Challenges for the Adoption of Model-Driven Web Engineering Approaches in Industry

Esteban Robles Luna[1], F. J. Domínguez-Mayo[3], José Matías Rivero[1, 2], J. A. García-García[3], J. M. Sánchez-Begines[3], M. J. Escalona[3] and G. Rossi[1, 2]
[1]*LIFIA, Facultad de Informática, UNLP, La Plata, Argentina*
[2]*Conicet, Argentina*
[3] *Web Engineering and Early Testing Group, University of Seville, Seville, Spain*

Keywords: Model-Driven Web Engineering, Human-centred Interfaces, MDE, Industry.

Abstract: Model-driven web engineering approaches have become an attractive research and technology solution for Web application development. However, after 20 years of development, they have attracted little attention from the Industry due to the mismatch between technical versus research requirements. In this joint work between academia and industry, the authors present the current problems of using these approaches in scale and provide guidelines to convert them into viable industry solutions.

1 INTRODUCTION

Model Driven Web engineering (MDWE) approaches appeared 20 years ago to fulfill a missing area of Model Driven Development: Web application development (Selic, 2003). From that moment, about 7 to 8 of MDWEs (Rossi et Al, 2007) were created but only a few ended up providing tool support and one became the mayor player with small company and support from OMG to convert its language to a standard.

Though MDWEs have claimed to improve multiple aspects of Web application development such as: code quality, development speed and level of abstraction , Why has the industry given little attention to it? A recent study (Hull, 2013) has presented the 20 obstacles that hinder Web application scalability and though they are not specifically targeted to the applications derived from MDWEs, they are still affected by them. Another study related to Web Engineering in the Cloud clearly explains some of the problems of moving the current tools to support this kind of applications. Though the study is not focus on why MDWEs are not a viable solution in the Industry, some of the problems presented clearly show related technical limitations of current MDWEs.

In this study, we present a list of issues that hinders the usage of MDWE in medium to big size companies and as a consequence shows clear practical problems that need to be resolved to support the claims that the MDWE community have done for years. To provide a context for the issues we have found, Figure 1 presents an agile development life-cycle process of a Web application. In this figure, the main phases appear in bold type format and the issues that hinders the usage of MDWE are represented by posits that are sticked to the picture in the affected phase in which the issue is localized.

The study uses small experiments to apply MDWE in a company in addition to exhaustive literature review to give support to its claims. As many different aspects are considered (not only technical) when dealing with limitations of MDWE approaches, we have created a categorization to stress the area where the issue was found:

- Social [S]: A social issue is related with problems between the people involved in the project or between the people and the software artifact.
- Technical [T]: A technical issue is related with the software elements that constitute the Web application.
- Economical [E]: An economical issue is related with the project's budget or money involved in the development of the application.

Luna, E., Domínguez-Mayo, F., Rivero, J., García-García, J., Sánchez-Begines, J., Escalona, M. and Rossi, G.
Challenges for the Adoption of Model-Driven Web Engineering Approaches in Industry.
DOI: 10.5220/0006382704150421
In *Proceedings of the 13th International Conference on Web Information Systems and Technologies (WEBIST 2017)*, pages 415-421
ISBN: 978-989-758-246-2

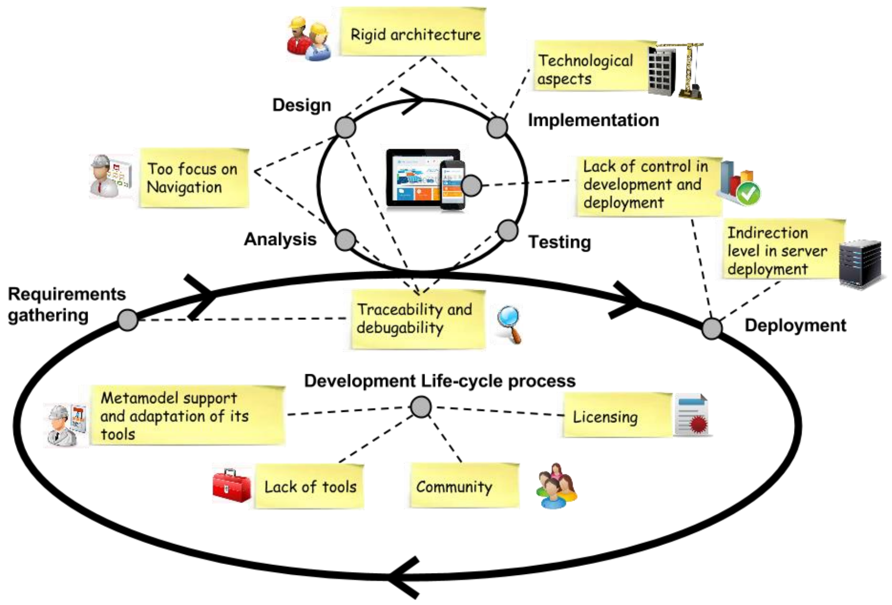
Rigid architecture
Technological aspects
Design
Implementation
Too focus on Navigation
Lack of control in development and deployment
Analysis
Testing
Indirection level in server deployment
Requirements gathering
Traceability and debugability
Deployment
Development Life-cycle process
Metamodel support and adaptation of its tools
Licensing
Lack of tools
Community

