# A Practical Environment to Apply Model-Driven Web Engineering

**Maria Jose Escalona, J.J. Gutiérrez, F. Morero, C.L. Parra, J. Nieto, F. Pérez, F. Martín and A. Llergo**

## Abstract

The application of a model-driven paradigm in the development of Web Systems has yielded very good research results. Several research groups are defining metamodels, transformations, and tools which offer a suitable environment, known as model-driven Web engineering (MDWE). However, there are very few practical experiences in real Web system developments using real development teams. This chapter presents a practical environment of MDWE based on the use of NDT (navigational development techniques) and Java Web systems, and it provides a practical evaluation of its application within a real project: specialized Diraya.

**Keywords**:

Model-driven engineering · Web engineering

## 1. Introduction

Web engineering [5] and the new paradigm of model-driven engineering [16] have been defined as suitable solutions for companies and Web development in the research environment [17]. New techniques for Web system developments introduced by Web engineering and the application of model-driven engineering in this area offer very interesting solutions of high quality and reduced cost.

However, although these ideas have been widely accepted by the research community, few practical applications can be obtained in the literature. In fact, during 2005 and 2006 we interviewed a group of 30 software companies in Andalusia, Spain. More than 50 project managers and 70 analysts were interviewed. These companies represent local, national, and international companies and thereby offer a very representative sample. One of our questions was whether the company knew anything about Web engineering. In big and medium-sized companies (more than 50 employees), 25% knew something about Web engineering, while in small companies, only 10% had heard about it. Overall, only 1% knew about Web engineering and applied it in projects. These results are very representative since, although the interviews indicate that Web engineering could be very useful, it is not currently in use.

One of the most important aspects relevant for the practical application of software engineering in general, and in Web engineering in particular, is the use of suitable tools that guarantee that the application of these techniques is profitable [6]. In comparative studies of Web engineering, one of the most important gaps detected is the lack of tools to support the application of Web approaches [4, 11]. Only some methodologies, such as UWE (UML Web engineering) [11] that offers ArgoUWE[1], or WebML [3] that offers WebRatio[2], introduce suitable solution tools.

This chapter presents a practical solution in the application of model-driven Web engineering. In the chapter, the Web methodology NDT (navigational development techniques) [9] is presented as a practical solution for Web developments. NDT is an approach to define and analyze Web systems and capture their requirements. The practical version of this approach is oriented to offer a suitable methodological environment for Web development. The chapter is structured as follows. In Section 2, the methodological solution is presented. This solution is oriented in the area of public administration in Andalusia in the south of Spain. In this environment, a methodology, named Métrica v3,[3] is used to develop software systems in public administration. Métrica v3 is a widely used methodology but is too ambiguous in some cases and offers no special treatment of Web characteristics. The methodological solution is composed of a fusion between Métrica and NDT. In Section 3, the set of tools to support this environment is presented: tools to support the methodology and tools to guarantee the results. In Section 4, a practical example of the application of this solution is given. In the Diraya project [8], this methodological environment is currently being applied with very good results. And, finally, in Section 5, conclusions are drawn and future work is proposed.

## 2. The Methodology

The methodological environment proposed in this chapter is based on the model-driven paradigm. This environment is a fusion between Métrica v3 and NDT, a model-driven Web methodology.

In this section, a comprehensive vision of this environment is shown and a presentation of the overall approach is put forward.

### 2.1. Métrica v3

Métrica is a methodological environment developed by the Spanish Ministry of Public Administration. In the latest version of Métrica v3, the object-oriented paradigm is included as a development option, and Métrica proposes the use of UML [14] to model different aspects in the life cycle.

Métrica v3 is the reference frame for the development of software in public organisms in Spain. Every software project developed for any government in Spain has to follow Métrica rules.

The fact that Métrica is obligatory for public organisms has provoked most software providers to use this methodology in their software projects. It has therefore become a widely used methodology in Spain. The life cycle of Métrica v3 is presented in Fig. 26.1.

