NDT-GLOSSARY
A MDE Approach for Glossary Generation

J. A. García-García, M. J. Escalona, C. R. Cutilla and M. Alba
Department of Computer Languages and Systems, University of Seville, Spain

Keywords: Model-Driven Engineering, Glossary Generation, NDT, Technical Validation of Requirements.

Abstract: This research paper is contextualized within the paradigm of Model-Driven Engineering (MDE) and it is specifically related to NDT. NDT is a methodology included within the MDE paradigm. The aim of this paper is to present a software tool to facilitate the work of requirements engineers during the requirements validation in a software project. The requirements validation activity takes place within the requirements phase of the life cycle in a software project. The developed tool is called NDT-Glossary and it implements an automatic procedure to generate, from the requirements defined in a project developed with the NDT methodology, the first example of the glossary of terms for this project.

1 MDE INTRODUCTION

The Model Driven Engineering paradigm (MDE) came up in order to tackle the complexity of platforms and the inability of third generation languages to relief this complexity and effectively express the domain concepts of the problem. This new paradigm, apart from raising the level of abstraction, intends to increase automation during the life cycle of software development.

Web Engineering works in a field which allows the successful implementation of this paradigm. The application of MDE in this domain is called Model-Driven Web Engineering, MDWE. In fact, the Model-Driven Engineering paradigm is being assumed by several research groups to improve the methodological proposals oriented to the Web. UWE (UML Web Engineering) (Koch, 2001), WebML (Web Modelling Languages) (Ceri, 2000) and OOH (Chubb, 2003) are some of the examples.

Model-Driven Engineering works, as the primary form of expression, with definitions of models and transformation rules among these models which entail the production of other models. Every model corresponds to a phase of the life cycle and is generally specified by means of UML modelling language.

Standardization was necessary in order to implement this new paradigm in real projects. OMG presented MDA, which stands for Model-Driven Architecture (MDA, 2003), as a platform to support the paradigm of Model-Driven Engineering.

MDA proposes to base the software development on models which make transformations be performed to generate code or another model with characteristics of a particular technology (or lowest level of abstraction). As transformations go on, it may be noticed that the models become more concrete and the abstract model changes into another one compatible with a particular technology or platform. MDA is based on four types of levels or models:

- *The CIM level* (Computation-Independent Model) is considered the highest level of business model and the most abstract level. It focuses on requirements specification and intends that anyone who knows the business and its processes can understand a CIM model, as this avoids any contact with the specific system.
- *The PIM level* (Platform-Independent Model) represents the business process model and system structure, without any reference to the platform on which the application will be implemented. It is usually the entry point for all the support tools for MDA.
- *The PSM level* (Platform-Specific Model) specifically relates to the platform where the system will be implemented, for example, with

DOI: 10.5220/0003462401700175
In *Proceedings of the 13th International Conference on Enterprise Information Systems* (ICEIS-2011), pages 170-175
ISBN: 978-989-8425-55-3

operating systems, programming languages or middleware platforms, among others.

- Finally, the Code level refers to the codification and suitable implementation of the system.

MDWE offers important advantages in software development. It specifically provides relevant results in Web-oriented projects.

The systematic generation of models based on previous models assures traceability of levels and can potentially reduce development time. In addition, if suitable tools were defined, this process could even be automatic.

However, for a full development of MDWE in real projects, it requires software tools that may ensure the quality of the results when using this paradigm.

2 NDT – NAVIGATIONAL DEVELOMENT TECNIQUES

The proposed methodology NDT (Escalona & Aragon, 2008), acronym for Navigational Development Techniques, belongs to the paradigm of Model-Driven Engineering, MDE.

Initially, NDT dealt with the definition of a set of formal metamodels for the requirements and analysis phases. In addition, NDT defined a set of derivation rules, stated with the standard QVT (Query-View-Transformation) (OMG, 2008), which generates the analysis models from requirements model. QVT standard defines a declarative and imperative language proposed by the OMG for model transformation in the context of Model-Driven Engineering.

Nowadays, NDT defines a set of metamodels for every phase of the life cycle of software development: the feasibility study phase, the requirements phase, the analysis phase, the design phase, the implementation phase, the testing phase, and finally, the maintenance phase. Besides, it states new transformation rules to systematically generate models.

Although the life cycle is represented sequentially, the NDT process is not sequential, as many of its parts can be reviewed to correct errors previously detected.

The first phases are briefly described below.

