Applying Model-Driven Paradigm for the Improvement of Web Requirement Validation

Gustavo Aragon, M. J. Escalona

IWT2 Group, University of Seville
ETS Ingeniería Informática, Av. Reina Mercedes S/N, 41012 Seville, Spain
gustavo.aragon@iwt2.org; mjescalona@us.es

Jose R. Hilera, Luis Fernandez-Sanz

University of Alcala, ETS Ingeniería Informática, Campus Universitario
Crta. Barcelona KM. 33.6, 28871 Alcalá de Henares, Madrid, Spain
jose.hilera@uah.es; luis.fernandezs@uah.es

Sanjay Misra

Department of Computer Engineering, Faculty of Engineering, Atilim University
Kızılcaşar Mh., 06836 Incek, Ankara, Turkey, smisra@atilim.edu.tr

Abstract: This paper proposes an approach for Web requirements validation by applying the model-driven paradigm in classical requirements validation techniques. In particular, we present how the Navigational Development Techniques (NDT) approach exploits the model-driven paradigm to improve its requirements validation task by exploring tool cases that systematize or even automate the application of requirements validation techniques. Our solution is validated by applying it in a real industrial environment. The results and the learned lessons are presented accordingly.

Keywords: Requirement; validation techniques; Web-application; NDT; Model-driven paradigm

1 Introduction

The requirement phase is the most critical phase of the software development process. The requirements phase consists of several different types of activities, starting from the requirement elicitations to the validation and management of the requirements. Web-engineering follows the principles and concepts of software

engineering in developing the Web-applications. Further changes in requirements and short time lines [4] are inherent features of web applications. It makes the task of requirement engineering for Web engineering more difficult. Escalona and Koch [2] found that the requirement phase is poorly managed in Web engineering methodologies [16]. Although the observation [2] is approximately seven years old, today's situation is not significantly different.

In recent years, although some attempts have been made in developing requirement phases for web-applications, they still need some improvements [3]. Further, requirements must go through the validation process. The requirements validation is defined as the part of the software engineering activity where requirements are valued, analyzed and reviewed with end-users and clients [4], [5]. In this task, the development team should guarantee that requirements are correctly structured and defined and all of them are detected. This task is usually carried out by analysts, end-users, clients and the remaining team members, who work in conjunction to assure consistent requirements. A variety of proposals for requirements validation (in general) [7-16], for Web approaches [17-21] and for using testing as a validation technique [22-24] can be found in the literature. Further, reviews, audits or prototypes are some of the techniques commonly used for this task. However, they are difficult to apply due to development time constraints, communication problems or lack of suitable tools, among other reasons. As previously stated, in the Web Engineering environment the situation of requirement validation is even more complicated [3]. In Web Engineering, end-users and clients are usually unknown, and some characteristics of the Web environment, for instance complex navigation systems, complex interface, security aspects or a heterogeneous development team, significantly complicate the task. This paper sets out some solutions to Web requirements validation through a model-driven approach. In particular, this paper analyzes how the model-driven paradigm helps to reduce time and cost on Web requirements and illustrates this idea with the case of the Navigational Development Techniques (NDT) methodology [6]. NDT is an approach supporting the requirements and analysis phases in Web engineering using the model-driven paradigm. In the recent years, NDT have improved some aspects, mainly focused on the use of the model-driven paradigm, to support the development. In this line of action, this article analyzes how requirements validation can be improved and how NDT can be enriched to make more systematic this complex task. It deeply presents techniques supported by NDT and the solution offered in its case, as well as tools using the model-driven paradigm. It also outlines some conclusions obtained from practical experiences.

This article is organized in seven sections. Section 2 presents related work. It studies the importance of requirements validation and introduces the current situation. Section 3 provides a global vision of NDT and its evolution in the last years and Section 4 states how the application of model-driven Engineering (MDE) can improve validation requirements on NDT and how this methodology

exploits MDE principles to make easier this task. Then, in Section 5, the paper includes some relevant experiences when applying these solutions in the business context. Finally, Sections 6 and 7 summarize some conclusions obtained and present future work.

2 Related Work: Requirements Validation in Web Engineering

The validation of the requirements for a Web application is an important but not a common topic of research. It was a difficult for us to locate the relevant literature on this topic. As a consequence, we focus our search on more general topics, e.g. requirement validation, web applications, and web engineering.

In 2004, Escalona & Koch [2] presents a survey that analyzes how Web requirements are covered by Web approaches. They showed how Web approaches use classical techniques for requirements treatment. In requirements validation, they numbered four techniques: review or walk-throughs, audits, traceability matrixes and prototypes. In this study, ten Web approaches are analyzed and the paper presents how they use these techniques for requirements validation. If compared with requirements specification and capture, requirements validation is the less considered phase. Although this paper was written in 2004, the situation has not recently achieved a significant change. Some Web approaches, like WebML [17], have incorporated requirements phase in their life cycle. However, very few improvements for requirements validation were proposed. Robles et al. [18] propose mockups as techniques to represent requirements, since they help to report results to users. Dargham and Semaan [19] propose a requirement validation technique based on validation through visualization and animation to verify completion, correctness and consistency of Web navigations. This approach is mainly aimed at verifying navigational requirements. In [20], Garrigós et al. proposes the adaptation of the i* modeling framework [21], an approach for analyzing stakeholders' goals and how the intended system would meet them.