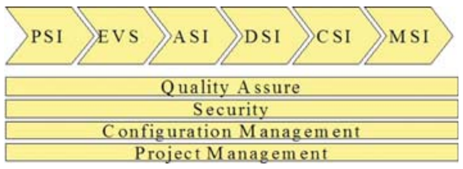

**Figure 26.1.** Life cycle of Métrica v3.

[1] http://www.pst.informatik.uni-muenchen.de /projekte/argouwe

[2] http://www.webratio.com

[3] http://www.map.es

This life cycle starts with the information system planning (PSI) where the organization is studied and the development environment for new systems in the organization is defined. PSI must be applied every 4 years in order to define the reference environment for the organization.

When the PSI defines the necessity of a new system, a viability study must be developed (EVS). EVS is normally an optional phase although it is mandatory in large and complex developments.

The next phases are obligatory for each system. The first phase, the analysis phase (ASI), must detect system requirements and then analyze them in order to define the scope of the system. After the analysis, the design phase must be tackled (DSI). The next phase is the construction of the system (CSI), where the system is translated into the selected programming language. And, finally, the maintenance phase must be applied (MSI).

In parallel to this life cycle, Métrica proposes four interfaces in order to control development.

- The quality assurance interface that applies quality techniques to control quality of the results.
- The security interface to control the security aspects of the systems.
- The configuration management interface to manage the structure and the organization's construction rules.
- The project management interface to control the management of the project throughout the life cycle.

For each phase, Métrica defines tasks and objectives that must be covered. Furthermore, Métrica offers a technique guide that can be applied in each task.

Métrica is a very complex and extensive approach. It offers a wide life cycle with a great number of techniques and tasks. However, for companies, it is sometimes very complex to identify which part of Métrica must be used or what products must be generated. This problem is very relevant in the treatment of requirements since, in Métrica, only use case diagram technique [14] is proposed for this phase. Use cases, mainly in complex environments, are very ambiguous and must be complemented with some description in order to solve this ambiguity [10, 18]. Moreover, use cases are insufficient for the extraction of the necessary information to attain analysis models. Furthermore, in the Web environment, Métrica does not offer special techniques or models to deal with the most critical characteristics of the Web: navigation, critical interface, multiple unknown final users, etc.

## 2.2. NDT – Navigational Development Techniques

NDT is a Web methodological process focused on both the requirement and the analysis phases. NDT offers a systematic way to deal with the special characteristics of the Web environment. NDT is based on the definition of formal metamodels [9] that allow derivation relations to be created between models. NDT takes this theoretic base and enriches it with the elements necessary for the definition of a methodology: techniques, models, methodological process, etc., in order to offer a suitable context for application in real projects.

In Fig. 26.2, the life cycle of NDT is presented. NDT only covers the requirement and the analysis phases. In the requirement phase, it involves the capture, definition, and validation of requirements. To this end, NDT proposes the division of requirements into different groups depending on their nature: storage information requirements, functional requirements, actors' requirements, interaction requirements, and non-functional requirements. In order to deal with each kind of requirement, NDT proposes the use of special patterns and UML techniques, such as the use case techniques. Requirements in NDT are formally presented in a requirement metamodel, where some constraints and relations are defined.

The life cycle then passes to the analysis phase. NDT proposes three models in this phase: the conceptual model, the navigational model, and the abstract interface model. The conceptual model of NDT is represented in the methodology using the class diagram of UML, and the other two models are represented using UWE notation [12].

The class diagram of UML and the navigational and the abstract interface of UWE have their own metamodels. From among the requirement metamodels and analysis metamodels, NDT defines a set of QVT transformations that are represented in the figure with the *QVTTransformation* stereotype. Thus, the

