Measuring the Quality of Model-Driven Projects with NDT-Quality

M.J. Escalona, J.J. Gutiérrez, M. Pérez-Pérez, A. Molina, E. Martínez-Force, and F.J. Domínguez-Mayo

Abstract Model-driven web engineering (MDWE) is a new paradigm which provides satisfactory results in the development of web software systems. However, as can be concluded from several research works, MDWE provokes traceability problems and the necessity of managing constraints in metamodel instances and transformation executions. The management of these aspects is usually executed manually in the most of MDWE approaches. Nevertheless, model-driven paradigm itself can offer suitable ways to manage them. This chapter presents NDT-Quality, an approach to measure the quality of web projects developed with NDT (navigational development techniques), and offers a view about the application of this tool in real web projects.

Keywords Web engineering · Model driven web engineering · Quality assurance · Tool support

1 Introduction

Model-driven engineering is a new paradigm that is being assumed by several research groups in the improvement of methodological approaches for web environment. UWE (UML web engineering) [12], WebML (web modeling languages) [2], and OOH [1] are only a few examples. MDWE is providing good results in this area, but for application to real projects some tools are necessary to assure the quality of results.

MDWE consists in the definition of a set of metamodels in each phase of the life cycle, followed by the establishment of a set of transformations between these metamodels which enable subsequent models to be derived.

For instance, Fig. 1, obtained from [14], shows a schema on the adaptation of standard MDA (model-driven architecture) [15] in web development.

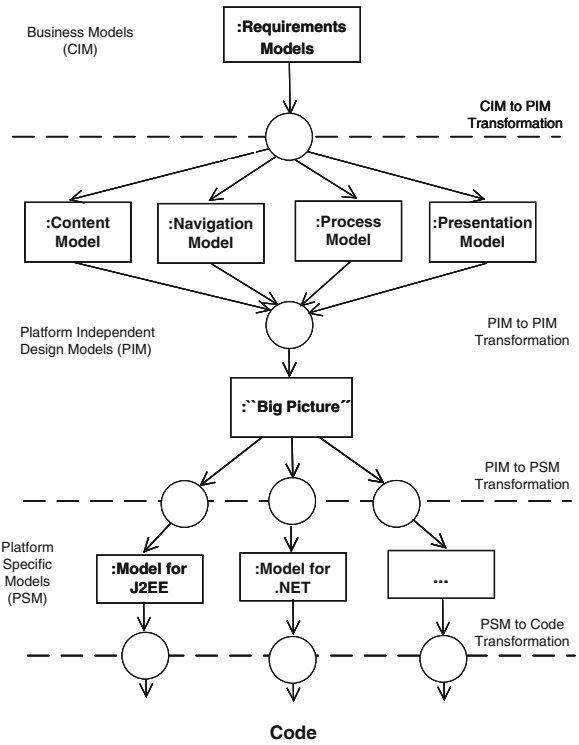

Fig. 1 Model-driven web engineering

In this environment, a set of metamodels on CIM (computer-independent model) level, requirements models, permit the capture of requirements information.

With CIM-to-PIM transformations, analysis models can be systematically obtained: content model, navigational model, etc. On the PIM (platform-independent model) level, some new transformations (PIM-to-PIM) can be applied in order to obtain design models.

Subsequently, in PSM (platform-specific model) models can also be obtained from the PIM level.

Finally, a code could be generated from PSM models with PSM-to-code transformations.

MDWE shows important advantages in software development. Transformations assure traceability among levels. Furthermore, the systematic generation of models based on early models can reduce the development time and, if suitable tools are defined, this process could even be automatic.

However, some questions arise. One of the most important is: What happened to the changes? For instance, if the analyst detects a problem and changes the model after the content model generation, then the traceability of requirements could be lost. To execute CIM-to-PIM, transformations may be considered again, but even a little change in some part may imply some delay or extra work.