The first main phase is *the Requirements phase*. Its main goal deals with defining the requirements phase in order to list the catalogue of requirements which contains the needs of the system to be developed. It is divided into a series of activities: capture, definition and validation of requirements. See Figure 1.

NDT classifies project requirements according to their nature: information storage requirements (storage requirements and new natures), functional requirements, actor requirements, interaction requirements, and non-functional requirements.

In order to define them, NDT provides special patterns and UML techniques (UML, 2005), such as the use cases technique for functional requirements specification. The use of patterns or templates to define every condition offers a structured and in-depth description. Moreover, the fact that some fields of such patterns only admit particular values makes results be obtained systematically in the remaining life cycle process of NDT.

Once the requirements specification phase has been completed and the catalogue of system requirements has been drafted and validated, NDT defines derivation rules to generate the system test model and the analysis phase models. Figure 1 shows all these transformations through the stereotype *«QVTTransformation»*.

NDT conceives the next phase, *the Testing phase*, as an early phase of software life cycle and proposes to carry it out together with the remaining phases.

The testing phase is divided into the following activities: drawing up the test plan, a parallel activity to the analysis phase**;** environment specification and test plan design a parallel activity to the design phase; and implementation of test plan, a parallel activity to the system construction and implementation phase.

NDT proposes three models in this phase (see Figure 1): implementation tests model, system tests model and acceptance tests model. Every model is described by means of the use case diagrams of UML.

The system tests model is the only one that can be generated systematically.

NDT proposes derivation rules to generate the basic model of system tests from the functional requirements defined in the requirements phase. The team of analysts can perform transformations in order to enrich and complete this basic model**.**

The following phase, the Analysis phase, will include the resulting products from the analysis, definition and organization of requirements in the previous phase.

At Analysis phase, NDT proposes four models (see Figure 1): the conceptual model, which represents the static structure of the system; the

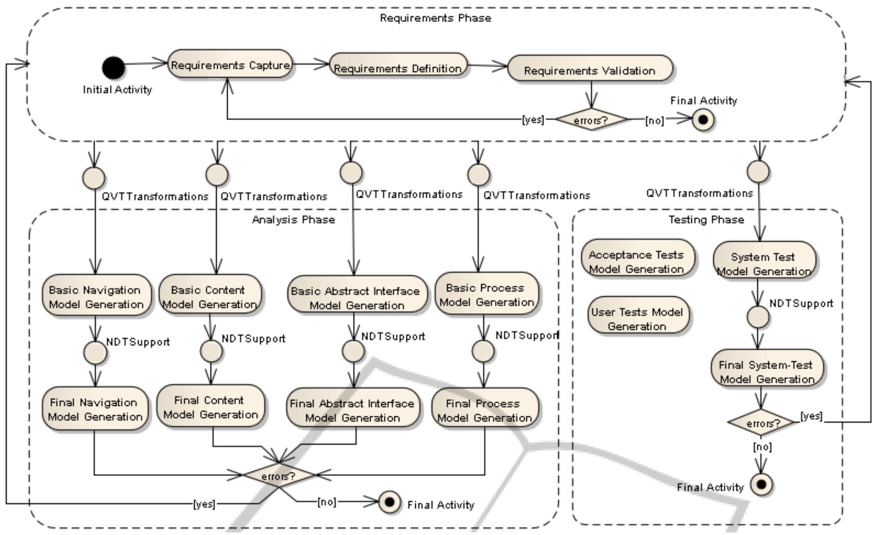
Requirements Phase
Initial Activity
Requirements Capture
Requirements Definition
Requirements Validation
Final Activity
[yes]
errors?
[no]
QVTTransformations
QVTTransformations
QVTTransformations
QVTTransformations
QVTTransformations
Analysis Phase
Testing Phase
Basic Navigation Model Generation
Basic Content Model Generation
Basic Abstract Interface Model Generation
Basic Process Model Generation
Acceptance Tests Model Generation
System Test Model Generation
User Tests Model Generation
NDTSupport
NDTSupport
NDTSupport
NDTSupport
NDTSupport
Final Navigation Model Generation
Final Content Model Generation
Final Abstract Interface Model Generation
Final Process Model Generation
Final System-Test Model Generation
errors?
[yes]
[no]
Final Activity
errors?
[yes]
[no]
Final Activity

Figure 1: NDT Transformations from Requirements to Analysis and from Requirements to Testing model.

process model, which represents the functional structure of the system; the navigation model, which shows how users can navigate through the system and the abstract interface model, a set of prototypes of the system interface.