Figure 1: Issues that hinders the usage of MDWE in medium to big size companies.

In Table 1, we sum up the categorization of the issues that we will next present in the following subsections. Each subsection explains the issues and offers a final guideline to solve it.

2 CHALLENGES

2.1 Lack of Control in Development and Deployment [S, T, E]

Creating a Web application is a complex process that involves not only coding/modeling but also having meetings with stakeholders and debugging the application to fix production problems. In particular, when the application is deployed, aspects such as monitoring, logging and profiling become more important (Hull, 2013). Therefore, the engineering teams that develop the web application have the ownership and responsibility for the deliverables and as a consequence they want to have to control the complete process (from development to deployment).

Though model driven technologies provide many benefits, they also add an extra level of complexity as the derivation process feels like a "magic wand" that obtains an application from a set of models. Most MDWE tools hide and make this process "close source" creating a dependency between the MDWE tool and the development teams. This dependency is not good with time constraints and not having the source code available to everyone makes things worst. Additionally, it is highly probable that the commitment from the MDWE tool team requires paying fees in the form of licenses that adds another economic cost to the development.

2.1.1 Guideline

To give more control of the actual development, MDWE tools need to provide:

- Ways to hook in the modeling and transformation phases so that developers can add and improve the development process based on the application they are building. This could be done if the transformation algorithm works using the template method design pattern.
- Make the code open source so that the tool is easy to debug by the development team in case problems arise during development or maintenance.

2.2 Too Focused in Navigation [T]

Original hypermedia based Web applications that were developed 10 years ago are rather different from the current integration Web paradigm were navigation is one tiny concern. Aspects such as Rich Internet applications, integration with different systems, search capabilities, personalization and recommendation have become aspects that are more complex and more important than navigation.

In the early stages, MDWEs were created to adapt the navigation paradigm from hypermedia to

Table 1: List of issues by category.

	Social	Technical	Economical
Lack of control in development and deployment	X	X	X
Too focused in navigation		X	
Metamodel support and adaptation of its tools	X	X	
Traceability and debugability		X	
Lack of tools		X	
Rigid architecture		X	
Technological aspects		X	
Community	X		X
Just in time development		X	
Licensing			X

the Web. Many new concepts were introduced and although most of these enhancements have been reported in the literature (Robles et Al, 2011) only a few of them has been actually implemented. As a consequence, still today, the main model of industry leading WebRatio is the navigational model that describes the navigational paths that a user can follow. On the contrary to what many researches have shown about the importance of the UI aspects, integration (among others), navigation is still the most important aspect for MDWE approaches.

2.2.1 Guideline

MDWEs need to detach from being focus in navigation to support navigation as one concern of the Web application lifecycle. Additionally, there is a special need to provide a real tool support for the features that MDWEs claim to have and in the case that they are not supported, provide hooks to perform manual coding of these features.

2.3 Metamodel Support and Adaptation of Its Tools [T, S]

The increasing number of technologies that are being developed every day in addition to customer's time constraints poses multiple challenges to Web application development. Although these tasks can be performed manually (in code based environments) and may be time consuming, they can be easily done by extending and hooking into the hot spots that code base frameworks provide.

In MDWEs, this kind of extensions may be either done by instantiating some preexisting metamodel classes or, if the functionality is not supported, by metamodel extensions. Extending a metamodel does not only require to add the new classes and transformations but also to adapt the tools in a timely manner. Some approaches such as NDT[1] or UWE[2] are extensions of the UML profile, so, these approaches can be adapted easily than others such as IFML[3].

2.3.1 Guideline

Due to the existing technology and economic time constraints, metamodels and tools need to be adapted within days to cope with customer's demands.