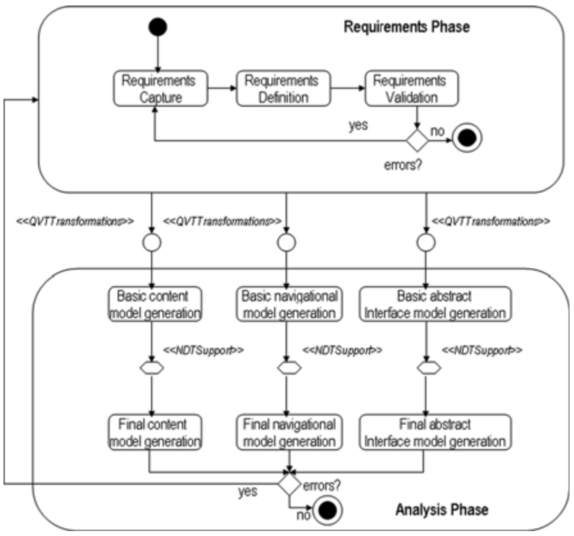

**Figure 26.2.** Life cycle of NDT.

shift from requirements to analysis in NDT is a systematic method based on these formal transformations. The direct application of these transformations generates a set of analysis models known in NDT as the basic analysis models. After the systematic generation, the analyst group can change these basic models by adding new relations, attributes, etc., that improve the models. This step depends on the analyst knowledge and is presented in the figure with the stereotype *NDTSupport*. This improvement generates the final analysis models. This second step is not systematic. However, NDT has to ensure that agreement between requirement and analysis models is maintained. Hence, this step is controlled by a set of rules and heuristics defined in NDT.

After the analysis model has been created, the development process can continue with another methodology, such as UWE or OOHDM [15], in order to obtain the code.

NDT offers a suitable environment for the development of Web systems. It offers specific techniques to deal with critical aspects in the Web environment. If a correlation with MDA (model-driven architecture) [13] is made, NDT presents a CIM (computational-independent model) in the requirements phase; a set of PIMs (platform-independent models), in the analysis phase; and a set of formal transformations between them.

NDT has been widely applied in practical environment and has achieved very good results, since it reduces the development time with the application of transformations and ensures agreement between requirements and analysis. In [7] the practical evolution of NDT is presented together with some of the most important practical applications.

## 2.3. A Practical Combination Between Métrica v3 and NDT

As stated in [7], Métrica and NDT can be easily merged to offer a suitable environment for Web development.

Métrica, in its object-oriented version, is based on UML models which are formally defined as extensions of NDT. Thus, the incorporation of NDT ideas in Métrica is, basically, the incorporation of its UML extensions.

As a practical solution, the research group of NDT has developed an approach which merges these two methodologies. This fusion is put forward in [7] and it follows the life cycle of Métrica presented in Fig. 26.1 albeit with some modifications. The ASI phase is divided into two parts: the requirement phase and the analysis phase. Both are developed using the life cycle of NDT. Thus, a navigational model and an abstract interface model are developed. Furthermore, metamodels, techniques, and transformations of NDT are applied. The DSI phase is also enriched with some specific concepts of Web engineering. Therefore, a design navigational model has to be defined.

This practical solution has been widely accepted in Andalusia. In this area, the software development in public administration is big business for consultants and software companies.

However, the application of this approach without a set of suitable tools to support the development is impossible. For this reason, a set of tools for supporting this fusion between Métrica and NDT is presented in the next section.

## 3. Tool Support

In order to support this practical approach, a set of tools were defined to help development teams apply this approach. The set of tools is composed of two main tools that support the development process. The first one, NDT-Profile, is oriented toward the first phases of the life cycle: requirements, analysis, and design and also includes artifacts for the test phase. The second one is CADI, a tool to ensure the traceability between design and code.

In this section, both tools are introduced. After the tool presentations, a global vision of the practical solution and the connection between the two tools is described.

### 3.1. NDT-Profile

As mentioned earlier, NDT is a methodology based on the extension of UML with special artifacts to deal with the special Web characteristics. NDT has an associated tool, named NDT-Tool [9], which supports its complete life cycle and that permits the application of all its transformations. However, there are several areas where NDT-Tool has yet to be applied in real projects.

The first one is the life cycle. NDT-Tool, as NDT, only covers requirements and analysis and, in real projects, design, implementation, and testing are necessary phases. Furthermore, NDT-Tool is completely based on NDT and no changes, modifications, or adaptations of the methodology to real necessities are permitted. For instance, NDT-Tool only works with patterns and use case diagrams in requirements. Other suitable diagrams such as activity or sequence diagrams are not supported. NDT-Tool was, therefore, not a suitable solution for the practical application of NDT and for its fusion with Métrica.

For this reason, a new solution was defined. This solution was named NDT-Profile. NDT-Profile is the definition of NDT metamodels and rules in a commercial tool called Enterprise Architect.[4] Enterprise Architect is a tool to support development with UML and it permits the definition of formal extensions of UML. The profile of NDT for the requirement phase in Enterprise Architect is shown in Fig. 26.3, In this profile, all the specific artifacts of NDT can be observed. For instance, AC is an NDT actor and is defined as a formal extension of UML actors. In www.iwt2.org a complete definition of NDT-Profile can be obtained.