The transition between the requirements model and the analysis model is standardized and automated and it is based on QVT transformations, which change the concepts of requirements metamodels to design the first versions of the analysis models. These models are known in NDT as basic models of analysis. For example, the basic conceptual model of analysis is obtained from the storage requirements defined during the requirements phase.

Thereafter, the team of analysts can transform these basic models to enrich and complete the final model of analysis. As this process is not automatic, the expertise of an analyst is required. Transformations are represented in Figure 1 through the stereotype *«NDTSupport»*. To ensure consistency between requirements and analysis models, NDT controls these transformations by means of a set of defined rules and heuristics.

To sum up, NDT offers an environment conducive to the development of Web systems, completely covering life cycle of software development. In fact, NDT has been applied in many practical environments and has succeeded due to the application of transformations among models, which has reduced development time (Escalona, 2007).

3 NDT-GLOSSARY

The application of MDWE and, particularly, the application of transformations among models may become monotonous and very expensive if there are no software tools that automate the process.

To meet this need, NDT has defined a set of supporting tools called NDT-Suite. In this suite is included NDT-Glossary. Currently, the suite of NDT comprises the following free Java tools:

- *NDT-Profile* is a specific profile for NDT, developed using Enterprise Architect (Enterprise Architect, 2010). This tool offers the chance of having all the artifacts that define NDT easily and quickly as they are integrated within the tool Enterprise Architect.
- *NDT-Quality* is a tool that automates most of the methodological review of a project developed with NDT-Profile. It checks both, the quality of using NDT methodology in each phase of software life cycle and the quality of traceability of MDE rules of NDT.
- *NDT-Driver* implements a set of automated procedures that enables to perform all transformations MDE among the different models of NDT that were described in the previous section.
- NDT-Prototype generates a set of XHTML prototypes from the navigation models, described in the analysis phase, of a project developed with NDT-Profile.

NDT-Glossary implements an automated procedure that generates the first instance of the glossary of terms of a project developed by means of NDT-Profile tool. The glossary of terms is generated from the requirements defined in the project. Among the different types of requirements that define NDT, the only ones that are necessary to include in the glossary are both, the storage and actor requirements:

a. Storage requirements should be registered in the glossary because they set the relevant and inherent concepts for the domain of the system being modelled.

b. Actor requirements must be registered in the glossary because they indicate the roles that each user can play within the business being modelled. Thus, the project team would have a clear definition of each of the possible roles that a user can play.

3.1 The Necessity of NDT-Glossary

In Software Engineering and in Web Engineering specifically, it is very important to carry out an exhaustive work of elicitation of requirements that meets the needs of end users and customers and ensures the quality of the developed system.

In the information system development process, led or not to the Web, the development team faces up the arduous task of defining the system requirements. It is a complex process since it must identify the requirements the system must meet in order to guarantee the fulfilment of the end users and customers' requests.

The requirements specification process is divided into three main activities: requirements elicitation, requirements definition and requirements validation.

The process begins with the completion of the activity of capturing or elicitation of requirements. In this activity, the analyst team extracts the needs, both explicit and implicit, of customers and users and their expectations for the system which will be developed.

System requirements are taken out from the information provided by users and customers. From this information, the analyst team elaborate the preliminary requirements catalogue. Subsequently, the process enters an iterative validation sub-process of the requirements catalogue. The validation process ends with the final version of the requirements catalogue.

During the project development, some terminology problems may take place mainly due to the different people participating in it: customers, end users, project managers, designers or engineers, among others. This problem is highlighted in Web information systems projects since their development teams are usually more multidisciplinary.

This problem is becoming more present during the validation of the requirements catalogue. One of the most common techniques for this activity, and also one of the easiest to apply, is the glossaries technique, which allows registering the knowledge acquired on the problem domain or universe of discourse. In addition, glossaries allow this knowledge is shared by all participants in the project. Basically, the glossary is a dictionary which defines and collects the most important and critical concepts to be used during the software project development. The objective pursued is to avoid ambiguities when using concepts of problem domain during the different phases of life cycle in the software project.

Each term is represented in the glossary with a couple of values: *Name* and *Description*. To preserve the integrity, the glossary cannot contain two terms with the same *Name*.

The technical glossary is not a full technical validation of requirements in itself, but it is useful for finding inconsistencies in vocabulary during the requirements phase.