2.4 Traceability and Debugability [T]

One of the most important aspects of software development is the ability to introspect, change and monitor the "live" application under development to quickly fix the problems; these actions are considered as "debugging" an application. Code based development in high order languages such as Smalltalk, Java and .NET have these features from the beginning and they make easy to iterate in the development process.
In MDWE tools where code is generated from models, the ability to debug is related to derive traceability links between the models and the generated code. Nowadays, only WebRatio partially supports this schema (Fraternali and Tissi, 2011) by allowing debugging the application under development. Still, WebRatio needs further work to help tracing back the problems while the application is running in production and exceptions occur as non-support is provided in this case. All these aspects make core engineers to avoid adopting MDWE tools as they lack control over the system under development.

[1] NDT and NDT-Suite, Retrieved April 2014 from http://www.iwt2.org
[2] UWE and MagicUWE, Retrieved April 2014 from http://uwe.pst.ifi.lmu.de
[3] IFML: The Interaction Flow Modeling Language, Retrieved April 2014 from http://www.ifml.org

2.4.1 Guideline

As high-level languages (e.g. Java) provide ways to trace back problems to concepts of the language (e.g. classes and line numbers in exception stack traces), MDWE should provide those features in order to detect the root causes of the problems. Additionally, further debug support must be provided to debug, evaluate and alter the application while it is running in a development environment.

2.5 Lack of Tools [T]

Building a Web application requires a set of tools that eases the process of development, deployment and monitoring. For example, a typical JEE Web application can be developed using Maven[4] as a build system, Jenkins[5] for continuous integration and a variety of frameworks to actually build the application (Spring, Hibernate and JQuery to name a few). To perform the actual deployment, a set of tools that automate and control the process from the moment the application is built to the instantiation of the servers and application deployment and its initial monitoring is provided. All these tools though from a lower level of abstraction (if compare with MDWE) clearly help to build and deploy an application.

However, in the MDWE area to much focus has been put in the actual development of the application from high-level models. Though that is correct from the MDWE philosophy, it increases the effort put in development and monitoring for many other topics discuss in this work (Technological features and Traceability and debugability). To provide concrete examples to the reader:

- There are no tools to trace back stacktrace exceptions back to the model elements. This aspect makes super hard to correct errors of the application deployed.
- There are no tools to support the monitoring of the model elements and as a consequence detecting performance issues is hard. By using New Relic[6], we can detect some of these aspects but those would be classes derived from the MDWE tool that may correspond to multiple model elements.

[4] Maven, Retrieved April 2014 from http://maven.apache.org
[5] Jenkins, Retrieved April 2014 from http://jenkins-ci.org
[6] New Relic, Retrieved April 2014 from http://newrelic.com

2.5.1 Guideline

To provide an industry valid approach, MDWEs need to support a handful set of tools that help with the complete application lifecycle. Some of tools mentioned in this section (Maven, Jenkins, New Relic) are fully extensible so building some of these tools can be fairly simple by having good traceability links and extending them with the right information. Tools around the Web applications are one of the most critical aspects to keep the application running 24 by 7.

2.6 Rigid Architecture [T]

Creating a Web application of any size may require small to big changes in a standard 3-tier Web application. Some aspects that need to be considered may involve integration with external services, asynchronously processing of queued information, exposed of REST services for external users or internal mobile applications, etc. As a consequence been able to adapt the architecture to support any of these types of requirements is extremely important.

In current MDWE tools, the architecture is not modeled at all and as a consequence they derive a simplistic 1 tier Web application[7]. A 1-tier architecture can only handle a few sets of uses cases and doesn't allow the development team to be able to adapt to future needs. A recent paper (Toffetti, 2012) showed the need to model theses aspects in some way so that they can be considered through the derivation process. In its current state, MDWE tools can derive simple applications that may not scale well, thus making harder to gain adoption in medium to big size companies.

2.6.1 Guideline

To be able to adapt to more complex requirements that may involve functional (e.g. processing offline data) or non functional (e.g. performance and scalability issues) requirements, MDWE need to model the architecture in such a way that development teams can decide which approach to use. We must stress that though some architectures (e.g. 3-tier Web app) can be pre-configured, it is important to model the architecture primitives and let development teams abstract higher-level concepts from them such as the 3-tier Web app instead of hardcoding it.

[7] A big limitation of this approach is that they are not able to handle more the a few hundred of users as they have a tight dependency with the database and database generally handle less than 300 concurrent connections.