In NDT-Profile, other profiles are defined: the analysis profile, a profile for design obtained from the fusion with Métrica, and a test profile also obtained from our fusion with Métrica. In Enterprise Architect when a profile is defined, it can be exported as an xmi file. When this file is included in a project, the set of artifacts defined in the profile are included in the set of tools of Enterprise Architect and can be used easily. Thus, in our practical collaborations, companies include the NDT-Profile in their projects, and NDT artifacts can be used as classic UML elements.

[4] www.sparxsystems.com

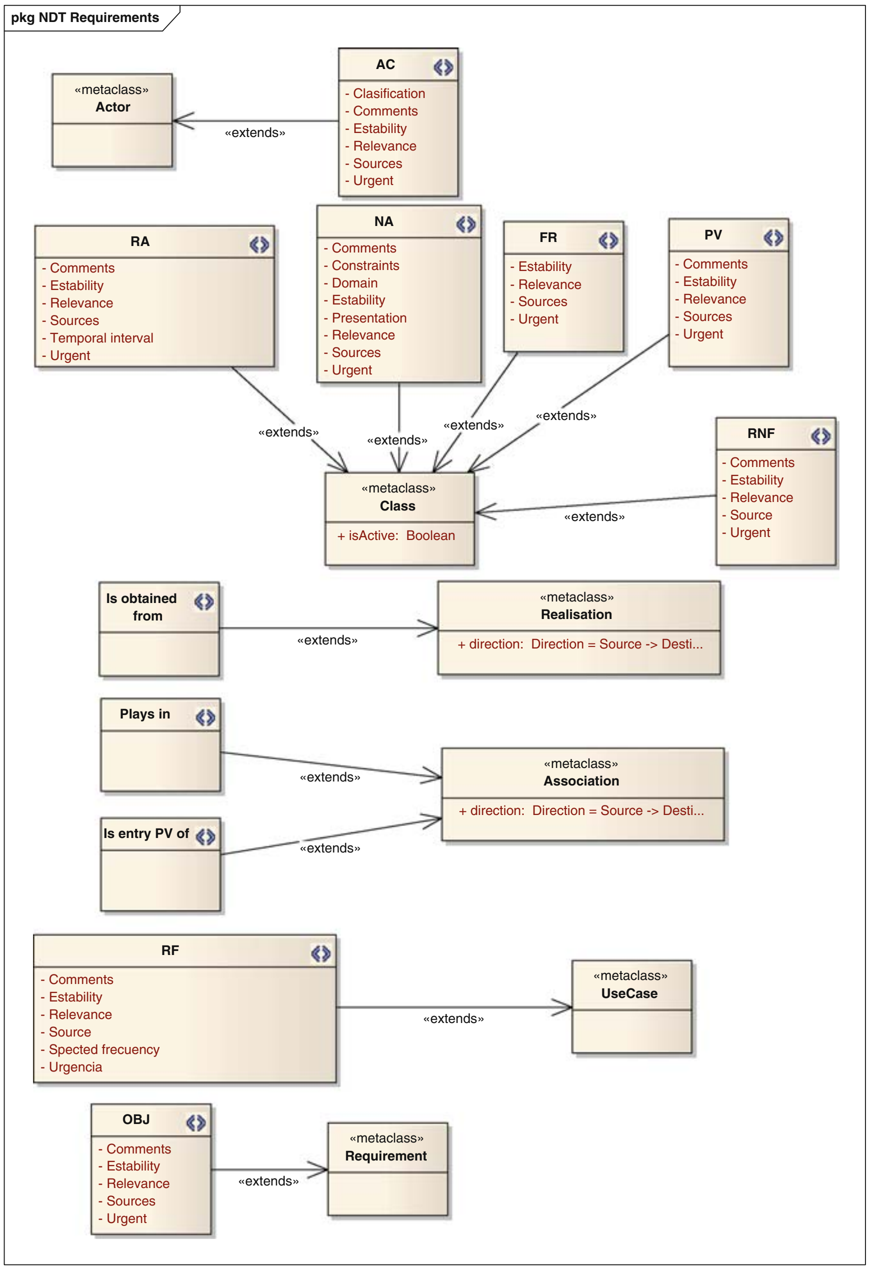

**Figure 26.3.** The NDT-Profile in enterprise architect for requirements.

### 3.2. CADI

NDT-Profile covers from the requirement to the design phase as well as the test phase, leaving a gap between them: implementation. CADI plays its role in this phase. Ever since UML reverse engineering tools appeared,[5] the common way to achieve traceability between the design and implementation phases was based on a human approach: the UML class diagrams created by the design team were manually