All glossaries of terms must verify simultaneously two rules (Leite, 1993): the Principle of Circularity and the Principle of Minimum Vocabulary. The former establishes that a glossary should be as AutoContent as possible. Therefore, it ensures that all terms are related and does not exclude knowledge of the universe problem from the glossary. Nevertheless, the latter points out those requirements should mainly be expressed by means of concepts included in the proper glossary. Thus, the glossary will be as understandable as possible.

In conclusion, it is necessary for engineers to gather and define the more relevant and critical concepts in the system. Furthermore, the use of a common language reduces the risk of ambiguities and facilitates communication between users and analysts.

3.2 The Interface of NDT-Glossary

The interface of NDT-Glossary is quite simple and intuitive. Figure 2 shows the main interface of the tool. In the *«EAP Project»* Section, you must indicate the path to a project developed using NDT-Profile tool. Once the project is selected, the tool automatically suggested both the name of the project (Section *«Name Project»*) and the path where the reports generated later are saved by the tool (Section *«Report Output Directory»*). If appropriate, the user may change the data manually before carrying out the expected operation.

Then, after entering the above data, the user can generate both, the glossary of terms and a report with the terms already stored in an NDT-Profile project, by only clicking either on *«Build Glossary»* or *«Build Report»*.

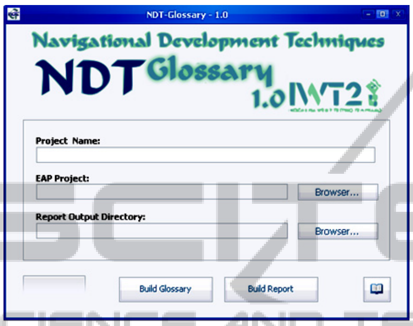

Figure 2: NDT-Glossary main interface.

3.3 The Architecture of NDT-Glossary

Enterprise Architect is based on the database model entity relationship. This model is implemented on different management systems databases: Microsoft Access, Oracle or MySQL. Thus, every element defined in a project developed with the NDT-Profile tool is stored in the database project.

NDT-Glossary tool was designed according to *the MVC design pattern*. MVC is an acronym for Model-View-Controller. This design pattern suggests that development should be divided into three layers: the presentation layer, corresponding to the graphical user interface; the business layer, which implements all the business logic of the tool, for instance, all transformations between NDT models, and data access layer. Additionally, some other patterns are used; the strategy pattern, the *Template Method Pattern*, and the combination of *Data Access Object Pattern* (DAO) with the *Singleton Pattern*.

Finally, the data access layer was implemented by means of the API provided by the company that developed Enterprise Architect, *SparxSystems*.

3.4 Enterprise Experiences

In the last ten years, NDT and NDT-Suite acted in a high number of real projects. In fact, they are currently used in several projects carried out by different companies either public or private and big or small.

Particularly, NDT was also widely applied in the e-health environment. In 2006, Alcer Foundation (Alcer) used it within the system to manage the degree of handicap. In this project, NDT-Suite was not fully developed and we used a previous tool, named NDT-Tool (Escalona, 2007).

However, this project is mentioned because it was the seed for detecting the necessity of NDT-Glossary. The medical systems environment works with a very specific terminology and the project caused a high number of inconsistence that could only be solved by elaborating a glossary manually.

Some years later, NDT-Suite was used in other e-health system, named Diraya (Escalona, 2008) which is a very complex system. The requirements phase was developed by a group of six companies with a high number of analysts. Each company was expert in a concrete aspect of Diraya.

The use of NDT-Profile and NDT-Glossary was essential to guarantee the unification of criteria in this multidisciplinary development team.

4 RELATED WORK

There are many papers related to technical validation of requirement on the Web or in the Literature. However, the validation techniques presented in these publications are applied manually. In fact, although MDWE is being assumed by several methodologies focused on Web Engineering, the use of tools to facilitate validation of requirements captured during the different interviews with clients and users in the first phase of a software life cycle is very low.

From a practical standpoint, the glossary of terms technique is an effective and efficient technique to complete the vocabulary of a project, although sometimes it requires a more complex glossary and even ontologies or thesaurus must be defined. Both, the glossaries technique and the ontologies or thesaurus technique not only allow defining concepts of the problem domain, but also relationships among concepts.

In the research (Escalona, 2005) technique called fuzzy thesaurus (Mirbel, 1995) (Mirbel, 1997) has been adapted to get the benefits of the definition of a thesaurus in NDT.

