Detecting Web Requirements conflicts and inconsistencies under a Model-Based Perspective

M.J. Escalona[1], M. Urbieta[2],
G. Rossi[2,3], J. A. Garcia-Garcia[1], E. Robles Luna[2]

[1] IWT2 Group. University of Seville, Spain
mjescalona@us.es, julian.garcia@iwt2.org
[2] LIFIA, Facultad de Informática, UNLP, La Plata, Argentina
{murbieta, gustavo, esteban.robles}@lifia.info.unlp.edu.ar
[3] Conicet

Abstract. Web requirements engineering is an essential phase in the software project life cycle for the project results. This phase covers different activities and tasks that in many situations, depending on the analyst's experience or intuition, help getting accurate specifications. One of these tasks is the conciliation of requirements in projects with different groups of users. This paper presents an approach for the systematic conciliation of requirements in big projects dealing with a model-based approach. The paper presents a possible implementation of the approach in the context of the NDT (Navigational Development Techniques) Methodology and shows the empirical evaluation in a real project by analyzing the improvements obtained with our approach.

Keywords. Web requirements, Consistency, Contradiction

1 Introduction

Eliciting Web application requirements implies understanding the needs of different stakeholders, those that are related to the same underlying enterprise business. Most of the times, requirements are agreed by stakeholders in such a way that the semantics and meanings of each term used are well understood. However if there are different points of view of the same business concept [17], ambiguities and/or inconsistencies may arise, becoming them detrimental to the Software Requirement Specification (SRS). Traditionally, conciliation tasks are performed through meeting-based tools [5], in order to eliminate requirements ambiguity and contradictions. Whenever requirement inconsistencies are not detected on time (being this one of the most severe reason of project cost overrun [18][34]), they may imply defects in the Web software. In this context, the effort to correct the faults is several orders of magnitude higher than correcting requirements at the early stages [22][18].

Besides, inconsistencies may also arise from new requirements, which introduce new functionality or enhancements to the application or, even, from existing requirements that change during the development process. For example, an online e-commerce site may plan a promotion for Christmas, where some products have free shipping for a period of time, whereas other products keep the standard shipping cost.

The user realizes the changes introduced by this new requirement through promotional banners in different pages. It is noteworthy that the shipping cost exception overrides and contradict the existing "shipping" requirement by introducing some ambiguities: what products have the free shipping promotion? In which way users are notified about the promotion? How long will the promotion be available?

During the last years, we have been studying different strategies to capture Web software requirements. Specifically, we have developed WebSpec [33], a domain specific language for capturing interaction and navigation requirements in Web applications. Similarity to other approaches like Molic [6] and WebRE [10], it provides primitives for describing inputs, outputs, navigations or interface transitions, among others, by helping to describe the main application concerns in a more accurate way.

As an initial work, we defined a model-based validation and inconsistency detection technique for Web application requirements, which was incorporated in WebSpec [30]. In this paper we adapt the approach to the practical environment of NDT (Navigational Development Techniques) [8]. NDT is a Web methodology mainly focused on requirements. Currently, this methodology has been and is being used in many projects

The paper has these aims: firstly, the incorporation of our requirements systematic validation in NDT; second the description of an empiric experiment to measure, in a real project developed with this methodology, how the inclusion of our approach in NDT can improve project results. This work opens new lines to extend our research to the enterprise environment.

The rest of this paper is structured as follows. Section 2 presents the problem which has been our catalyst to carry out this research (for this, we rely on a real project) and our goals. Section 3 presents some related work in requirements validation. Section 4 offers a global vision of NDT and a characterization of Web requirement conflicts. Section 5 shows our approach for detecting inconsistencies and dealing with them based on NDT by means of an illustrative example. Section 6 presents results of our experiment in a real project that was developed with NDT. And, finally, Section 7 concludes by discussing the lessons learned, our main conclusions and some further work on this subject.