The aim of this chapter is oriented toward solving these maintenance problems in MDWE. It presents a MDWE methodology, NDT (navigational development

techniques) [11], in Section 2. Section 3 introduces one of its associate tools, NDT-Quality. NDT-Quality manages the maintenance of traceability and the assurance of quality in model-driven applications. A real application in a public organization in Spain is given. Related work and conclusions are drawn in the final section.

2 NDT – Navigational Development Techniques

NDT [9] is a methodological web 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 that allow derivation relations to be created between models. NDT takes this theoretic base and enriches it the elements necessary for the definition of a methodology: techniques, models, methodological process, etc. in order to offer a suitable context for its application to real projects.

Figure 2 presents the life cycle of NDT. Initially, NDT only covered the requirement and the analysis phases.

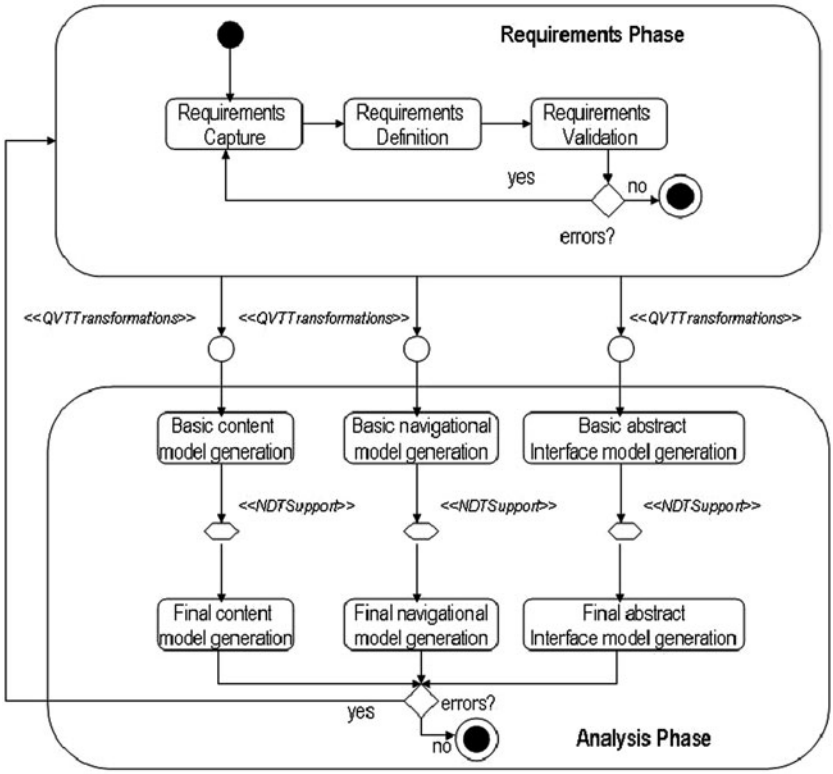

Fig. 2 Life cycle of NDT

The capture, definition, and validation of requirements are involved into the requirements phase. 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. To cope with each kind of requirement, NDT proposes the use of special patterns [4] and UML [18] techniques, such as the use of 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 by UWE notation [13].

The class diagram of UML and the navigational and the abstract interface of UWE have their own metamodels. From 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 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's 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 (object-oriented hypermedia design method) [19], 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 (see Fig. 1) is made, NDT presents a CIM in the requirements phase; a set of PIMs in the analysis phase; and a set of formal transformations among them.

NDT has been widely applied in practical environments 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.

3 NDT-Quality

The application of MDWE and transformations is difficult and quite expensive if it lacks in a set of tools which automate the procedure.

NDT-Suite[1] consists in a set of tools defined to support the development of web systems with NDT and it is composed of four tools:

1. *NDT-Profile:* This is a specific profile for NDT, developed using Enterprise Architect [10]. This tool offers an environment to define specific profiles, and NDT-Profile has adapted Enterprise Architect to support each artifact of NDT.
2. *NDT-Driver:* This is a tool to execute transformations of NDT. NDT-Driver is a Java-free tool which implements QVTTransformations (see Fig. 2) and allows analysis models to be obtained automatically from the requirements model.
3. *NDT-Quality:* This is a tool that checks the quality of a project developed with NDT-Profile.
4. *NDT-Report:* This is a tool that prepares formal documents in order to be validated by final users and clients. For instance, it enables the automatic generation of a requirements document with the format defined by clients.

