the exception of WebSA [53].

7.7 Evaluation of MDWE approaches

As in other areas of MDSE, there has been scarce empirical research about the impact of MDWE on developers' productivity and application quality. While this evaluation might not be the key for encouraging adoption, it represents a pending issue in this discipline. However, we can cite two or three important research works "measuring" systematically the effect of MDWE in different aspects of Web applications development [60, 61]. Another study regarding MDD performance against code-based was performed at [74] and it focused on verifying some of the most cited benefits of MDD. By means of empirical evaluations with students, they analyzed quality, effort, productivity and satisfaction aspects on MDD and code-based projects. The research throws as result that MDD does not always yield better results than a traditional method but also highlighted that MDD provides better accuracy when developing functional requirements.

8-Concluding Remarks

Model-Driven Web Engineering (MDWE) arose in the late 90's to address the challenge of developing complex Web applications by focusing on models instead of code. Since the early days of MDWE the Web has continued its evolution and in the same way users became more demanding; as a result, Web engineers have adapted their toolboxes to face these new demands. MDWE has therefore evolved with new and fresh approaches. In these 25 years a lively community has emerged and this community has contributed with the advance of the general software engineering field. In this paper we presented a rapid overview of the road followed by the community since the early, primitive modeling languages, to the development of the new IFML standard. We showed that during this period MDWE researchers built support for different Web technologies, such as RIA, Semantic Web, Mobile Web and also improved application life cycle coverage by addressing requirement specification, testing and maintenance. The emergence of IFML is certainly not the end of the road but an important milestone; as we also discussed in this paper, there is still a lot of work to be done in order to disseminate these ideas to assure that modern Web applications are developed faster, safer, and with less errors and higher quality.

9-References

1. Abel, F. et al.: Leveraging User Modeling on the Social Web with Linked Data. In: Web Engineering - 12th International Conference, {ICWE} 2012, Berlin, Germany, July 23-27, 2012. Proceedings. pp. 378–385 (2012) DOI:10.1007/978-3-642-31753-8_31.

2. Ameller, D. et al.: Dealing with Non-Functional Requirements in Model-Driven Development. In: RE 2010, 18th IEEE International Requirements Engineering Conference, Sydney, New South Wales, Australia, September 27 - October 1, 2010. pp. 189–198 (2010) DOI:10.1109/RE.2010.32.

3. Baker, P. et al.: Model-Driven Engineering in a Large Industrial Context --- Motorola Case Study. In: Briand, L. and Williams, C. (eds.) Model Driven Engineering Languages and Systems: 8th International Conference, MoDELS 2005, Montego Bay, Jamaica, October 2-7, 2005. Proceedings. pp. 476–491 Springer Berlin Heidelberg, Berlin, Heidelberg (2005) DOI:10.1007/11557432_36.

4. Baumeister, H. et al.: Modelling adaptivity with aspects. In: Lecture Notes in Computer Science. pp. 406–416 (2005) DOI:10.1007/11531371_53.

5. Bernardi, M.L. et al.: A model-driven approach for the fast prototyping of web applications. In: Proceedings - 13th IEEE International Symposium on Web Systems Evolution, WSE 2011. pp. 65–74 (2011) DOI:10.1109/WSE.2011.6081821.

6. Bochicchio, M.A., Longo, A.: UWA+: bridging Web systems design and Business process modeling. In: Hypermedia Development & Web Engineering Principles and Techniques: Put them in use International Workshop on Web Engineering. (2004).

7. Bozzon, A. et al.: Conceptual modeling and code generation for rich internet applications.

In: Proceedings of the 6th international conference on Web engineering ICWE 06. p. 353 (2006) DOI:10.1145/1145581.1145649.

8. Brambilla, M. et al.: Declarative specification of Web applications exploiting Web services and workflows. In: Proceedings of the 2004 ACM SIGMOD international conference on Management of data - SIGMOD '04. p. 909 ACM Press, New York, New York, USA (2004) DOI:10.1145/1007568.1007688.