Another trend refers to requirements testing. Sommerville and other authors propose requirements testing as a validation technique. Recently, some Web approaches have included this tendency in their life cycles. Thus, WebML [22] includes BPMN (Business Process Management Notation) [230] as Computation Independent Model and proposes the systematic generation of test cases by means of model-driven paradigm. Robles et al. [24] are carrying out something similar. They are working to include testing requirements in their approaches. NDT, as it is presented in the following sections, also includes this possibility in the life cycle. From the previous paragraphs it can be summarized that although requirements validation is a very critical task in requirements engineering, it is poorly covered by Web approaches [16]. There is poor support of concrete techniques and tools,

even though techniques used in Web requirements validation are the same as those applied in classical approaches, e.g reviews, prototypes, traceability matrix, etc. Reviews and prototypes validation are considered "psychological" techniques [16] because they depend on the stakeholders' background and their objective point of view [9]. Development teams have to decide either if a guided or a free revision should be better. This aspect is more complex in the Web environment because end-users are frequently unknown, and assessing requirements with them is not possible. Further, the generation of prototypes, traceability matrixes and requirements testing are usually quite expensive for projects, and this cost is only assumed if duly justified. Besides, the maintenance of traceability matrixes and early tests could be also quite expensive if not supported by a tool case. On Web systems, this problem could becomes even worse since maintenance on Web systems is usually more complex than in classical projects: they must run 24 hours a day, 7 days a week, 365 days a year.

In fact, there are several commercial tools that contribute to the application of these techniques. For instance, IBM Rational Dynamic Object Oriented Requirements System (Doors) [25], HP Requirements Management [26], Blueprint Requirements Center [27], IRQ-A [28] and Polarion Requirements [29] are some examples of generic and commercial tools for requirements management. Each of these offers suitable solutions for the general management of requirements as well as for requirements validation: they execute reviews or support traceability matrices, among others. In addition, they offer a way to continue with the lifecycle. As an example, HP Requirements Management is integrated with the HP Application Lifecycle Management tools. IBM Rational Doors enables the generation of UML 2.0 models that can be exported to UML tool cases, and IRQ-a connection with Enterprise Architect [30].

However, these tools mark a higher distance between requirements and the remaining lifecycle. For instance, if we used IRQ-A or Doors for requirements management and, later, we exported them to another tool, a change in requirements would imply the development team would have to manage this change manually[1].

In addition, they do not offer a concrete solution for requirements validation in the Web environment. All of them offer some mechanisms to define and create different categories of requirements, but the result can be too general without a specific solution.

In conclusion, techniques for Web requirements validation seem to be clear, although their application must improve. Katasonov and Sakkinen [15] highlight that the main problem of requirements validation is communicating requirements because customers and end-users most likely do not have any technical expertise.

[1] In Section 5.1 of this paper there is a reference about this aspect in the Mosaico Project where, initially, Doors was used.

The next section explains how to apply these recommendations and the classical techniques in the Web environment followed by NDT. Furthermore, NDT shows the latest trend to use the model-driven paradigm for this aim and suggests some suitable tools to support the application of these techniques at a lower cost.

3 An Overview of NDT

NDT (Navigational Development Techniques) is a model-driven Web methodology that was initially defined to deal with requirements in Web development. NDT has evolved in the last years and offers a complete support for the whole life cycle. NDT is completely supported by a set of free tools, grouped in the NDT-Suite [31]. It selects a set of metamodels for each development phase. All concepts in every phase of NDT are metamodeled and formally related to other concepts by means of associations and/or OCL constraints [32].

In order to offer suitable support for NDT, we studied a set of different possibilities before starting to develop the NDT-Suite.

The first proposal was to use UML as the basis for NDT models. We defined a set of UML profiles to offer a suitable syntaxis for the use of the NDT metamodel. We selected UML profiles because, after some empirical studies, we realized that it was the easiest device for people in companies; on one hand, UML is commonly used in software companies, and thus development teams already know its notation. On the other hand, they usually work with UML-based tools.

After this consideration, we studied the possibility of developing our own tool for NDT or to use an existing UML based tool where we could define our UML profiles. In this sense, after studying some possibilities, we chose to use Enteprise Architect as the UML tool for NDT. The decision required a comparative study carried out together by our research group and the Andalusian Government. It determined the tool which offered the best position in price/quality ratio[2].

Another suitable possibility was to use Eclipse and EMF technologies [35]. However, as NDT is mainly oriented to requirements and end users' work, it does not offer as good interfaces as UML models, for instance, with use cases [36].

To conclude the sort presentation of NDT, we summarize the following points.

1 NDT is a MDE methodology that covers the whole life cycle. However, it is mainly focused on the requirements phase. In this phase, NDT offers a set of techniques to capture, define and validate requirements of different kinds.

[2] This study was written in Spanish. It was not published, but it can be consulted in www.iwt2.org.

These requirements are formally defined by a metamodel and they can be traced to the remaining artifacts of the life cycle by managing them in a suitable manner.

2 Despite its application in classical environments, NDT is developed in relation to the Web. In this sense, it supports special characteristics like navigation, complex interfaces or RIA [33]. In the requirements validation, NDT is oriented to cover classical techniques like traceability, prototypes, etc. but enriched to support these special web characteristics.

3 The degree of automation of NDT is one of its more relevant qualities. NDT is a theoretical approach, based on metamodels, transformations, etc. However, it is also an approach often used in companies[3].

4 Techniques and Tools for Requirements Validation in NDT

NDT supports different requirements validation techniques and offers several tools to automate their applications. This section presents the solution offered by NDT for requirements validation.

NDT supports the following requirements validation techniques:

- ***Requirements reviews:*** This technique checks requirements in detail. In reviews, clients and analysts need to check the consistency of requirements and whether they were correctly defined. Reviews are sometimes quite difficult to implement because they have a relevant psychological component [15]. Flaws in understanding requirements imply a false acceptance of requirements, which may lead to important errors in the system [2].
- ***Glossaries***: In several studies, such as [2], glossaries are not described as a validation technique. Nevertheless, NDT proposes using them in order to check the terminology consistency in requirements definition as an auxiliary technique. A glossary is a dictionary of terms for the system [5]. It is quite useful in systems with a heterogeneous development team to achieve lexical consistency during the process.
- ***Prototypes***: A prototype is partial implementation of a system that helps to carry out the performance and assessment of the future system with users. Prototypes are useful tools to work with users because they enhance interaction and ease communication. They can be classified in different ways, either upwards or downwards or according to high fidelity or low fidelity. All offer different possibilities to validate systems [36] [37].