[5] http://uml.netbeans.org/

compared with those obtained from source code via reverse engineering. This approach has two main disadvantages. It is a highly time-consuming task which is also extremely error prone. Therefore, an automatic solution is needed. As always, there are several solutions for the automatization of this traceability task, each with its own advantages and disadvantages. The solution chosen by CADI is perhaps a very theoretical one, but is undoubtedly very effective and flexible: it is based on the ASTs (Abstract Syntax Trees) which underpin the theory of compilers.

ASTs [2] are well known in the theory of compilers and widely used [1]. Their main advantage for the purpose of this tool resides in the point that they do not make a direct but a symbolic representation of the code, thereby eliminating any inherent ambiguity in the codification process. ASTs allow all kinds of analysis to be carried out (syntactical, grammatical, and semantic) against the code; and these can be performed separately. ASTs create a clean interface between the code and the later phases that has to be accomplished. In this way, the process of traceability from the design phase to that of implementation is reduced (from a simplistic perspective) to the comparison of two ASTs, the one from design and the one from implementation. The latter is obtained directly, provided by the development team, and the former can be easily obtained by a feature that most software similar to Enterprise Architect offers: to generate source code from UML class diagrams.

These are the theoretical principles behind CADI. And even if CADI is implemented in Java, it is clear that the same procedures could be used with other computing languages. Moreover, CADI works with Java grammar although it is language neutral. It can be easily modified to work with any language on the condition that a grammar definition suitable for use by JJTree/JavaCC exists.[6]

CADI implementation: this is a command line tool written in Java that processes two directories of source code files recursively, one containing files from design and the other directory containing files from implementation. By implementing the Visitor Pattern, several methods are invoked while the source files are traversed, and information built in the form of ASTs is received. This method behavior can be modified by altering a system of rules defined in a properties file named "exclusion_rules.properties."

This technique gives the flexibility required for CADI to adapt to quality assurance at different levels of the project. This information provided by methods that implement the Visitor Pattern is used to generate detailed reports on the level of consistence between design and implementation. Reports are executed using a pluggable mechanism which loads at runtime. These reports are executed sequentially and appear in the file "output_config.properties." It provides a very simple, flexible, and powerful system for the generation of all kinds of reports.

At the time of writing this document, only the Java language is supported, but in the next version it is planned to implement a mechanism to handle pluggable syntax definition files, thereby allowing the use of CADI for different programming languages. CADI is fast, it can compare around 24,000 classes per minute using a single processor laptop, thereby allowing not only 100% traceability to be covered but also these checks to be performed as frequently as desired. CADI Java class files are fully documented and there is also a PDF document explaining how it was built (unfortunately both in Spanish). It has been satisfactorily applied in several projects since the end of 2007 and we are currently working in the English version of the tool.

## 4. The Practical Experience: The Diraya Project

The Diraya project is a complex system currently under development in Andalusia. The practical methodological environment presented in the previous section is used in this project and is described in [9].

Diraya is a system for the management of health information in Andalusia. It is divided into two systems: primary Diraya and specialized Diraya. The former is oriented to manage health information for primary health assistants. The latter offers the functionality to manage health information in hospitals and specialized centers.

[6] https://javacc.dev.java.net

The practical environment presented in this chapter is being applied in specialized Diraya. Here, six important companies are working for the Andalusian Health Government[7] to develop this system: Everis, Telvent, Indra, Accenture, Isoft, and Tecnova. The development team is composed of more than 80 people and the use of a suitable methodology and a suitable set of tools must be applied.

The structure of the tools is given in Fig. 26.4. In the requirement, analysis, design, and test phases, NDT-Profile is used. The whole development team uses Enterprise Architect with the special extension for NDT in their daily work.