2 Motivating Scenario: Mosaico Project

We summarize here which are the problems that our proposal aims to solve. To do this, we rely on one of the large projects we have been involved: Mosaico [9]. This project is a large project developed by the Regional Cultural Ministry of Andalusia using NDT. This Web application is oriented towards managing and spreading out each monument in the south area of Spain. It covers archaeological, architectural and ethnological historic heritage. First, we will briefly introduce the system and then we describe the original experience while gathering requirements. Finally, we present our research objectives.

2.1 Mosaico Project

The Regional Cultural Ministry of Andalusia started to develop Mosaico in 2004. The idea of this Web application was born from the need to manage all the information on historic heritage in Andalusia. Before Mosaico, there were several systems in charge of managing this information, what caused a lot of problems since the information was distributed, disconnected and different users worked in different platforms. Consequently, the growing need of managing and maintaining historic heritage promoted a project like this. Mosaico was developed by two important companies and it covered 5670 requirements, out of which 3253 were functional requirements.

Due to space restrictions, we will present the improvement our approach meant in the requirement gathering step of a Mosaico's module called *Subject to registration*.

2.2 Subject to registration: a Mosaico's module

In this subsystem, Mosaico stores the functionality to manage the basic information of each kind of monument or historical site that it supports. This module defines the basic structure of each piece of information managed by Mosaico.

The requirements phase of this subsystem was executed during three months and more than 60 end-users participated in this phase. The basic problem of this subsystem can be summarized with this example: The Giralda. This Arabian tower can be analysed under different points of view. An archaeologist, for example, will be interested in a set of attributes of the tower different from those an architect will be interested in.

This subsystem is complex because the different terminology used by each expert makes difficult the analysis of requirements conflicts. For this reason, it was the subsystem selected to present and to study our approach. We next analyse how the requirements gathering phase of this subsystem was executed:

Step 1. Selecting experts. We first selected a group of 10 experts from 60 final users. This group included people who worked in historic heritage research, diffusion and management. We also included some other experts such as archaeologists, architects, ethnologists and art historians.

Step 2. Brainstorming alone for presentation. Each specialist was required to define the elements needed under his/her view and criteria.

Step 3. First conciliation. The requirements analyst group received each proposal from each expert in a spreadsheet document with attributes grouped by categories and typologies. Manually, they review each approach and decided to organise historic monuments in three groups:

a. Immovable heritage, which included physical monuments, like The Giralda.
b. Movable heritage, which grouped physical but movable monuments such as historic pictures or statues.
c. Immaterial heritage, which grouped non-physical historic heritage, for instance, the Flamenco.

Step 4. Second conciliation. After organising each proposal, a set of attributes was defined for each group. They were sent to each expert and one week later we started to work together to find out inconsistencies. We hold seven meetings of three hours working in liaison to define the set of attributes for each kind of historic heritage. During these meetings**,** we defined each attribute and those which presented inconsistencies, for instance in terminology, were discussed.

Step 5. Brainstorming alone for presentation. After deciding which elements were stored in each kind of heritage, experts were required to decide how this information had to be presented to different end-users.

Step 6. Third conciliation. The requirements analyst group received each proposal and, again by hand, tried to define a conciliated model. According to them, attributes were grouped in different sets depending on their nature. In total, seven different groups were defined. For instance, *Basic information group***,** that presented the essential data on historic heritage such as its name or the *description information group***,** which presents information that described the monument, such as its short history or its historic style.

Step 7. Final Validation. The result of the third conciliation was presented to the experts and, again, some meetings were celebrated. After eight meetings of two hours, the model was closed.

The group of analysts that participated in this process was composed by three experts: a project manager and two analysts.

We have no information about the number of hours dedicated by experts to this process, although the complete number of hours required by the analyst group was registered. The complete process took 34 work-days. During these days, the project manager dedicated 20% of the time and analysts worked at 100%.

Table 1 presents the total result in hours, with an average of eight hours per day. We calculated the cost of an hour according to the official cost used by the Andalusian Government in 2005 [15], when this requirements phase was executed.

Role	Dedication	Hours per day	Total of hours	Cost per hour (€)	Total cost (€)
Project manager	20	1,6	54,4	**68,20**	3.682,80
Analyst 1	100	8	272	**51,16**	13.915,62
Analyst 2	100	8	272	**51,16**	13.915,62

Table 1. Initial Cost

According to that, the total cost for the analyst group in this aspect was 31.513,84 € (Euros).