[3] In www.iwt2.org in the Project section a detailed list of projects where NDT is used can be checked.

- ***Matrix of traceability***: This technique consists of the use of matrixes to establish correspondence among different artifacts in the system's development. In requirements, this technique allows, for instance, knowing how objectives are satisfied with a set of requirements [7].
- ***Requirements testing***: The requirements testing generates test cases to enable requirements testing [5]. Tests are not executed till the system is implemented. However, the early generation of tests validates the requirements definition by analyzing, in collaboration with end-users, functional paths that in the future should be tested. This technique is frequently named *early testing* [12].

NDT bears out all these techniques and offers tools to support their applications. As an introduction, Table 1 represents a matrix with each technique and the tool that supports it.

In the next section, each tool is presented in order to explain how they support every technique by means of the model-driven paradigm. Screens and examples offered in test introduction were obtained from an example, named Hotel Ambassador, which can be downloaded from [31]. It is an example fully developed with NDT-Profile to test our tools.

Table 1
Tool supplied for each technique

	Reviews	Glossaries	Prototypes	Traceability Matrix	Requirements Testing
NDT-Driver					X
NDT-Quality				X	
NDT-Glossary		X			
NDT-Report	X	X		X	X
NDT-Prototypes	X		X		
NDT-Checked	X				

4.1 NDT-Driver

NDT-Driver is a tool that supports each transformation defined in NDT. It implements transformations from requirements to analysis, analysis to design and requirements to test, as illustrated in Figure 1. The last transformation (Design to Code) is not executed by NDT-Driver in NDT, as it is supported by a plug-in of Enterprise Architect. They define a generation process based on QVT Transformations [34]. Figure 1 presents the idea more specifically.

NDT defines two transformations T1 and T2 in QVT Operational, based on the NDT functional requirements metamodel. T1 generates possible test scenarios from these functional requirements. The method used is the Path Analysis method

[39]. T2 is another transformation that, through the Category-Partition method [40], generates operational values for these test scenarios. A new transformation T3 is defined from both metamodels. T3 is thought to generate the test case metamodel. This process is fully presented in [41].

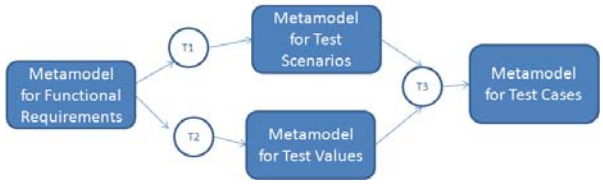


Figure 1
NDT- Driver process to generate requirements testing

In order to use this approach in NDT-Suite, three UML profiles for these new metamodels are defined and included in Enterprise Architect and within the NDT-Profile: test values, test scenarios and test cases profiles. Thus, a concrete syntax to define their test cases is offered. Transformations are translated in Java and they appear in the set of transformations of NDT-Driver.

Through NDT-Report, as the next sections state, the development team can generate functional test documents. They can execute two different scenarios with this approach, as shown in Table 2. Both scenarios start with the definition of functional requirements with NDT-Profile. NDT-Profile proposes an extension of use cases, activity diagrams and some specific patterns to represent them. After checking their quality by means of NDT-Quality, with the possibilities that are presented in next sections, scenarios offer two paths.

Table 2
Scenarios provided by using the NDT-Driver for functional test generation

Scenario 1	Scenario 2
1 The development team defines functional requirements in NDT-Profile.	1 The development team defines functional requirements in NDT-Profile.
2 They check requirements quality with NDT-Quality.	2 They check requirements quality with NDT-Quality.
3 They generate the requirements catalogue, using NDT-Report, and validate it with users.	3 They use NDT-Driver and execute Req2Test transformations to generate the functional test catalogue in NDT-Profile.
4 They use NDT-Driver and execute Req2Test transformations to generate the functional test catalogue in NDT-Profile.	4 They use NDT-Report to generate a printable version of the functional test catalogue.
5 They use NDT-Report to generate a printable version of the functional test catalogue.	5 They validate requirements with users through this functional test catalogue.
6 When the test phase is executed, they use this functional test catalogue generated in the requirements phase.	6 When the test phase is executed, they use this functional test catalogue generated in the requirements phase.

In Scenario 1, the development team validates functional requirements by means of other techniques (reviews, prototypes, etc.) supported by NDT with some alternative tools, as is presented in following sections. Once errors and mistakes

have been amended, the development team can execute Requirements to Test transformations in order to generate the functional test cases that will be used in the future when the system may be implemented and the test phase starts.

However, Scenario 2 presents an option oriented to validate requirements through testing. In this situation, once NDT-Quality has checked the requirements quality, functional test cases are generated before the official requirements validation takes place, and they are used to validate requirements with users.

This idea provides NDT with the possibility of validating requirements from tests based on the model-driven paradigm. As Section 5 presents, this event adds some important advantages in the enterprise environment as the process is automatic and offers a powerful mechanism to facilitate the communication with users. For them, analysis of a functional circuit modeled as a test case is frequently easy and clear.

Obviously, this possibility is only a part of the process because it only considers functional requirements. Now, we are working in delivering this test generation from other kinds of requirements, such as navigation requirements.

4.2 NDT-Quality

In NDT, the application of a set of transformations executed by NDT-Driver supports each step in the life cycle. However, NDT-Driver not only carries out these transformations, but goes further; it saves the relation between the source artifact and the target artifact when a transformation is executed. For instance, Figure 1 shows how NDT-Driver saves this relation when a test case X is generated from a functional requirements Y with the transformations Req2Test. The storage of this relation offers several advantages, mainly oriented to the system traceability and maintenance.