9. Brambilla, M. et al.: Model-Driven Software Engineering in Practice. Morgan & Claypool Publishers (2012) DOI:10.2200/S00441ED1V01Y201208SWE001.

10. Brambilla, M. et al.: Process modeling in Web applications. ACM Trans. Softw. Eng. Methodol. 15, 4, 360–409 (2006) DOI:10.1145/1178625.1178627.

11. Brambilla, M., Fraternali, P.: Interaction Flow Modeling Language: Model-Driven UI Engineering of Web and Mobile Apps with IFML. Morgan Kaufmann (2014).

12. Casteleyn, S. et al.: Aspect-oriented adaptation specification in web information systems: a semantics-based approach. New Rev. Hypermedia Multimed. 15, 1, 39–71 (2009) DOI:10.1080/13614560902818297.

13. Ceri, S. et al.: An Approach to User-Behavior-Aware Web Applications. In: International Conference on Web Engineering (ICWE2005). pp. 417–428 (2005) DOI:10.1007/11531371_54.

14. Ceri, S. et al.: Designing Data-Intensive Web Applications. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2002).

15. Cohn, M.: Succeeding with Agile: Software Development Using Scrum. Addison-Wesley Professional (2009).

16. Conallen, J.: Building Web Applications with UML. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2000).

17. Conallen, J.: Building Web Applications with UML Second Edition. Addison-Wesley (2003).

18. Consortium Uwa: Ubiquitous Web Applications. In: In Proc. of the eBusiness and eWork Conference 2002. (2002).

19. Cowan, D.D., Lucena, C.J.P.: Abstract data views: an interface specification concept to enhance design for reuse. IEEE Trans. Softw. Eng. 21, 3, 229–243 (1995) DOI:10.1109/32.372150.

20. Cuaresma, M.J.E. et al.: An overview on test generation from functional requirements. J. Syst. Softw. 84, 8, 1379–1393 (2011) DOI:10.1016/j.jss.2011.03.051.

21. Daniel, F., Matera, M.: Mashing Up Context-Aware Web Applications: A Component-Based Development Approach. In: Web Information Systems Engineering - WISE 2008. pp. 250–263 Springer Berlin Heidelberg, Berlin, Heidelberg (2008) DOI:10.1007/978-3-540-85481-4_20.

22. Daniel, F., Matera, M.: Mashups - Concepts, Models and Architectures. Springer (2014) DOI:10.1007/978-3-642-55049-2.

23. Deshpande, Y. et al.: Web Engineering. J. Web Eng. 1, 1, 3–17 (2002).

24. Diaz, A., Isakowitz, T.: RMCase: Computer-aided support for hypermedia design and development. In: Hypermedia Design. pp. 3–15 Springer (1996).

25. Díaz, O., Arellano, C.: The Augmented Web: Rationales, Opportunities, and Challenges on Browser-Side Transcoding. TWEB. 9, 2, 8 (2015) DOI:10.1145/2735633.

26. Distante, D. et al.: A comprehensive design model for integrating business processes in web applications. Int. J. Web Eng. Technol. 3, 1, 43–72 (2007) DOI:10.1504/IJWET.2007.011527.

27. Distante, D. et al.: Model-Driven Development of Web Applications with UWA, MVC and JavaServer Faces. In: Baresi, L. et al. (eds.) Web Engineering: 7th International Conference, ICWE 2007 Como, Italy, July 16-20, 2007 Proceedings. pp. 457–472 Springer Berlin Heidelberg, Berlin, Heidelberg (2007) DOI:10.1007/978-3-540-73597-7_38.

28. Distante, D., Tilley, S.: Conceptual Modeling of Web Application Transactions: Towards a Revised and Extended Version of the UWA Transaction Design Model. In: 11th International Multimedia Modelling Conference. pp. 439–445 (2005) DOI:10.1109/MMMC.2005.28.

29. Driver, M. et al.: Rich Internet Applications are the next evolution of the Web. Gartner Research (2005).

30. Escalona, M.J., Aragón, G.: NDT. A model-driven approach for web requirements. IEEE Trans. Softw. Eng. 34, 3, 377–394 (2008) DOI:10.1109/TSE.2008.27.