The use of this technique offers the possibility of finding similarities and semantic conflicts in different class diagrams with the aim to integrate them into a unique class diagram.

5 CONCLUSIONS

In Web Engineering in particular, it is very important to carry out an exhaustive work of elicitation of requirements that meet the needs of end users and customers in order to ensure the quality of the developed system.

With this aim, the technical team should be able to identify the customers and clients' needs, both explicit and implicit, and their expectations about the system under development. System requirements are taken out from the information provided by users and customers.

The nature of the problems associated with this activity is primarily social and psychological rather than technological. For example, among others, there are communicative problems caused by using different vocabularies, or even motivated by a different culture, or problems due to cognitive limitations for not knowing the problem domain.

There are several specific techniques to mitigate these problems substantially during the different activities related to the specification of the project requirements. For the activity of requirements capture, there are techniques such as interviews. For example, Joint Application Development is a popular exploratory technique that includes users as active participants in the development process. Some others are brainstorming or questionnaires.

The most common techniques for requirements validation activity in Web Engineering are interviews or walk-through, audits and glossaries.

The research work presented in this document is framed in this context. Particularly, this paper focuses on the glossary of terms technique as a suitable technique to carry out the validation requirements of a project developed by means of NDT methodology.

This paper presents NDT-Glossary: A tool that implements a set of QVT transformations to systematically generate the glossary of terms of a project which deals with Web information system developed using the NDT methodology. All of it is focused on the perspective of the paradigm of Model- Driven Engineering.

ACKNOWLEDGEMENTS

This research has been supported by the project QSimTest (TIN2007-67843-C06_03) and by the Tempros project (TIN2010-20057-C03-02) of the Ministry of Education and Science, Spain.

REFERENCES

Alcer. Federación Nacional de Asociaciones para la lucha contra las enfermedades renales. *www.alcer.org*. Accessed February 2011.

Cachero, C., 2003. Una extensión a los métodos OO para el modelado y generación automática de interfaces hipermediales. Thesis.

Ceri, S., Fraternali, P., Bongio, P., 2000. Web Modelling Language (WebML): A Modelling Language for Designing Web Sites.

Enterprise Architect, 2010. *www.sparxsystems.com*

Escalona, M. J., Torres, J., Mejías, M., Jurado, M. C., Fillerat, L. L., 2003. NDT: Navigational Development Techniques.

Escalona, M .J., 2004. Modelos y Técnicas para la Especificación y el Análisis de la Navegación en Sistemas Software.

Escalona, M. J., Koch, N., 2004. Requirements Engineering for Web Applications: a Comparative Study. Journal of Web Engineering.

Escalona, M. J., Cavarero J. L., 2005. Techniques to Validate Requirements in NDT.

Escalona, M. J., Aragón, G., 2008. NDT: A Model-Driven Approach for Web Requirements.

Escalona, M. J., Gutierrez, J. J., Villadiego, D., León, A., Torres, A. H., 2007. Practical Experience in Web Engineering. Advances in Information System Development. New Methods and Practice for the Networked Society.

Escalona, M. J., Aragón, G, 2007. NDT-Tool. A Model-Driven tool to deal with Web Requirements. ACM/IEE International Conference on Model Driven Engineering Languages and Systems. USA.

Escalona, M. J., Parra, C. L., Martín, F. M., Nieto, J., Llergó, A., Pérez P, 2008. A practical example for Model-Driven Web Engineering. Information System Development. Challenges in Practice, Theory and Education Springer Science + Business Media LCC. Vol. 1. pp. 157-168.

Leite, J. C. S. P., 1993. Eliciting Requirements Using a Natural Language Based Approach: The Case of the Meeting Scheduler Problem.

Mirbel, I., 1995. A Fuzzy Thesaurus for Semantic Integration of Schemes

Mirbel, I., 1997. Semantic Integration of Conceptual Schemas.

OMG, 2003. MDA Guide of OMG. Version 1.0.1, *http://www.omg.org/docs/omg/03-06-01.pdf*

OMG, 2008. Documents Associated with Meta Object Facility (MOF) 2.0 Query/View/Transformation, *http://www.omg.org/spec/QVT/1.0/*

Schmidt, D. C., 2006. Model-Driven Engineering. Published by the IEEE Computer Society vol 39 nº 2.

UML, 2005. Unified Modeling Language: Superstructure. Specification, OMG, 2005. *http://www.omg.org/cgi-bin/doc?formal/05-07*