In the future, if the functional requirements Y changes, the test case X must be automatically reviewed, as it will probably change. The tool to check quality and traceability in the system is NDT-Quality. It controls three aspects in a system:

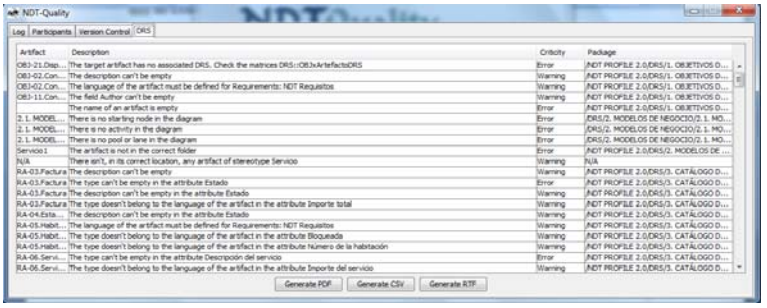
1 It implements a set of rules, defined by NDT as OCL constraints or invariants in their metamodels. Once a development team finishes each phase in the life cycle of NDT, they are expected to execute NDT-Quality to check that each rule or constraint defined by NDT is followed, e.g. it ensures that each requirement is defined by a unique identification code and a short description.

2 It implements a set of rules, defined by UML or general rules. It guarantees that an activity diagram is well-defined, e.g. without independent activities.

3 It implements a set of rules to check the traceability of the system. Following the previous example, it ensures that a change in requirements Y implies a review in test case Y.

NDT-Quality reports the detected errors and recommendations when executed. In Figure 2, Section a. presents the interface of NDT-Quality, which shows aspects that can be checked with NDT-Quality in different phases and traceability aspects whereas Section b. shows an example of a report.

a. Main interface of NDT-Quality



b. Example of report in NDT-Quality

Figure 2
Interface of NDT-Quality

In addition, NDT-Quality manages the traceability matrix of the system generated for relations created from transformations and NDT rules. Figure 3 offers an example of traceability matrix. It is automatically generated and manages how objectives are partially covered by functional requirements. NDT-Quality controls a high number of traceability matrixes: objectives-requirements, storage requirements-analysis classes, functional requirements or test cases, among others. All of them are automatically generated and can be required by the development team, if needed.

The automatic generation of a traceability matrix is a powerful tool in NDT derived from the application of the model-driven paradigm. This is one of the most frequent techniques to validate requirements and it is compulsory in relevant good practices and quality standards, like CMMi (Capability Maturity Model Integration) [42]. Section 4 explains its relevance in empirical experiences.

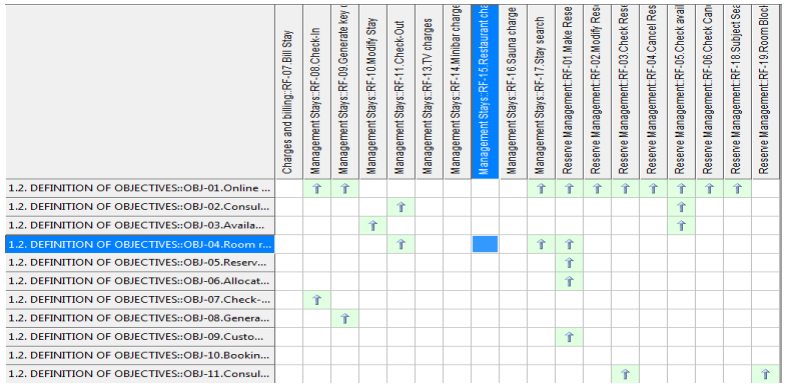


Figure 3
Example of traceability matrixes

4.3 NDT-Glossary

The glossary of terms in software projects allows the development team to store and exchange the knowledge acquired in the system domain. Basically, it is a dictionary that defines the most important concepts used during the development process of a software project. It is oriented to unify the vocabulary and control inconsistencies and ambiguities of concepts within the system domain in the life cycle. Every term is represented in the glossary as a couple, such as name and description, and through a set of relations with other terms; synonyms, related, etc. To keep the integrity of the glossary, each name must be unique. Glossary elaboration is not a requirements validation technique itself, but it results in quite an efficient way to find lexical inconsistencies during the requirements phase. Each glossary must verify two principles [8]: the Principle of Circularity and the Principle of Minimum Vocabulary. On one hand, the Principle of Circularity establishes that a glossary should be as self-content as possible. In this way, it ensures that all terms are related. At the same time every term and the relation with the remaining terms are included in the glossary. On the other hand, the Principle of Minimum Vocabulary states that requirements should be mainly expressed by concepts in the glossary, and thus it will be as understandable as possible.

In conclusion, engineers need to gather and define the most relevant and critical concepts for the system. Furthermore, a common language reduces the risk of misunderstandings and facilitates communication between users and analysts. NDT offers a tool in its suite, named NDT-Glossary, which uses model-driven paradigm to generate a glossary from the requirements model. Figure 4 represents this idea, where NDT defines a metamodel to represent a glossary, and later, a set of QVT transformations from the requirements metamodel to the glossary

metamodel is defined. Thus, these transformations start from the storage information requirements. These types of requirements in NDT define which information the system must manage and they are described with the user's vocabulary. This information is transformed to a glossary model. Both metamodels have an associated profile implemented in NDT-Profile whereas NDT-Glossary implements transformations in Java. Thus, a development team can get a first instance of the glossary from the requirements metamodel, or more precisely, from the storage information requirements metamodel.