31. Escalona, M.J., Koch, N.: Metamodeling the Requirements of Web Systems. Web Inf. Syst. Technol. - Lect. Notes Bus. Inf. Process. - Springer. 1, 267–280 (2007) DOI:10.1007/978-3-540-74063-6.

32. Escalona, M.J., Koch, N.: Requirements engineering for web applications: a comparative study. J. Web Eng. 2, 3, 193–212 (2003).

33. Finkelstein, A. et al.: Ubiquitous Web Application Development - A Framework for Understanding. In: Proc. of SCI2002. pp. 431–438 (2001).

34. Firmenich, D. et al.: CrowdMock: an approach for defining and evolving web augmentation requirements. Requir. Eng. 1–29 (2016) DOI:10.1007/s00766-016-0257-3.

35. Fraternali, P., Tisi, M.: Multi-level Tests for Model Driven Web Applications. In: Benatallah, B. et al. (eds.) Web Engineering: 10th International Conference, ICWE 2010, Vienna Austria, July 5-9, 2010. Proceedings. pp. 158–172 Springer Berlin Heidelberg, Berlin, Heidelberg (2010) DOI:10.1007/978-3-642-13911-6_11.

36. Gaedke, M., Gräf, G.: Development and Evolution of Web-Applications Using the WebComposition Process Model. In: Web Engineering. pp. 58–76 (2001) DOI:10.1007/3-540-45144-7.

37. Garzotto, F. et al.: HDM - A Model for the Design of Hypertext Applications. In: Hypertext ’91 Proceedings. pp. 313–328 (1991) DOI:10.1145/122974.123004.

38. Garzotto, F. et al.: HDM---a model-based approach to hypertext application design. ACM

Trans. Inf. Syst. 11, 1, 1–26 (1993) DOI:10.1145/151480.151483.

39. Garzotto, F. et al.: “modeling-by-patterns” of web applications. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 293–306 (1999) DOI:10.1007/3-540-48054-4_24.

40. Gómez, J., Cachero, C.: OO-H Method : Extending UML to Model Web Interfaces. Inf. Model. internet Appl. 144–173 (2003).

41. Guinard, D. et al.: A resource oriented architecture for the Web of Things. In: Internet of Things (IOT), 2010. pp. 1–8 (2010) DOI:10.1109/iot.2010.5678452.

42. Hennicker, R., Koch, N.: A UML-based methodology for hypermedia design. UML 2000 Unified Model. Lang. 410–424 (2000) DOI:10.1007/3-540-40011-7_30.

43. Hermida, J.M. et al.: Developing Semantic Rich Internet Applications Using a Model-Driven Approach. In: Web Information Systems Engineering - Wise 2010 Workshops. pp. 198–211 (2011) DOI:10.1007/978-3-642-24396-7_16.

44. Hutchinson, J. et al.: Empirical Assessment of MDE in Industry. In: Proceedings of the 33rd International Conference on Software Engineering. pp. 471–480 ACM, New York, NY, USA (2011) DOI:10.1145/1985793.1985858.

45. Isakowitz, T. et al.: RMM: a methodology for structured hypermedia design. Commun. ACM. 38, 8, 34–44 (1995) DOI:10.1145/208344.208346.

46. Jacobson, I., Bylund, S.: The road to the unified software development process. Cambridge University Press (2000).

47. Kienle, H.M., Distante, D.: Evolution of Web Systems. In: Mens, T. et al. (eds.) Evolving Software Systems. pp. 201–228 Springer Berlin Heidelberg, Berlin, Heidelberg (2014) DOI:10.1007/978-3-642-45398-4_7.

48. Knapp, A. et al.: Modeling Business Processes in Web Applications with ArgoUWE. In: 2004 - The Unified Modelling Language. pp. 69–83 (2004).

49. Koch, N.: Classification of model transformation techniques used in UML-based Web engineering. IET Softw. 1, 3, 98–111 (2007) DOI:10.1049/iet-sen:20060063.