3.1 The Necessity of NDT-Quality

One of the most relevant characteristics of NDT and NDT-Suite is their practical application. The real application of MDE provides an important source of knowledge for the improvement and adaptation of the methodology and its associate tools.

NDT-Quality was developed as a practical necessity when NDT-Profile started to be applied in real companies.

Although NDT-Profile offers a suitable environment to NDT and it manages the use of artifacts and UML and NDT constraints, development teams could still make errors and inconsistencies in the definitions of systems.

When a development team develops a system with NDT-Profile, they create requirements, classes, use cases, etc. with the tool box defined in Enterprise Architect to deal with NDT.

This product must be checked in order to assure two important parts:

1. The quality of the use of NDT in each development phase.
2. The quality of the traceability with the MDE rules of NDT.

In the first group, NDT-Quality checks the use of any artifact of the methodology. For instance, NDT says that each storage information requirements has to be named and described. NDT-Quality checks that every storage information requirement has a name and a description.

When a development team finishes the requirements phase and generates the analysis phase with NDT-Driver, they can make changes to adapt the results. Some of these changes are allowed in NDT, even though the traceability must be assured.

[1] All these tools and their manuals can be downloaded from www.iwt2.org. This chapter focuses on the presentation of NDT-Quality.

Therefore, NDT-Quality also checks this traceability. It compiles a set of rules to ensure that NDTSupport rules (see Fig. 2) are kept in final models.

In the enterprise environment a tool such as NDT-Quality is crucial. The execution of MDE transformations is only possible if metamodels are correctly implemented whereby the rules of the methodology are followed. NDT-Quality guarantees the quality in the application of NDT.

3.2 The Interface of NDT-Quality

NDT-Quality is completely based on NDT-Profile. A working group who wants this environment to be used has to define the project through NDT-Profile and then saves it in an Enterprise Architect file.

The interface of NDT-Quality is quite simple. Figure 3 shows the main interface of the tools. NDT-Quality, like all other NDT-Suite tools, is available in both Spanish and English.

In Project Name section, the name of the project must be introduced. With the search button, the Enterprise Architect file with the project must be selected and, with the set of checks on the left of the screen, the user can select which part of the project must be checked by NDT-Quality.

Although NDT only focused on the first phases of the life cycle, requirements, and analysis, a practical solution of the tool was prepared to work in the enterprise environment. It was presented in [8]. This extension of NDT covers all phases of the life cycle. For this reason, in the screen, NDT-Quality now provides the user with several options:

1. *Requirements, analysis, design, and tests:* If any of these checks are selected, NDT-Quality checks artifacts and NDT rules in this phase. For instance,

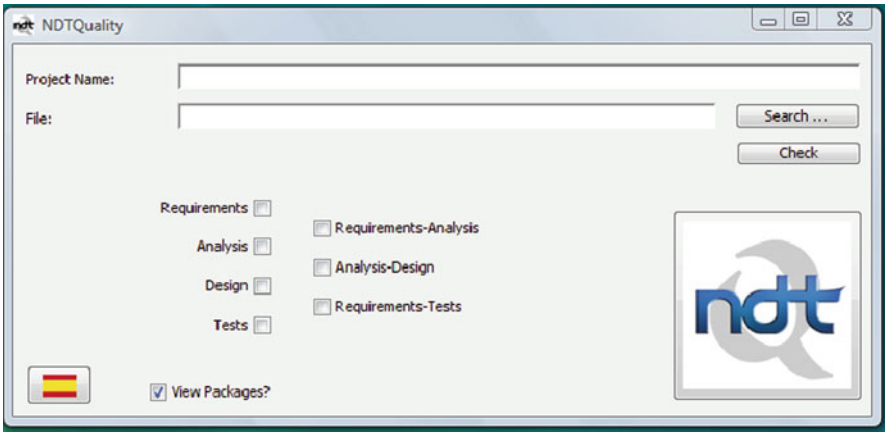

Fig. 3 NDT-Quality main interface

if we want to check the requirements phase, we must select "requirements check."