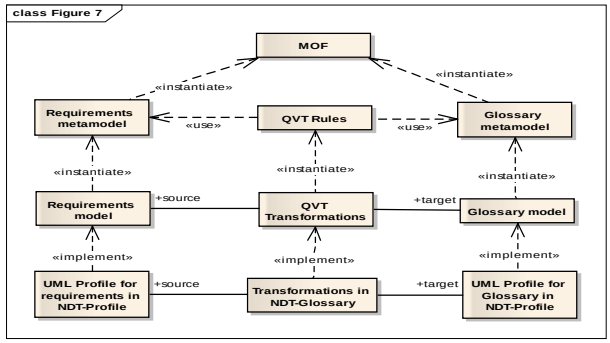


Figure 4
Glossary model transformation pattern

4.4 NDT-Report

When NDT started working in the enterprise environment, we noticed that it had an important bug. It offered a set of powerful tools like NDT-Driver or NDT-Quality quite oriented to make easier the use of the methodology for the development team. However, results given by NDT-Profile were not the best for the end-users' review. NDT-Profiles stores NDT models (mainly represented as UML models and patterns) whose interface is quite useful for development teams, but too complicated for end-users.

For this reason, a new tool named NDT-Report was developed. This tool defines a set of patterns to present NDT results to users. It can generate output in word files, pdf files or html files, as well as implement a set of transformations from NDT metamodels and generate the output to these patterns.

NDT-Report is essential for end-user participation in the NDT development cycle. It can prepare suitable outputs for each phase (requirements, analysis, testing, and so on) but it can also offer an external view of the traceability matrix of NDT-Quality and the glossaries generated by NDT-Glossary.

This tool is not a model-driven case tool. However, it offers suitable outputs to apply classical requirements validation techniques.

NDT-Report is the tool which carries out the revision in liaison with users. In fact, if a suitable and comprehensive output of the requirements is not offered, users cannot assess them [13] [14]. NDT-Report is the tool to get a printable version of requirements, glossaries, traceability matrices and functional tests.

4.5 NDT-Prototypes

Using prototypes to validate requirements is one of the most used techniques in the enterprise environment [43]. Prototypes normally assure that the end-user can easily understand the future system. It is also considered a very useful technique since it involves users in the requirements phase.

However, prototypes can have some disadvantages in use. They generally increase the development time. So they can delay the project due to the extra time needed for development. Normally, this time is paid back by the detection of errors and inconsistences in the first phases of the life cycle, which improves the final system quality. In addition, the granularity or the degree of development of the prototype may be a problem. If it is very detailed, users could consider that the prototype is the final version of the system, while if it is very general, it could not be relevant for evaluation [36].

In this sense, a new tool for the NDT-Suite was developed in order to get prototypes advantages and reduce the elaboration cost. This tool, named NDT-Prototypes, generates a set of prototypes from the requirements model of a system and uses the same ideas of NDT-Glossary. It implements a set of QVT transformations from the requirements model to the prototypes model in JAVA. However, in this case, the source is the interaction requirement. NDT supports interaction requirements presented in its metamodel as Visualization Prototypes. A Visualization Prototype instance represents how users process the information and how they can execute functional requirements and navigate through the system. This information described in the requirements model is transformed to a prototype model executing this set of transformations, which are implemented in Java as in the previous examples.

The interface of NDT-Prototype is quite simple and generates a set of html and css Websites. Figure 5 presents a screen generated with NDT-Prototype for the Ambassador example.

Each specific field is translated into a text field in the prototype. Depending on the values of their attributes (name and type) one specific user interface element is used. As an example, text boxes are used if type is “String”. Buttons offer the possibility of executing functional requirements. Thus, the model includes a relation between the visualization prototype “Create Reserve” and the functional requirements “carry out a reserve”. This functional aspect can be executed from the screen derived from “Create Reserve”.

Figure 5
Screen generated by NDT-Prototype from interaction requirements for the Hotel Ambassador

This model is not complete, as relations with activities, navigation and others are not presented in the figure. Nevertheless, it illustrates how NDT-Prototypes can help to understand the model. The use of prototypes is an essential technique for users' validation [44]. The full example is available in [31] in the Hotel Ambassador example.

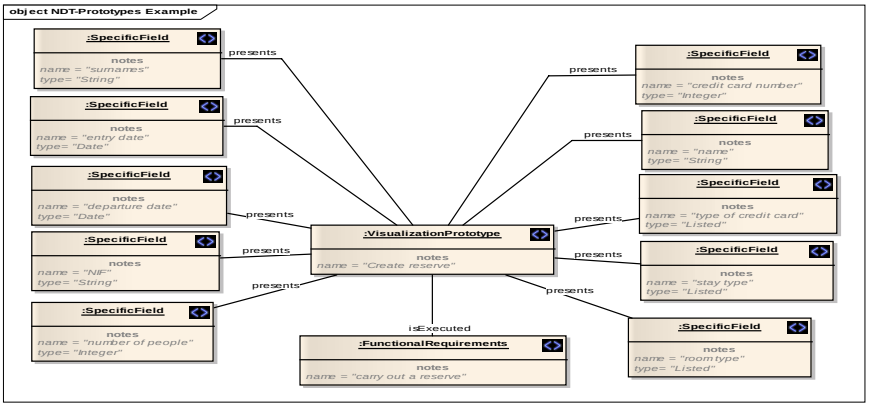


Figure 6
Original model for example in Figure 8

4.6 NDT-Checked

NDT-Checked and NDT-Report are the only tools in NDT-Suite that are not based on the model-driven paradigm. The NDT-Checked tool includes different sheets for each NDT product. These sheets give a set of check lists that should be manually reviewed with users in requirements reviews.

As it is presented in the Related Work section, requirements review is one of the most used techniques. It is essential for validation. However, it is sometimes difficult to consider which aspect must be reviewed with end-users [8]. There are

two main options: *free revision*, in which case the user reviews the requirements catalogue alone and freely; and *guided revision*, where the user reviews the requirements catalogue with the help of a development team member.

In NDT, the first one can be executed generating the requirements catalogue with NDT-Report and presenting the results to the user for revision. In the second option, NDT-Checked was developed. NDT-Checked, which is mainly based on enterprise experience, offers this set of checklists to assess and review each product generated in the life cycle of NDT.