Checking the dedication in each activity, the result was the as follows:

Task	Project manager	Analyst 1	Analyst 2
Meetings with users	37	37	35
First Conciliation	6	80	82
Third Conciliation	5	93	101
Documentation	4	47	42
Others	2,4	15	12

Table 2. Detailed cost by phases

2.3 Our Research Goals

In view of the results shown in Section 2.2, one can easily understand why the requirements elicitation task is one of the most expensive tasks in the development process. In addition, when a conciliation of requirements is necessary costs are significantly increased.

The analyst group dedicated too much time to the first and third conciliations; it supposed 367 hours for the analyst group, almost 61% of the work. It was only executed by analysts, without experts, and it aimed at reaching a consensus on the results obtained from different users. This pattern does not only appear in Mosaico. In fact, for the complexity of this system, this problem is very relevant but the necessity of requirements conciliation is crucial in software development.

We aim at improving time and effort saving using a model-driven approach for modelling requirements that will help requirement gathering tasks. In this work, we will assess the use of the model-driven paradigm and the definition of a requirement conciliation process that systematized requirements conciliation. We will study in next sections how model-checking can help reducing required effort for detecting and resolving conflicts that can be automated. In the following sections we analyse how these activities can be improved.

3 Related Works

Both the analysis and detection of conflicts, errors and mistakes in the requirements phase are the most critical tasks in Requirements Engineering [26]. Although there are several approaches for requirements treatment, a global view presented in [11] divides this phase in three main tasks: requirements capture, requirements definition and requirements validation. The detection of conflicts is normally executed in the last one. In [11] the authors surveyed the way in which Web Engineering approaches treated these three phases and concluded that most approaches use classical requirements techniques to deal with requirements. According to these techniques, there are four main techniques for requirements validation: reviews, audits, traceability matrix and prototypes; in the Web Engineering literature, requirements

validation is one of the less studied subjects. Besides, none of these techniques offers a systematic detection of conflicts in requirements.

In the broad field of software engineering, [29] enriches this set of techniques adding requirements test. It consists in the generation of early test cases derived from requirements, which enables the early validation with users.

Several works related to requirements validation can also be mentioned in this section, mainly, if we do not only focus on Web methodologies.

For instance, Leite [19] analyzes how relevant is a natural language in a systematic way is so as to improve the communication with the user. In [20], the idea of reviews based on requirements validation under different point of views is recommended. In this sense, they introduce the concept of Viewpoint *as a standing or mental position used by an individual when examining or observing a universe of discourse* and they focus on the importance of assessing requirements under this different point of view.

Silva & Do Santos [28] propose the use of Petri Nets as a specific technique to validate the consistency of requirements defined as use cases. This approach generates Petri Nets from use cases and studies their consistency. It seems quite interesting as it tries to normalize requirements validation with an important constraint, since it is oriented to use cases described with a very specific notation. This technique cannot be used, if any special extension of use cases operates or other techniques to describe requirements are applied.

However, this literature is not specific focused on techniques to try to sure the consistency of requirements.

In this sense, Katasonov and Sakkinen [16] integrate some of the most classical requirements validation techniques to cover a wider concept: the requirements quality. They highlight the importance of both, requirements validation and requirements verification, to assure the quality of the requirements phase. With this aim, they present a framework of different techniques to review requirements by incorporating prototypes, requirements testing and reviews. They present a complete taxonomy for this framework and explain how to apply their approach. However, the lack of a practical evaluation hinders the application of their ideas.

In the Web Engineering field, the situation is not different. Despite some methodologies improved their requirements phase in the last years, the study of the requirements remains too “handcrafted” and non-systematized yet. Thus, recently, some Web design approaches, such as WebML[4] and NDT[8], improved requirements management by using the model-driven paradigm. Nevertheless, even offering systematic (or even automatic) support for early testing, the detection of inconsistencies in the specification of requirements still depends on the analysts’ experience and their capability to support the review with customers and users. A thorough survey presented in [31] describes how each Web Engineering methodology handles Web requirements specification. Most approaches depict requirements using use cases documents based on informal textual descriptions without any support for reasoning over the resultant documents. In this survey, NDT appears as a leading methodology providing a requirement meta-model. In this paper, we show how this meta-model will help saving time and effort by allowing reasoning on requirements models.