2. *Requirements analysis, analysis design, and requirements test:* These options enable the MDE traceability between phases to be checked. For instance, if we select requirements analysis, NDT-Quality checks the rules to assure the traceability between both the analysis metamodels and requirements metamodels.
3. *View package:* It is an option to see the package where an artifact with names is defined. When these options are selected, and with the check bottom pressed, NDT-Quality starts the checking process.

The response time in NDT-Quality is quite short. As is presented in Section 2.3, it evaluates the NDT rules on a database. For a project with about 100 requirements it takes only a few seconds.

The checked results are presented on a screen similar to that shown in Fig. 4.

For each check selected NDT-Quality presents tabs with the check results. For instance, in Fig. 4, the mistakes found in the requirements phase are presented in the first tab.

The columns of this report are the following ones:

Artifact: The artifact where the error was found.
Description: A description of the problem.
Criticality: An indication of the grade of importance of the mistake. They can be warning, error, or fatal error. With errors or fatal errors NDT-Driver cannot be used.
Package: This option is presented only if the check in Fig. 3 is selected. It shows the path in the NDT-Profile file where the artifact is located.

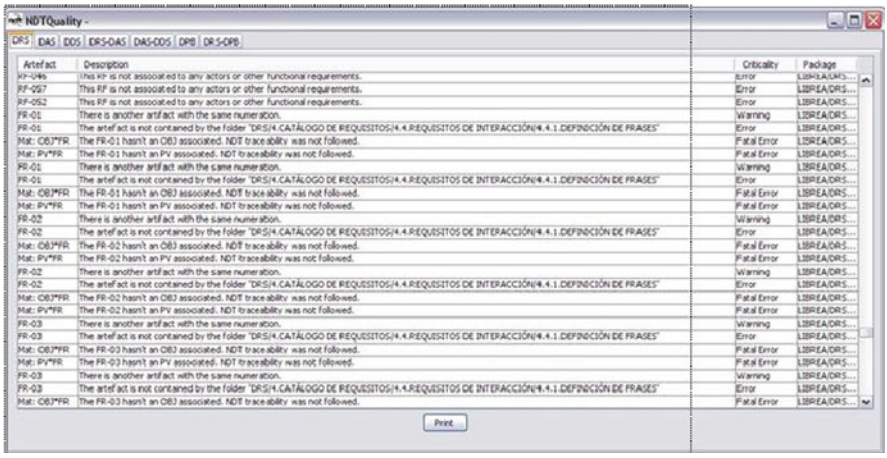

Fig. 4 NDT-Quality results screen

3.3 The Architecture of NDT-Quality

Enterprise Architect is supported by a database (Access, Oracle, or MySQL can be configured) with a specific relational structure.

When a development team uses NDT-Profile, each element defined in the project is stored in the Enterprise database.

NDT-Quality was developed using Java. NDT-Quality has a specific set of rules that must be applied in each phase and are relevant in the project.

3.4 Practical References

NDT-Suite has been applied to several real projects. In fact, nowadays, some public organizations like Culture Government in Andalusia [3] or Emasesa [5] are working with this tool environment to develop their web projects.

One of the first projects where NDT-Quality started to be used was the project for the digitalization and diffusion of the historical archive of the Medinaceli Duke Foundation. This project was developed by a company for the Culture Andalusia Government and it was aimed at spreading documental resources of this famous family via Internet.

This historical archive consists in a set of documental resources produced and received from any noble house, states, or historical patrimony that were included in Medinaceli family through marriage or family alliances.