2.7 Technological Aspects [T]

Logging, caching, load balancing and profiling (to name a few) are some of the aspects that engineers need to build high scalable Web applications (Hull, 2013). The lack of any of these poses some limitations on the type of application that you can build. For instance, lacking a caching strategy forces the application to compute or fetch information for every request limiting the application growth. Additionally it may add the following problems:

- Run out of DB connections: Being not able to cache information stored in a DB requires the usage of a DB connection for every request. Thus, because of a DB limitation issue, the maximum number of users able to access the Web application is equal to the number of DB connections and as a consequence, new users won’t be able to access the DB.
- Increasing response time: If we can’t cache external service calls, those calls need to happen every time thus increasing the overall response time of the requested Web page.
- Increase in hardware needed: If no cache is provided, we may need to use more hardware to recompute values that were computed before.

Though caching is fairly simple aspect that is intrinsic to application development, MDWE consider it, and the aforementioned aspects, as “technological”. Being part of this category means that little importance has being paid in the models and as a consequence engineers will have to tweak them in the generated code. As none of the MDWE tools provide a roundtrip between the generated code and the models, these “technological” tweaks have to be adjusted every time the application is derived.

2.7.1 Guideline

The “technological” aspects need to be considered in some way inside the model driven development. If MDWE move to consider and model them, it would provide a great benefit for the size and quality of the application that can be built with MDWE tools. At the same time, the response time can be tweak to decrease while a bigger amount of work is handled by the same amount of hardware; and that will clearly show the benefits of using a model base solution.

2.8 Community [S, E]

In the MDWE research area, there is a good initiative like the MDWEnet[8] which the main research focus is on meta-modelling and on model transformations, and it was created with the aims of improving the interoperability between existing MDWE approaches and their tools and to provide better methods and solutions to the industry. Today, the fact is that most approaches have a great lot of not agreed aspects; i.e. (Dominguez-Mayo et Al, 2012): Meta-models and models are different, different way to implement transformations or different tools and used technology among other things. Then, there is a lack of consensus and documentation between approach designers and this entire context is causing different situations:

- On one hand, organizations do not know how they can take advantage of these approaches and how they can be helped in their particular context due to the diversity set of characteristics offered by these approaches and the global heterogeneity associated with specific aspects or ideas processed by each approach.
- On the other hand, under this situation is very complicated for designers of approaches to identify the real organization’s needs and demands in order to improve their approaches or design new ones.

However, within the context of MDWE, there is an exception with WebRatio. This tool support provides a big community with a big variety of tutorials, webinars, user guides, support, forums, and different types of tool certifications for users. There is no doubt that WebRatio is the current leader tool in the market within the context of MDWE approaches. However, neither this tool nor this community can be compared to other communities in the world related to Web development.

The other side of the coin of MDWE are existing and very extended Web development frameworks in the world like Ruby on Rails[9], Django[10], Grails[11] or Codeigniter[12] among others. All of them count with big communities of developers that provide lot of documentation, tutorials, user guides, forums and support among other things. In addition, these

[8] MDWEnet, Retrieved April 2014 from http://www.iswe-ev.de/activities/2007/mdwe/
[9] Ruby on Rails, Retrieved April 2014 from http://rubyonrails.org
[10] Django, Retrieved April 2014 from https://www.djangoproject.com
[11] Grails, Retrieved April 2014 from http://grails.org
[12] Codeigniter, Retrieved April 2014 from http://ellislab.com/codeigniter

frameworks have in common lot of features and common components that let developers compare these frameworks between them.

2.8.1 Guideline

So, these limitations and problems of description on MDWE not only entail understanding these issues, but also require unifying criteria and define common strategies in a shared quality model (Dominguez-Mayo et Al, 2012). In addition, this common model could help to approach designers when improving or designing new approaches in future. Besides, a common model can help to developers to compare all these MDWE approaches between them.

2.9 Licensing [E]

Today, two main business models to exploit these approaches in industry have been found on MDWE.

The first business model consists in implementing a specific tool support for the approach from free source environments. This specific tool can be offered to organizations by a license fees or freely. As regards license fees, there are different types of fees by each tool depend on the case of the organization. According to existing license fees on MDWE, we have classified their costs in "High" (more than 5.000 $ by activated seat), "Medium" (between 5.000 $ and 1000 $ by activated seat) and "Low" (less than 1000 $ by activated seat) costs. For instance, WebRatio is developed under a free source environment like eclipse but with extensions that transform the eclipse environment in a practical and valuable product like WebRatio. In this case, WebRatio is a powerful tool to support the IFML visual modeling standard. In this case, WebRatio offers an enterprise edition license in which organizations must pay a fee for each activated seat. This price can be classified by "High" costs.