Focusing only on the detection of conflicts, [3] presents an approach to detect concerned conflicts. The authors propose using a Multiple Criteria Decision Making

method to support aspectual conflicts management in aspect-oriented requirements. It results limited since it points out the treatment of aspect-oriented requirements and it only deals with concerned conflicts.

In other phases of the life cycle, the conflict-detection process has been deeply studied by the model-driven community mainly based on UML model conflicts. In [2] the author proposes detecting conflicts in a twofold process: analysing syntactic differences by raising candidate conflicts and understanding these differences from a semantic view. In [32] an approach based reasoning on logic descriptor; UML models are transformed into logic descriptor documents that are later processed by a first-order logic engine in charge of reasoning.

In [27], Sardinha et al. present a tool for identifying conflict in aspect-oriented requirements called EA-Analized that process Requirement Definition Language (RDL) specifications. By classifying text using Naive Bayes learning method, it is possible to detect conflict dependencies with high accuracy. Despite of this work, our work infers over Web requirement model in order to detect conflicts. We believe both Sardinha et al. and our works are complementary in the point of view that [27] helps detecting FR and NFR conflicts and our work focuses on detecting navigation ambiguities.

4 Background

4.1 A global vision of NDT

NDT is a model-driven Web engineering approach. Initially, NDT dealt with the definition of a set of formal meta-models for the requirements and analysis phases. In addition, NDT defined a set of derivation rules, stated with the standard QVT [25], which generated the analysis models from requirements model.

Subsequently, the methodology was improved and nowadays, NDT defines a set of meta-models for every phase of the lifecycle 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 (these new models are known as basic models). These transformations are identified by the stereotype *«QVTTransformation»*. Figure 1 shows the first part of the NDT lifecycle.

After carrying out these transformations systematic, NDT allow analysts can carry out transformations in order to enrich and complete this basic model**.** Transformations are represented in Figure 1 through the stereotype *«NDTSupport»*.

In the last years, NDT has evolved again and now, NDT offers a complete support for the whole development life cycle; additionally, In the last year, it evolved to support different types of life cycles such as sequential, iterative and agile processes. For the sake of conciseness we will focus mostly on those aspects of NDT related with requirements.

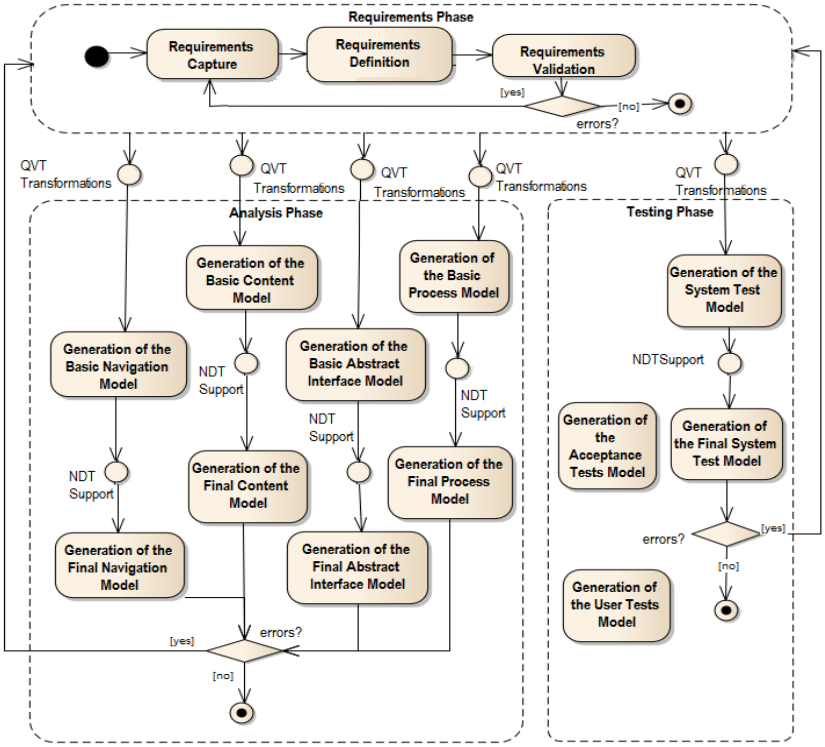

Fig. 1. First part of the NDT sequential lifecycle.