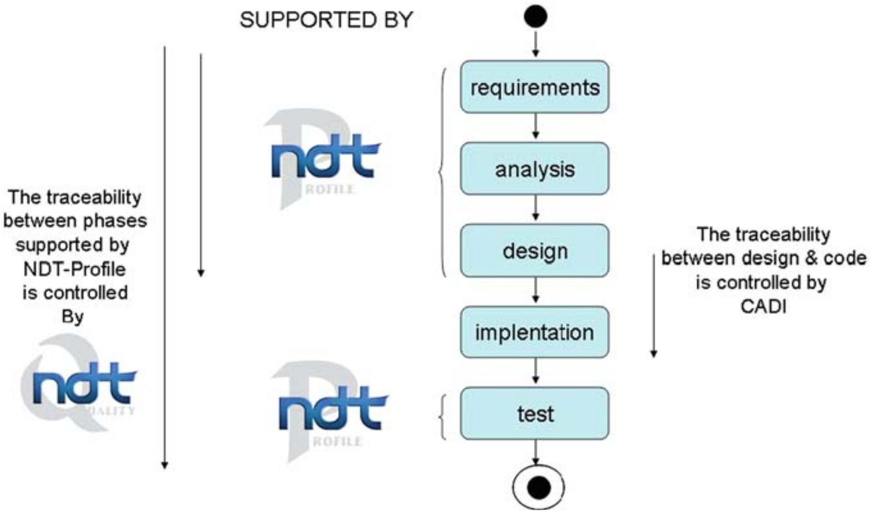
SUPPORTED BY
requirements
analysis
design
implentation
test
The traceability between phases supported by NDT-Profile is controlled By
The traceability between design & code is controlled by CADI
ndt PROFILE
ndt QUALITY

**Figure 26.4.** Tool solution for Diraya project.

When the design phase is finished, the code is generated in the Java language.

In order to improve and ensure traceability between design results and code, the CADI tool is used.

Furthermore, to assure traceability between requirements and analysis, analysis and design, and requirements and tests are going to be controlled by another new tool called NDT-Quality, which is currently under development and will be presented in the section for conclusions and future work.

This tool environment in Diraya is highly suitable for development support. The collaborative work of several companies, the communication between a highly developed team and the complexity of a system such as specialized Diraya can only be tackled by a mechanical and homogeneous environment such as the one presented.

This environment is being used in other projects in Andalusia and can be observed in detail in www.iwt2.org.

## 5. Conclusions and Future Works

This chapter presents a practical environment solution to apply model-driven techniques in the enterprise environment. We show how, by using the extension mechanisms of UML and formal metamodels, the two approaches, Métrica and NDT, can be fused in order to attain more suitable results. Furthermore, we provide a tool support solution to help in the application of this methodological environment.

[7] http://www.juntadeandalucia.es/servicioandaluzdesalud

As conclusions, the results of the practical applications are very positive. The theoretical fusion of approaches, metamodels, and the profile definition is transparent for the development team which works with only the Enterprise Architect interface. No theoretical lessons about model-driven paradigm have to be learnt by the development team, and the learning time is short since NDT-Profile and CADI offer a very intuitive and suitable interface.

Moreover, the assurance of traceability is a major improvement. In the project, traceability between design and code is 100% guaranteed with the use of CADI. As future work, we are working on the full application of NDT-Quality. This is a new tool developed by the NDT group which analyzes a project developed with NDT-Profile and controls the quality of this project and the traceability between phases. To this end, NDT-Quality controls some rules defined by the QVT transformations of NDT. This tool is fully developed and is starting to be used in specialized Diraya. With this incorporation, the quality assurance is greater. Additionally, the NDT group is working on two further tools. The first, named NDT-Translations, is an implementation of QVT transformations. To date, the development team has had to apply NDT transformations manually, which is not only time consuming but also prone to error. With the use of NDT-Translations the development time will be reduced and the number of manual mistakes will be lower. The second new tool is NDT-Reports. This tool is oriented toward obtaining suitable results from NDT-Profile. Enterprise Architect generates word and html results. However, the adaptation to specific results and format is a little complex and this tool attempts to facilitate this generation. Clearly, we wish to continue this relation with the enterprise environment for our future research. Conclusions obtained from practical applications are essential for the provision of suitable methodological environments.