This application contains three important modules:

1. *A module to manage any document:* It allows the introduction of new documents into the systems, their description, and location in the system. This functionality can only be executed by the administrator.
2. *A module to manage any search in the system:* With a historical researcher, a tourist, or anybody interested in this historical archive, who wants to search for something in the system, this module offers an efficient and a very intuitive way to manage any kind of search.
3. *A module to download documents from the system:* The system provides copies of documents for research and work purposes. Some special users can use this function and obtain information to include in their studies.

The first phase completed in the project was the requirements phase. In the first presentation of the NDT-Profile file, the company runs without NDT-Quality. The number of errors appears in the first row of Table 1. NDT-Quality was presented to the company in the representation of the requirements phase, the number of error decreased to only one warning. This implementation was quite simple and the requirements phase was accepted.

The company then used NDT-Driver in order to generate the analysis phase. In the first presentation of the analysis phase only three errors were detected.

Table 1 Number of errors detected in each phase

Phase	Warning	Errors	Fatal errors	Comments
Requirements v. 1.00	13	6	3	Without NDT-Quality
Requirements v. 2.00	1	0	0	With NDT-Quality
Analysis v. 1.00	0	3	0	MDE traceability problems
Analysis v. 2.00	0	0	0	

NDT-Quality was used and they solved their problems before presenting the results. These three errors (third row in the table) were produced by MDE traceability problems. Once it was explained that the relations must be kept, they solved the problem easily. In the design phase the number of errors was 0.

As can be concluded from the table, the number of errors was reduced with the use of NDT-Quality. This tool is highly suitable for measuring quality by the final client, in this example the Culture Government, or by their own software provider that without any additional cost can offer a more suitable result.

In other projects where NDT-Quality was used, the results were similar. The improvement in the results was very important since both providers and final users have the same objective and an automatic way to measure the quality of the project.

4 Related Works

Very few related work can be found on the web or in literature. In fact, although MDWE is being assumed by several web engineering methodologies, the use of tools to measure the quality of its application is scarce.

For instance, UWE methodology provides a set of metamodels for each of its phases [13]. It offers a set of transformations and a tool, named MagicUWE [6], oriented to support the MDE development with UWE. However, no special support to manage the quality of models developed is included.

Similarly for WebML, several metamodels were developed regarding this methodology [16, 20], and its transformations, mainly oriented toward code generation, offer a suitable and very powerful MDE environment. Nevertheless, the measure of quality was something last.

In fact, in the last MDWE workshop [17], the issue of measuring the quality when MDWE is applied was established as a high-priority procedure.

In the literature, several approaches to measure the quality of software models can be found, for instance, SDMetrics [21], an environment with a set of metrics to measure the quality of each UML model. In fact, none of them offers special metrics for the MDE paradigm.

5 Conclusions

This chapter presents NDT-Quality, a tool to measure the quality of the application of a MDE methodology, NDT.

The chapter presents the problem in Section 1 and then introduces NDT and details of NDT-Quality. It also describes a global view of how NDT-Quality reduces project mistakes.

The scarcity of research in this area is pointed out. There are several MDWE methodologies and standard frameworks with guides to measure the quality of the system. However, the use of metrics and methods of measuring the quality of MDE applications needs further study.

As future work, the evolution of NDT and NDT-Suite is our highest priority. With the continuous feedback obtained from our practical applications, these tools are frequently being improved.

For instance, one important aspect is to adapt our methodological environment to technological evolution projects. That is, our environment is oriented toward requirements and continues with the classic life cycle. However, what happens when an old application, without documentation or models, needs to be migrated into new technology? or what happens to the maintenance of a project? and how can knowledge be reused and kept if only a part is to be changed?

These questions open new research lines to better our tools and our methodology.