The second business model consists in implementing the approach under the environment of a powerful but payment CASE tool support. So, the use of the approach is free but organizations must pay for the license of this CASE tool support in which the approach is supported. For example, the NDT-Suite is a tool to support that is developed under the Enterprise Architect [13] environment. This CASE tool allows organizations to have and work with all elements of the NDT approach and, under the environment of Enterprise Architect enables organizations to work with the concepts of the approach and lot of other visual modeling diagrams and characteristics that EA additionally offers to organizations. In this case, one activated seat of EA licenses can be classified by "Low" costs. Other example of this business model is MagicUWE, which is a tool that has been developed for the computer-aided design of Web applications using the UWE (UML-based Web Engineering) approach. MagicUWE has been built as a plugin of MagicDraw[14], which is a tool support that can be classified by "Medium" costs.

We must consider that organizations just going to pay a fee for these licenses if it is sure that they are going to receive the value they need but with minimal costs, risks and incertitude. As regards value of tool support, it is not possible to know what the most valuable approach is because it depends on the context. So, each tool support has its strength and weakness points. So, just with the context we could say which approach is the most suitable one for it. The adoption of these tools in industry could be achieved maximizing their competitiveness: Competitiveness = Value / (Cost + Risk + Incertitude).

In fact, in the market, there are other very known and used frameworks for the development of Web applications that they are not so abstract approaches but effective solutions for developers. Frameworks like Ruby on Rails or Django that encourages rapid development and clean, pragmatic designs for developers and they are completely free of costs. And one of the most important things is that they are currently very extended solutions by developers. So, this increases the trust in organizations to consider their use.

2.9.1 Guideline

Key questions are not only the costs of a license but also the value (although it depends on the context), risks and incertitude that organizations assume with the implementation of these approaches and tools. So, it is important to offer a high value solution, with no costs and minimal risks and incertitude for developers.

ACKNOWLEDGEMENTS

This research has been partially supported by the POLOLAS project (code TIN2016-76956-C3-2-R) of the Spanish Ministry of Science and Innovation,

[13] Enterprise Architect, Retrieved April 2014 from http://www.sparxsystems.com

[14] MagicDraw, Retrieved April 2014 from http://www.nomagic.com/products/magicdraw.html

by the SoftPLM Network (TIN2015-71938-REDT) of the Spanish the Ministry of Economy and Competitiveness.

REFERENCES

Selic, B., "The pragmatics of model-driven development" IEEE Software, vol.20, no.5, pp.19,25, 2003, doi: 10.1109/MS.2003.1231146

Rossi, G., Pastor, O., Schwabe, D., Olsina, L.: Web Engineering: Modelling and Implementing Web Applications. Springer. (2007)

Hull, S. (2013), "20 Obstacles to Scalability". Communications of the ACM, Vol. 56 No. 9, Pages 54-59, DOI: 10.1145/2500468.2500475.

Esteban Robles Luna, Gustavo Rossi, Irene Garrigós: WebSpec: a visual language for specifying interaction and navigation requirements in web applications. *Requir. Eng. 16*(4): 297-321 (2011)

Rivero, J. M., Grigera, J., Rossi, G., Robles Luna, E., Montero, F., & Gaedke, M. (2014). Mockup-Driven Development: Providing agile support for Model-Driven Web Engineering. Information and Software Technology, 56(6), 670–687. doi:10.1016/ j.infsof.2014.01.011

Nathalie Moreno, Antonio Vallecillo: Towards interoperable Web engineering methods. *JASIST 59*(7): 1073-1092 (2008)

Piero Fraternali, Massimo Tisi: Using Traceability Links and Higher Order Transformations for Easing Regression Testing of Web Applications. *J. Web Eng. 10*(1): 1-20 (2011)

Francisco José Domínguez Mayo, M. José Escalona, Manuel Mejías, M. Ross, G. Staples: Quality evaluation for Model-Driven Web Engineering methodologies. *Information & Software Technology 54*(11): 1265-1282 (2012).

Toffetti G.: Web engineering for cloud computing (web engineering forecast: cloudy with a chance of opportunities). In *Proceedings of the 12th international conference on Current Trends in Web Engineering (ICWE'12)*. Springer-Verlag, Berlin, Heidelberg, 5-19 (2012)

Schön, E. M., Escalona, M. J., & Thomaschewski, J. Agile values and their implementation in practice. *IJIMAI*, 3(5), 61-66.(2015)