5 Learned Lessons from Industry Experiences

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

Some specific projects have been selected in order to put forward some learned lessons from the empirical experience achieved when using the presented tools for requirements validation. All of them were developed with NDT, and its tools were used during their life cycles.

5.1 Projects for Cultural Heritage Management

The Andalusian Regional Cultural Ministry [45] has been applying NDT since 2004. Over the past several years, more than 90 projects of Web systems with different providers, users or development teams have worked with our approach. This experience is quite relevant, mainly in the use of NDT-Report and NDT-Quality. The Cultural Ministry does not accept any results or documents of a project if they are not checked with NDT-Quality.

The use of NDT-Report is also quite relevant. This set of users extends CSS and NDT-Report output design with their own patterns. In addition, they notice that when the same user participates in two or more projects, he or she can easily use this notation. In fact, Mosaico users [46] directly need NDT-Profile to validate its requirements. Mosaico is a big project that started in 2004, but it is continually being improved. These tools are essential to validate and trace requirements. In fact, at the beginning, for the requirements phase of Mosaico, the company that developed it suggested using a profile of Doors. However, in the first iteration, the development team noticed that the traceability of requirements with the rest of the life cycle was quite difficult, due to a disconnection between requirements tools and analysis (in this case, Enterprise Architect).This enhanced our interest in improving the traceability from requirements to analysis in NDT-Suite and, more specifically in NDT-Quality (see Figure 2a where NDT-Quality supports requirements-analysis traceability).

One of our future lines of work in this environment consists in applying NDT-Prototypes. This is one of our youngest tools, only applied in some projects. According to previous experiences in this environment, we conclude that requirements validation can be improved by adapting this tool to this environment.

5.2 AQUA-WS Project

The AQUA-WS (AQUA-WebServices) project [07] is a very important project carried out in Emasesa [08] over three years and finished in 2011. AQUA-WS is very relevant for the application of NDT-Driver in the test phase.

The AQUA-WS project included the development and implementation of an integrated business system for customer management, interventions in water distribution and clean-up, and management work or projects.

This project was launched when Emasesa needed to integrate the existing systems into a single one along and to upgrade the technological platform of the system. The existing systems were the customer management system (AQUA-SiC), network management system (AQUA-ReD) and the work and projects management system (AQUA-SigO). Thus, as the project was a technological migration of old systems, users only took part in the testing phase.

The project followed an iterative life cycle mainly based in RUP [09]. In each iteration, the development team, composed of more than 20 analysts from two companies, defined requirements, after studying the previous systems, and introduced them into NDT-Profile. Then, they were checked with NDT-Quality and NDT-Checker and, later, functional test cases were generated. NDT-Report had these functional test cases presented as functional paths reviewed with users.

The systematic way of generating test cases from functional offers a suitable and quite agile support for validating these functional requirements with users. However, a relevant conclusion is obtained from this experience. The quality and the suitability of derived test cases depend on the quality of the requirements. In some functional iterations, requirements have to be reviewed and written again, even before test generation, since they are poorly described.

5.3 Projects for e-Health Systems

NDT was also widely applied in the e-health environment. In 2006, Alcer Foundation [50] used it within the system to manage the degree of handicap for disabled people. In this project, NDT-Suite was not fully developed and we used a previous tool, named NDT-Tool [51]. However, this project is mentioned as it was the seed for detecting the need for NDT-Glossary. The medical systems environment works with very specific terminology: the project caused a high number of inconsistences only solved by elaborating a glossary manually.

Some years later, NDT-Suite was used in another e-health system, named Diraya [02], 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.

Conclusions and Future Work

This paper analyses the importance of requirements validation in Web Engineering. It presents an overview of today's situation in this research line and concludes with the need for offering systematic mechanisms to improve and even automatize this task. The article defends the idea of applying the model-driven paradigm to these aims. NDT-Suite is presented to validate this idea and more specifically, the tools that support requirements validation techniques.

We included some references to real projects which used these tools to support the requirements validation task.

As lessons learned from our experience with the model-driven paradigm in Web treatment, we could state that the use of this paradigm in this environment can improve the project results. However, development teams do not find the model-driven paradigm too intuitive in practical environments. The concepts of metamodels and transformations, among others, are not common notations for daily practice in industry, as they seem too abstract.

Nevertheless, we conclude that the use of UML profiles and UML-based tools offer an interface to deal with instances of metamodels suitable for analysts, designers and even for expert users.

In the same way, transformations in QVT do not appear easy to understand. However, our users do not work with QVT, but with a very easy interface, like the NDT-Quality interface in Figure 2a, to benefit the power of transformations.

As a summary, our experience has confirmed that requirements validation involves one of the most difficult and critical tasks. The project success heavily depends on the results of this phase; therefore, it is essential to manage it correctly. The lack of systematic or automatic techniques that help to support requirements validation represents an important limitation for software development. They frequently depend on psychological aspects and they are rarely based on tools. Our experience demonstrates that the model-driven paradigm can help to systematize and even automate the most classical requirements validation techniques. It offers cost and time reduction in this phase and helps to increase the quality of results.

The main advantage of our approach is that it uses a model-driven mechanism when offering the requirements output, which reduces the cost of their generation. The division into different types of requirements described in NDT metamodel

implies that each group of stakeholders works in an established group or sub-group: it reinforces the need of giving them an explicit task to accomplish requirements validation, if possible, based on systematic and guided techniques. In our approach, the development team can support several techniques to manage this task, e.g. reviewing a part of the glossary developed by NDT-Glossary; reviewing a set of requirements, such as a functional module in HTML obtained with NDT-Report; or reviewing the coverage and the requirements traceability with traceability matrixes generated by NDT-Quality. Once again, the cost reduction of a model-driven paradigm supports automatically these options and enables improving the reviews.

Finally, it is again easy to divide the review process into problems and small steps to be reviewed using this environment because outputs can be obtained in separate ways without added costs.