50. Koch, N.: Software Engineering for Adaptive Hypermedia Systems: Reference Model, Modeling Techniques and Development Process. (2000).

51. Krasner, G.E. et al.: A description of the model-view-controller user interface paradigm in the smalltalk-80 system. J. object oriented Program. 1, 3, 26–49 (1988).

52. Larman, C., Basili, V.R.: Iterative and incremental developments. A brief history. Computer (Long. Beach. Calif). 36, 6, 47–56 (2003) DOI:10.1109/MC.2003.1204375.

53. Leff, A., Rayfield, J.T.: Web-application development using the model/view/controller design pattern. In: Enterprise Distributed Object Computing Conference, 2001. EDOC’01. Proceedings. Fifth IEEE International. pp. 118–127 (2001).

54. Limbourg, Q. et al.: USIXML: A Language Supporting Multi-path Development of User Interfaces. In: Ehci/Ds-Vis. pp. 200–220 (2005) DOI:10.1007/11431879_12.

55. Liu, C. et al.: Mashroom+: An Interactive Data Mashup Approach with Uncertainty Handling. J. Grid Comput. 12, 2, 221–244 (2014) DOI:10.1007/s10723-013-9280-5.

56. Lowe, D. et al.: Web extensions to UML: Using the MVC triad. In: International Conference on Conceptual Modeling. pp. 105–119 (2002).

57. Luna, E.R. et al.: Bridging Test and Model-Driven Approaches in Web Engineering. In: Web Engineering, 9th International Conference, ICWE 2009, San Sebastián, Spain, June 24-26, 2009, Proceedings. pp. 136–150 (2009) DOI:10.1007/978-3-642-02818-2_10.

58. Luna Robles, E. et al.: WebSpec: A visual language for specifying interaction and navigation requirements in web applications. Requir. Eng. 16, 297–321 (2011) DOI:10.1007/s00766-011-0124-1.

59. Martin, R.C.: Agile software development: principles, patterns, and practices. Prentice Hall PTR (2003).

60. Martinez, Y. et al.: Evaluating the Impact of a Model-Driven Web Engineering Approach on the Productivity and the Satisfaction of Software Development Teams. In: Web Engineering - 12th International Conference, {ICWE} 2012, Berlin, Germany, July 23-27, 2012. Proceedings. pp. 223–237 (2012) DOI:10.1007/978-3-642-31753-8_17.

61. Martínez, Y. et al.: Empirical study on the maintainability of Web applications: Model-driven Engineering vs Code-centric. Empir. Softw. Eng. 19, 6, 1887–1920 (2013) DOI:10.1007/s10664-013-9269-5.

62. Melia, S. et al.: MDA transformations applied to Web application development. In: Lecture Notes in Computer Science. pp. 465–471 (2005) DOI:10.1007/11531371_59.

63. Meliá, S., Gomez, J.: The WebSA Approach: Applying Model Driven Engineering to Web Applications. J. Web Eng. - dl.acm.org. 5, 2, 121–149 (2006).

64. Mohagheghi, P. et al.: An empirical study of the state of the practice and acceptance of model-driven engineering in four industrial cases. Empir. Softw. Eng. 18, 1, 89–116 (2013) DOI:10.1007/s10664-012-9196-x.

65. Moreno, N. et al.: An overview of model-driven web engineering and the MDA. In: Web Engineering: Modelling and Implementing Web Applications. pp. 353–382 Springer (2008).

66. Moreno, N., Vallecillo, A.: Towards Interoperable Web Engineering Methods. J. Am. Soc. Inf. Sci. Technol. 59, 7, 1073–1092 (2008) DOI:10.1002/asi.

67. Nanard, M. et al.: Pushing Reuse in Hypermedia Design : Golden Rules , Design Patterns and Constructive Templates. In: ACM Conference on Hypertext & Media. pp. 11–20 (1998) DOI:10.1145/276627.276629.