## Acknowledgments

This research has been supported by the project QSimTest (TIN2007-67843-C06_03) and by the RePRIS project of the Ministerio de Educación y Ciencia (TIN2007-30391-E), Spain.

## References

1. V. Aho, Ravi Sethi, Jeffrey D. Ullman. "Compilers: Principles, Techniques, and Tools". ISBN: 978-0201100884. Addison Wesley, New York, 1986
2. E. Börger, R. Stärk, "Abstract State Machines: A Method for High-Level System Design and Analysis". ISBN: 978-3540007029. Springer, New York, 2003
3. S. Ceri, P. Fraternali, P. Bongio. "Web Modelling Language (WebML): A Modelling Language for Designing Web Sites". WWW9/Computer Networks 33, 137–157, 2000.
4. C. Cachero. "Una extensión a los métodos OO para el modelado y generación automática de interfaces hipermediales". PhD Thesis. University of Alicante, Alicante, Sapin, 2003.
5. Y. Deshpande, S. Marugesan, A. Ginige, S. Hanse, D. Schawabe, M. Gaedke, B. White. "Web Engineering". Journal of Web Engineering 1(1), 3–17, 2002. Rinton Press.
6. M.J. Escalona, J. Torres, M. Mejías, J.J. Gutierrez, D. Villadiego. The Treatment of Navigation in Web Engineering. Advances in Engineering Software 38(4), 267–282, 2007.
7. M.J. Escalona, J.J. Gutiérrez, J.A. Ortega, I. Ramos. NDT & METRICA V3-An Approach for Public Organizations Based on Model Driven Engineering WEBIST 2008. Proceeding of the 4th International Conference on Web Information Systems, Portugal, Vol. 1, ISBN. 978-989-8111-26-5, 2008.
8. M.J. Escalona, C.L. Parra, F.M. Martín, J. Nieto, A. Llergó, F. Pérez. "A Practical Example From Model-Driven Web Engineering Advance in Engineering Software". Springer Verlag, New York, Vol. 1, ISBN: 978-0-387-30403-8, 2008.
9. M.J. Escalona, G. Aragón. NDT: A Model Driven Approach for Web Requirements. IEEE Transaction on Software Engineering; United States (2008–05), Vol. 34, No. 3, pp. 377–390, ISSN: 0098-5589, 2008.
10. E. Insfrán, O. Pastor, R. Wieringa. "Requirements Engineering-Based Conceptual Modelling". Requirements Engineering Journal 7 (1), 2002.
11. N. Koch. "Software Engineering for Adaptive Hypermedia Applications". Ph. Thesis, FAST Reihe Softwaretechnik 12, Uni-Druck Publishing Company, Munich. Germany, 2001.
12. A. Kraus, N. Koch. "A Metamodel for UWE". Technical Report 0301, Ludwig-Maximilians-Universität München, January 2003.
13. OMG: MDA Guide, http://www.omg.org/docs/omg/03-06-01.pdf. V. 1.0.1, 2003.

14. OMG. Unified Modeling Language: Superstructure, version 2.0. Specification, OMG, 2005. http://www.omg.org/cgi-bin/doc?formal/05-07-04.
15. G. Rossi. "An Object-Oriented Method for Designing Hypermedia Applications". PHD Thesis. University of PUC-Rio. Rio de Janeiro. Brazil, 1996.
16. D.C. Schmidt. "Model-Driven Engineering". IEEE Computer, February 2006.
17. A. Vallecillo, N. Koch, C. Cachero, S. Comai, P. Fraternali, I. Garrigós, J. Gómez, G. Kappel, A. Knapp, M. Matera, S. Meliá, N. Moreno, B. Pröll, T. Reiter, W. Retschitzegger, J.E. Rivera, W. Schwinger, M. Wimmer, G. Zhang. MDWEnet: A Practical Approach to Achieving Interoperatiblity of Model-Driven Web Engineering Methods. 3rd Workshop on Model-Driven Web Engineering. MDWE 07, pp. 246–254, 2007.
18. P. Vilain, D. Schwabe, C. Sieckenius, "A diagrammatic Tool for Representing User Interaction in UML". Lecture Notes in Computer Science. UML'2000. York, England 2002.