As an added value, although it is not the aim of this article, we want to remark that NDT is not only a requirements environment, but offers a connection with the remaining life cycle, even with other activities such as quality assurance or project management. In this sense, requirements information is available for connection with automated methods for test case generation [53]. The model-driven principles presented in this article for requirements validation can also be adapted to the rest of the methodology. Attending to our practical experiences, we can highlight that this paradigm might provide suitable results to companies using some abstract concepts like metamodels or transformations. In fact, we consider that this paradigm, apart from being used by the research community in the last years, is now starting to offer results and may become a very useful mechanism for building software, as well as for its maintenance or management.

Research work presented in this paper allows future development in different lines of work. We would like to add more tools to the NDT-Suite to develop Web systems. Currently, we are working on a new tool, named NDT-Counter, that it is oriented to estimate the cost of a Web system at the beginning of the life cycle. It is based on the model-driven paradigm and applies the Use Case Point technique to measure the development time of a system.

The environment introduced in this paper, which is used in different companies, is certificated under ISO 9001:2008 [54], UNE EN 166002 [05] and ISO 14001[06]. We are increasing the possibility to include processes to support some other standards like CMMi level 2 and ITIL v3 (Information Technology Infrastructure Library) [07]. NDTQ-Framework offers a set of processes to deal with these standards. Thus, if a company uses NDT and NDT-Suite while they want to pursue the certification under these standards, they could take NDTQ-Framework processes as reference. The tool does not only offer these processes, but also defines metrics, outputs and techniques useful for this goal. The implementation of metrics help to overcome limitations to their application in SME [58].

Acknowledgements

This research has been supported by the project Tempros project (TIN2010-20057-C03-02) of the Ministerio de Ciencia e Innovación, Spain and NDTQ-Framework project of the Junta de Andalucia, Spain (TIC-5789).

References

[1] Molina F., Toval A., Integrating Usability Requirements that Can Be Evaluated in Design Time into Model Driven Engineering of Web Information Systems. Advances in Engineering Software. Vol. 40, Issue 12, December 2009, pp. 1306-1317

[2] Escalona, M. J., Koch, N. Requirements Engineering for Web Applications: A Survey. Journal of Web Engineering, Vol. II, Nº2, pp. 193-212, 2004

[3] Aguilar, J. A., Garrigós, I., Mazón, J. N., Trujillo, J. Web Egineering Approaches for Requirements Analysis- A systematic Literature Review. Proceedings of WebIST 2010, pp. 187-190, 2010

[4] Pressman, R. S. Software Engineering. A practitioner's approach. Mc Graw Hill, 2004

[5] Sommerville, I. Software Engineering. Addisson Wesley, 9th Edition. 2010

[6] Escalona, M. J., Aragón, G. NDT: A Model-Driven Approach for Web requirements, IEEE Transactions on Software Engineering. Vol. 34, No. 3, pp. 370-390, 2008

[7] Bernárdez, B., Durán, A., Genero, M. Empirical Evaluation and Review of a Metrics-based Approach for Use Case Verification. Journal of Research and Practice in Information Technology. Vol. 36, No. 4, pp. 247-258, 2004

[8] Leite, J. C. S. P., Eliciting Requirements Using a Natural Language-based Approach: The Case of the Meeting Scheduler Problem. Monografias em Ciência da ComputaÇao. No. 13, 1993

[9] Leite, J. C. S. P., Requirements Validation through Viewpoint Resolution. IEEE Transaction on Software Engineering. Vol. 17, No. 12, pp. 1253-1269, 1991

[10] Silva, J. R., dos Santos, E. A., Applying Petri Nets to Requirements Validation. ABCM Symposium. Series in Mechatronics. Vol. 1, pp. 508-517, 2004

[11] Zhu, H., Jin, Lingzi, Diaper, D., Bai, G. Software Requirements Validation via Task Analysis. The Journal of System and Software. No. 61, pp. 145-169, 2002

[12] Escalona, M. J., Gutierrez, J. J., Mejías, M., Aragon, G., Ramos, I., Torres, J., Domínguez-Mayo, F. J. An Overview on Test Generation from Functional Requirements. Journal of Systems and Software. No. 84, pp. 1379-1393, 2011

[13] Gemino, A. Empirical Comparison of Animation and Narration in Requirements Validation. Requirements Engineering. No. 9, pp. 153-168, 2004

[14] Uchitel, S., Chatley, R., Kramer, J., Magee J. Fluent-based Animation: Exploiting the Relation between Goals and Scenarios for Requirements Validation. Requirements Engineering. 2004

[15] Katasonov, A., Shakkien, M. Requirements Quality Control. A Unifying Framework. Vol. 11, No. 1, pp. 42-57, 2006

[16] Sulehri, L. H. Comparative Selection of Requirements Validation Techniques Based on Industrial Survey. Department of Interaction and System Design. School of Engineering. Blekinge Institute of Technology. Master Thesis. December 2009, Ronneby, Sweden

[17] Brambilla, M., Butti, S., Fraternali, P. WebRatio BPM: A Tool for Designing and Deploying Business Processes on the Web. International Conference on Web Engineering. pp. 415-429 Web Austria 2010

[18] Robles, E., Garrigós, I., Manzón, J. N., Trujillo, J., Rossi, G. An i*-based Approach for Modeling and Testing Web Requirements. Journal on Web Engineering. Vol. 9, Nº4, pp. 302-326, 2010

[19] Dargham, J., Semaan, R. A Navigational Web Requirements Validation through Animation. Third International Conference on Internet and Web Applications and Services. pp. 211-216, Greece, 2008

[20] Garrigós, I., Mazón, J. N., Trujillo, J. A Requirements Analysis Approach for Using i* in Web Engineering. International Conference on Web Engineering. LNCS 5648, Spain, 2009

[21] Yu, E. Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering. 3rd International Symposium on Requirements Engineering. pp. 226-235, 1997