68. Navarrete, J.I.P. et al.: Introducing Usability in a Conceptual Modeling-Based Software Development Process. In: Atzeni, P. et al. (eds.) Conceptual Modeling - 31st International Conference ER 2012, Florence, Italy, October 15-18, 2012. Proceedings. pp. 525–530 Springer (2012) DOI:10.1007/978-3-642-34002-4_41.

69. Nebut, C. et al.: Automatic test generation: A use case driven approach. IEEE Trans. Softw. Eng. 32, 3, 140–155 (2006).

70. Nielsen, J.: Multimedia and Hypertext: The Internet and Beyond. AP Professional (1995).

71. Norrie, M.C. et al.: The Forgotten Many? A Survey of Modern Web Development Practices. In: 14th International Conference, ICWE 2014, Toulouse, France, July 1-4, 2014. Proceedings. pp. 290–307 Springer International Publishing (2014) DOI:10.1007/978-3-319-08245-5_17.

72. Norrie, M.C. et al.: X-Themes: supporting design-by-example. In: Web Engineering. pp. 480–489 Springer (2014).

73. Olsina, L. et al.: Web quality. In: Web Engineering. pp. 109–142 (2006) DOI:10.1007/3-540-28218-1_4.

74. Panach, J.I. et al.: In Search of Evidence for Model-Driven Development Claims: An Experiment on Quality, Effort, Productivity and Satisfaction. Inf. Softw. Technol. 62, 164–186 (2015) DOI:10.1016/j.infsof.2015.02.012.

75. Pastor, O. et al.: Conceptual modelling of web applications: The OOWS approach. In: Web Engineering. pp. 277–302 (2006) DOI:10.1007/3-540-28218-1_9.

76. Pastor, O. et al.: The OO-Method approach for information systems modeling: From object-oriented conceptual modeling to automated programming. Inf. Syst. 26, 7, 507–534 (2001) DOI:10.1016/S0306-4379(01)00035-7.

77. Paternò, F. et al.: ConcurTaskTrees: A diagrammatic notation for specifying task models. In: Human-Computer Interaction INTERACT'97. pp. 362–369 (1997).

78. Paternò, F., Santos, I.: Designing And Developing Multi-User, Multi-Device Web Interfaces. In: Computer-Aided Design Of User Interfaces V, Proceedings of the Sixth International Conference on Computer-Aided Design of User Interfaces, {CADUI} 2006 6-8 June 2006, Bucharest, Romania. pp. 111–122 (2006) DOI:10.1007/978-1-4020-5820-2_9.

79. Plessers, P. et al.: Accessibility: a Web engineering approach. World Wide Web. (2005).

80. Preciado, J.C. et al.: Designing rich internet applications combining UWE and RUX-method. In: Proceedings - 8th International Conference on Web Engineering, ICWE 2008. pp. 148–154 (2008) DOI:10.1109/ICWE.2008.26.

81. Prutsachainimmit, K. et al.: A Mashup Construction Approach for Cooperation of Mobile Devices. In: Current Trends in Web Engineering - ICWE 2012 International Workshops: MDWE, ComposableWeb, WeRE, QWE, and Doctoral Consortium, Berlin, Germany, July 23-27, 2012, Revised Selected Papers. pp. 97–108 Springer Berlin Heidelberg (2012) DOI:10.1007/978-3-642-35623-0_11.

82. Rivero, J.M. et al.: Mockup-Driven Development: Providing agile support for Model-Driven Web Engineering. Inf. Softw. Technol. 56, 6, 1–18 (2014) DOI:10.1016/j.infsof.2014.01.011.

83. Rossi, G. et al.: Designing personalized web applications. In: Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001. pp. 275–284 (2001) DOI:10.1145/371920.372069.

84. Rossi, G. et al.: Improving Web information systems with navigational patterns. Comput. Networks. 31, 11, 1667–1678 (1999) DOI:10.1016/S1389-1286(99)00015-8.

85. Rossi, G. et al.: Web Engineering: Modelling and Implementing Web Applications. Springer-Verlag London (2008) DOI:10.1007/978-1-84628-923-1.