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. C. Cachero. Una extensión a los métodos OO para el modelado y generación automática de interfaces hipermediales. PhD Thesis. University of Alicante. Alicante, Spain 2003.
2. S. Ceri, P. Fraternali and P. Bongio. Web Modelling Language (WebML): a modelling language for designing web sites. *Conference WWW9/Computer Networks*. 33(1–6), pp. 137–157. Mayo 2000.
3. Consejería de Cultura. Junta de Andalucía. www.juntadeandalucia.es/ccul.
4. Durán, B. Bernárdez, A. Ruiz and M. Toro. A requirements elicitation approach based in templates and patterns. *Workshop de Engenharia de Requisitos. Buenos Aires*, Argentina. 1999.
5. Emasesa. Empresa Municipal de Aguas de Sevilla. http://www.aguasdesevilla.com.
6. MagicUWE. http://www.pst.informatik.uni-muenchen.de/projekte/uwe/toolMagicUWE.html.
7. M.J. Escalona, J.J.Gutierrez, D.Villadiego, A. León and A.H. Torres. Practical experience in web engineering. *Advances in Information System Development*. New Methods and Practice for the Networked Society. ISBN: 13-978-0-387-70801-0.V.2. pp. 421–434. 2007.
8. M.J. Escalona, J.J. Gutiérrez, J.A. Ortega, and I. RamosHYPERLINK "publicaciones.php?id=81"NDT & METRICA V3. An approach for public organizations based on Model Driven Engineering WEBIST 2008Proceedings of the 4th International conference on web information systems; Portugal (2008), Vol. 1, pp. 224–227, ISBN: 978-989-8111-26-5
9. M.J. Escalona, M. Mejías, J. Torres, and A.M. ReinaHYPERLINK "publicaciones.php?id=140"The NDT development process, Lecture Notes in Computer Science. Springer Verlag; Alemania (2003), Vol. 2722, pp. 463–467, ISBN: 0302–9743, 2003
10. Enterprise Architect. www.sparxsystems.com
11. M.J. Escalona and G. Aragón. NDT A Model-Driven approach for Web requirements. *IEEE Transaction on Software Engineering*. 34(3). pp. 370–390. 2008.

12. N. Koch. Software engineering for adaptive hypermedia applications. Ph. Thesis, FAST Reihe Softwaretechnik, Vol. 12, Uni-Druck Publishing Company, Munich. Germany. 2001.
13. N. Koch. Transformation techniques in the model-driven development process of UWE. International Conference On Web Engineering; Vol. 155. Workshop proceedings of the sixth international conference on Web engineering. Palo Alto, California. WORKSHOP SESSION: *Second international workshop on model driven web engineering (MDWE'06)*, Article No.: 3. 2006 ISBN:1-59593-435-9, 2006.
14. N. Koch, G. Zhang and M.J. Escalona. Model transformations from requirements to web system design. *ACM International Conference Proceeding Series. Proceedings of the 6th International Conference on Web Engineering (ICWE 2006)*. Ed. ACM. pp. 281–288. 2006.
15. OMG: MDA Guide, http://www.omg.org/docs/omg/03-06-01.pdf. Version 1.0.1. 2003.
16. N. Moreno, P. Fraternali and A. Vallecillo. A UML 2.0 profile for WebML modelling. *II International Workshop on Model-Driven Web Engineering*. Palo Alto, California. 2006.
17. MDWE Workshop. http://mdwe2008.pst.ifi.lmu.de/.
18. OMG. *Unified Modeling Language: Superstructure*, version 2.0. Specification, OMG, 2005. http://www.omg.org/cgi-bin/doc?formal/05-07-04.
19. G. Rossi. An object-oriented method for designing hypermedia applications. PhD Thesis. University of PUC-Rio. Rio de Janeiro. Brazil. 1996.
20. Schauerhuber, A., Wimmer, M. and Kapsammer, E. 2006. Bridging existing web modeling languages to model-driven engineering: a metamodel for WebML. *2nd International Workshop on Model-Driven Web Engineering*, Palo Alto, California.
21. SDMetrics. http://www.sdmetrics.com/.