[22] Fraternali, P., Tisi, M. Multi-Level Tests for Model Driven Web Applications. International Conference on Web Engineering. pp. 158-172, Austria, 2010

[23] Business Process Management Initiative. Available in www.bpmn.org. Accessed January 2012

[24] Robles, E., Grigera, J., Rossi, G. Bridging Test and Model-Driven Approaches in Web Engineering. International Conference on Web Engineering. pp. 136-150, Spain, 2009

[25] IBM Rational Doors. Available in www-01.ibm.com/software/awdtools/doors. Accessed in September 2012

[26] HP Requirements Management. Available in www8.hp.com/us/en/ software/software-solution.html?compURI=tcm:245-937050. Accessed in September 2012

[27] Blueprint Requirements Center. Available in www.blueprintsys.com/ products, Accessed in September 2012

[28] IRQ-A. Version 3.0. Available in www.visuresolutions.com. Accessed in September 2012

[29] Polarion Requirements. Available in www.polarion.com. Accessed in September 2012

[30] Enterprise Architect 9.0. Available in www.sparxsystems.com.au. Accessed in September 2012

[31] NDT-Suite. Available in www.iwt2.org. Accessed in September 2012

[32] Object Constraint Language. Available in www.omg.org/spec/OCL/2.2/. Release 2.2. 2010. Accessed in September 2012

[33] Robles, E., Escalona, M. J., Rossi, G. A Requirements Metamodel for Rich Internet Application. ICSoft 2010 Selected paper. Communications in Computer and Information Science. Springer Verlag. To be published. 2012

[34] Query/View/Transformation. Available in www.omg.org/spec/QVT/1.1/. Release 1.1. 2011. Accessed in September 2012

[35] EMF Technologies. Available in www.eclipse.org/modeling/emft/. Accessed in September 2012

[36] Valderas, P., Pelechano, V., Pastor, O. A Transformational Approach to Produce Web Application Prototypes from a Web Requirements Model. Vol. 3, Nº1, International Journal of Web Engineering and Technology. pp. 1476-1289, 2006

[37] Chavarriaga E., Macías J. A., A Model-driven Approach to Building Modern Semantic Web-based User Interfaces. Advances in Engineering Software. Vol. 40, N. 12, pp. 1329-1334, 2009

[38] Garcia-Garcia, J., Cutilla, C. R., Escalona, M. J., Alba, M., Torres, J. NDT-Driver, a Java Tool to Support QVT Transformations for NDT. The Twentieth International Conference on Information Systems Development (ISD) To be published. 2012

[39] Naresh, A. Testing From Use Cases Using Path Analysis Technique. International Conference on Software Testing Analysis & Review. 2002

[40] Ostrand, TJ, Balcer, MJ. Category-Partition Method. Communications of the ACM. 676-686. 1988

[41] Gutiérrez, J. J., Escalona, M. J., Mejías, M., Torres, J., Torres-Zenteno, A. H. A Case Study for Generating Test Cases from Use Cases. Proceedings of RCIS 2008, Morocco, pp. 223-228, 2008

[42] Capability Maturity Model Integration (CMMi). Available in www.sei.cmu.edu/cmmi/. Accessed in September 2012

[43] Kasser, J. A Prototype Tool for Improving the Wording of Requirements. 12th Annual International Symposium of the INCOSE, pp. 1-12, USA, 2002

[44] Mátrai, R., Kosztyán, Z. T. A New Method for the Caracterization of the Perspicuity of User Interfaces. Acta Polytechnica Hungarica. Journal of Applied Sciences, Vol. 9, N. 1, pp. 139-156, 2012

[44] Consejería de Cultura. www.juntadeandalucia.es/ccul. Accessed in September 2012

[45] Escalona, M. J., Aragón, G., Molina, A., Martinez-Force, E. A MDWE Methodological Environment for Culture Heritage. Technologies for Tourism Destination Management and Marketing: Tools and Trends. IGI Global. To be published. 2011

[46] Escalona, M. J., Gutiérrez, J. J., Rodríguez-Catalán, L., Guevara, A. Model-Driven in reverse. The Practical Experience of the AQUA Project. Euro American Conference on Telematics and Information Systems. pp. 90-95, Czech Republic, 2009

[47] Emasesa. www.aguasdesevilla.com. Accessed in September 2012

[48] RUP. Rational Unified Process. Available in www-01.ibm.com/software /awdtools/rup/. Accessed in September 2012

[49] Alcer. Federación Nacional de Asociaciones para la lucha contra las enfermedades renales. www.alcer.org. Accessed in September 2012

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

[51] Escalona, M. J., Parra, C. L., Martín, F. M., Nieto, J., Llergó, A., Pérez P. 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, 2008

[52] Escalona, M. J., Aragón, G., Molina, A., Martinez-Force, E. A Model-Driven Tool Framework for the Improvement in the Use of NDT. 8th International Conference on Software Quality Management. The British Computer Society. pp. 147-157, UK, 2010

[53] Fernandez-Sanz, L. and Misra, S., Practical Application of UML Activity Diagrams for the Generation of Test Cases, Proceedings of the Romanian Academy, Series A, Vol. 13, N. 3/2012, pp. 251-260

[54] ISO9001-2008. www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_ detail.htm?csnumber=46486. Accessed in September 2012

[55] UNE 166002. www.aenor.org. Accessed in September 2012

[56] 14001-2004. www.iso.org/iso/catalogue_detail?csnumber=31807. Accessed in September 2012

[57] Information Technology Infrastructure Library. ITIL Open Guide. Available in www.itil-officialsite.com. Accessed in September 2012

[58] Pusatli, O. T. and Misra, S., Software Measurement Activities in Small and Medium Enterprises: an Empirical Assessment, Acta Polytechnica Hungarica, Journal of Applied Sciences, Vol. 9, N. 1, pp. 139-156, 2012