86. Rubart, J.: Hypermedia design patterns. In: Proceedings of the Nineteenth ACM Conference on Hypertext and Hypermedia. pp. 243–244 (2008) DOI:10.1145/1379092.1379146.

87. Ruiz, M. et al.: Applying a web engineering method to design web services. In: International Conference on Service-Oriented Computing. pp. 576–581 (2005).

88. Rumbaugh, J. et al.: Object-oriented modeling and design. Prentice-hall Englewood Cliffs, NJ (1991).

89. Schauerhuber, A. et al.: Aspect-oriented modeling of ubiquitous Web applications: The aspectWebML approach. In: Proceedings of the International Symposium and Workshop on Engineering of Computer Based Systems. pp. 569–576 (2007) DOI:10.1109/ECBS.2007.20.

90. Schauerhuber, A. et al.: Bridging existing Web Modeling Languages to Model- Driven Engineering : A Metamodel for WebML. In: Conceptual Modeling and Code Generation for Rich Internet Applications (ICWE'06) Workshops. pp. 10–14 (2006) DOI:1-59593-435-9/06/07.

91. Schmid, H.A., Donnerhak, O.: OOHDMDA – An MDA Approach for OOHDM. In: Web Engineering. Springer Berlin Heidelberg. pp. 569–574 (2005).

92. Schmid, H.A., Rossi, G.: Modeling and designing processes in E-commerce applications. IEEE Internet Comput. 8, 1, 19–27 (2004) DOI:10.1109/MIC.2004.1260699.

93. Schwabe, D., Rossi, G.: An Object Oriented Approach to Web-Based Application Design. Theory Pract. Object Syst. 4, 207–225 (1998) DOI:10.1.1.29.57.

94. Schwabe, D., Rossi, G.: The object-oriented hypermedia design model. Commun. ACM. 38, 8, 45–46 (1995) DOI:10.1145/208344.208354.

95. Schwinger, W. et al.: A survey on web modeling approaches for ubiquitous web applications. Int. J. Web Inf. Syst. 4, 3, 234–305 (2008) DOI:10.1108/17440080810901089.

96. de Souza Bomfim, M.H., Schwabe, D.: Design and Implementation of Linked Data Applications Using SHDM and Synth. In: Proc. of the 11th International Conference on Web Engineering. pp. 121–136 (2011) DOI:10.1007/978-3-642-22233-7_9.

97. Troyer, O. De et al.: Wsdm : Web Semantics Design Method. In: Web Engineering: Modeling and Implementing Web Applications. pp. 303–351 (2008) DOI:10.1007/978-1-84628-923-1_11.

98. Troyer, O. De et al.: Wsdm : Web Semantics Design Method. In: Web Engineering: Modeling and Implementing Web Applications. pp. 303–351 (2008) DOI:10.1007/978-1-84628-923-1_11.

99. Urbieta, M. et al.: Designing the Interface of Rich Internet Applications. In: 2007 Latin American Web Conference (LA-WEB 2007). pp. 144–153 IEEE (2007) DOI:10.1109/LA-Web.2007.14.

100. Urbieta, M. et al.: Modeling, Deploying, and Controlling Volatile Functionalities in Web Applications. Int. J. Softw. Eng. Knowl. Eng. 22, 129–155 (2012)

DOI:10.1142/S0218194012500064.

101. Valderas, P. et al.: A transformational approach to produce web application prototypes from a web requirements model. Int. J. Web Eng. Technol. 3, 1, 4–42 (2007) DOI:10.1504/IJWET.2007.011526.

102. Valderas, P. et al.: Using Task Descriptions for the Specification of Web Application Requirements. In: Anais do WER05 - Workshop em Engenharia de Requisitos, Porto, Portugal, Junho 13-14, 2005. pp. 257–268 (2005).

103. Valderas, P., Pelechano, V.: A Survey of Requirements Specification in Model-Driven Development of Web Applications. ACM Trans. Web. 5, 2, 1–51 (2011) DOI:10.1145/1961659.1961664.

104. Vallecillo, A. et al.: MDWEnet: A practical approach to achieving interoperability of model-driven web engineering methods. In: CEUR Workshop Proceedings. (2007).

105. Vdovjak, R. et al.: Engineering semantic web information systems in hera. J. Web Eng. 0, 0, 1–24 (2003).

106. Vilain, P. et al.: A Diagrammatic Tool for Representing User Interaction in UML. In: Evans, A. et al. (eds.) International Conference on the Unified Modeling Language. pp. 133–147 Springer (2000) DOI:10.1007/3-540-40011-7_10.

107. Wakil, K., Jawawi, D.N.A.: Model Driven Web Engineering: A Systematic Mapping Study. e-Informatica. 9, 1, 87–122 (2015).

108. Whittle, J. et al.: Industrial adoption of model-driven engineering: Are the tools really the problem? In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 1–17 (2013) DOI:10.1007/978-3-642-41533-3_1.

109. Whittle, J. et al.: The State of Practice in Model-Driven Engineering. IEEE Softw. 31, 3, 79–85 (2014) DOI:10.1109/MS.2013.65.

110. Yoo, J., Bieber, M.: Finding Linking Opportunities Through Relationship-based Analysis. In: Proceedings of the Eleventh ACM on Hypertext and Hypermedia. pp. 181–190 ACM, New York, NY, USA (2000) DOI:10.1145/336296.336359.

111. 7th Model-Driven Web Engineering Workshop, http://mdwe2011.pst.ifi.lmu.de/.

112. 8th International Workshop on Web-Oriented Software Technologies (IWWOST), https://www.irit.fr/recherches/ICS/events/conferences/iwwost/iwwost2009/.

113. ATL Transformation Language, http://www.eclipse.org/atl/.

114. BPMN Specification - Business Process Model and Notation, http://www.bpmn.org/.

115. Conference on Advanced Information Systems Engineering (CAiSE), http://dblp.uni-trier.de/db/conf/caise/index.html.

116. Drive digital transformation & innovation with Mendix aPaaS, https://www.mendix.com/.

117. IBM - Rational Unified Modeling Language - Products - UML, https://www-01.ibm.com/software/rational/uml/products/.

118. ICWE Conference Series, http://www.iswe-ev.de/conferences/icwe/.

119. Integranova, http://www.integranova.com/.

120. Interaction Flow Modeling Language, http://www.omg.org/spec/IFML/.

121. International Conference on Software Engineering (ICSE), http://www.icse-conferences.org/.

122. International World Wide Web Conference (WWW), http://www.iw3c2.org/.

123. MDA Specifications, http://www.omg.org/mda/specs.htm.

124. Model To Text (M2T), https://eclipse.org/modeling/m2t/?project=xpand.

125. MOFM2T, http://www.omg.org/spec/MOFM2T/.

126. MPS overview, https://www.jetbrains.com/mps/.

127. NDT Suite, http://iwt2.org/actividad-grupo/investigacion/resultados/ndt/ndt-suite/.

128. Object Management Group, http://www.omg.org.

129. OMG's MetaObject Facility (MOF), http://www.omg.org/mof/.

130. OutSystems Platform, https://www.outsystems.com/.

131. Query/View/Transformation (QVT), http://www.omg.org/spec/QVT/.

132. Seventh International World-Wide Web Conference, http://www7.wwwconference.org/.

133. The Unified Modeling Language (UML) - Conferences and Workshops, http://dblp.uni-trier.de/db/conf/uml/index.html.

134. Unified Modeling Language, http://www.omg.org/spec/UML/.

135. UWE - MagicUWE, http://uwe.pst.ifi.lmu.de/toolMagicUWE.html.

136. UWE Metamodel and Profile, http://uwe.pst.ifi.lmu.de/publicationsMetamodelAndProfile.html.

137. Visual Paradigm, https://www.visual-paradigm.com/.

138. WebRatio Platform, http://www.webratio.com/site/content/en/web-application-development.

139. Workshop on Model-driven Web Engineering (MDWE 2005), http://www.lcc.uma.es/~av/mdwe2005/.