On the other hand, an important number of companies in Spain, such as Icosis[1], Everis[2], Emasesa[3], and some institutions such as the Andalusian Regional Government, Emasesa and other, work with NDT and the associated tools for software development. This is possible due to the fact that NDT is completely supported by a set of tools, grouped in the NDT-Suite [23][13]. NDT-Suite works on/with a UML-based tool named Enterprise Architect (EA) [7]. To select Enterprise Architect did not result an easy task. In fact, a comparative study developed by our research group and the Andalusian Regional Government concluded that this was the tool that offered the best ranking in price/quality[4]. Furthermore, EA offers several important advantages, such as the possibility of defining profiles or tools for document management by drawing UML diagrams, for instance, which have been very relevant to carry out our work.

NDT-Profile is the main tool of NDT-Suite. NDT-Profile is a specific UML-profile for NDT developed by means of Enterprise Architect. NDT-Profile offers the chance

[1] Icosis's website is http://www.icosis.es
[2] Everis's website is http://www.everis.com/
[3] Emasesa's website is http://www.aguasdesevilla.com/
[4] This study was written in Spanish. It was not published but can be asked in www.iwt2.org.

of having all the artefacts defining NDT easy and quickly, as they are integrated within the EA tool. Apart from this tool, NDT-Suite integrates the following main tools:

- NDT-Quality. It is a tool for measuring automatically the quality using NDT methodology. 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. It also provides a report in different formats describing the inconsistencies appeared during the review.
- NDT-Driver is a tool that allows the application of QVT transformations in NDT. For instance, in the requirements phase, NDT-Driver enables the automatic generation of transformations presented in Figure 1.

In addition, NDT-Suite has more tools: NDT-Report, NDT-Glossary, NDT-Checker and NDT-Counter. You can see the purpose of these tools on IWT2 website[5].

In conclusion, NDT-Suite enables the definition and use of every process and task supported by NDT and offers relevant resources to develop software projects in terms of quality assurance, management and metrics.

Moreover, just as we have commented previously, NDT uses a set of meta-models for each development phase (requirements, analysis, design, implementation, construction, test and maintenance) in order to support each artefact defined in the methodology. All concepts in every phase of NDT are meta-modelled and formally related to other concepts by means of associations and/or OCL constraints [24], as it is presented in [11]. In this paper, we focused on the NDT requirement meta-models, which are presented in detail in the next section. NDT implements its meta-models with a set of profiles represented in NDT-Profile in order to offer a mechanism to use them.

4.2 Supporting RIA features with NDT

RIAs (Rich Internet Applications) have particular features like sophisticated interactive behaviour, client-side feedback of "slow" operations and different kinds of client-side behaviour depending on the occurrence on the events, among others. For this reason, the NDT requirement meta-model was enriched with these concepts as shown in Figure 2.

In NDT, the original packages, *structure* and *behaviour*, were kept to preserve the mapping between the concepts present in NDT and its ancestors.

In the structure package each concept deals with the conceptual aspect of Web requirements. Since RIA applications are specially focused on client-side behaviour, we added the *UIElement* metaclass. Instances of this metaclass are: *buttons*, *textfields*, *images*, *checkboxes*, etc

The *behaviour* package includes metaclasses to represent users' interaction and navigation. We extended the package with the *RIAEvent* metaclass that is important to clarify different situations; for example, when the user places the mouse over an item or when the user types something on a field. In this case, we differentiate between two

[5] www.iwt2.org

different subclasses: those events originated with the keyboard (subclass *KeyboardEvent*) and those originated with the mouse (subclass *MouseEvent*).

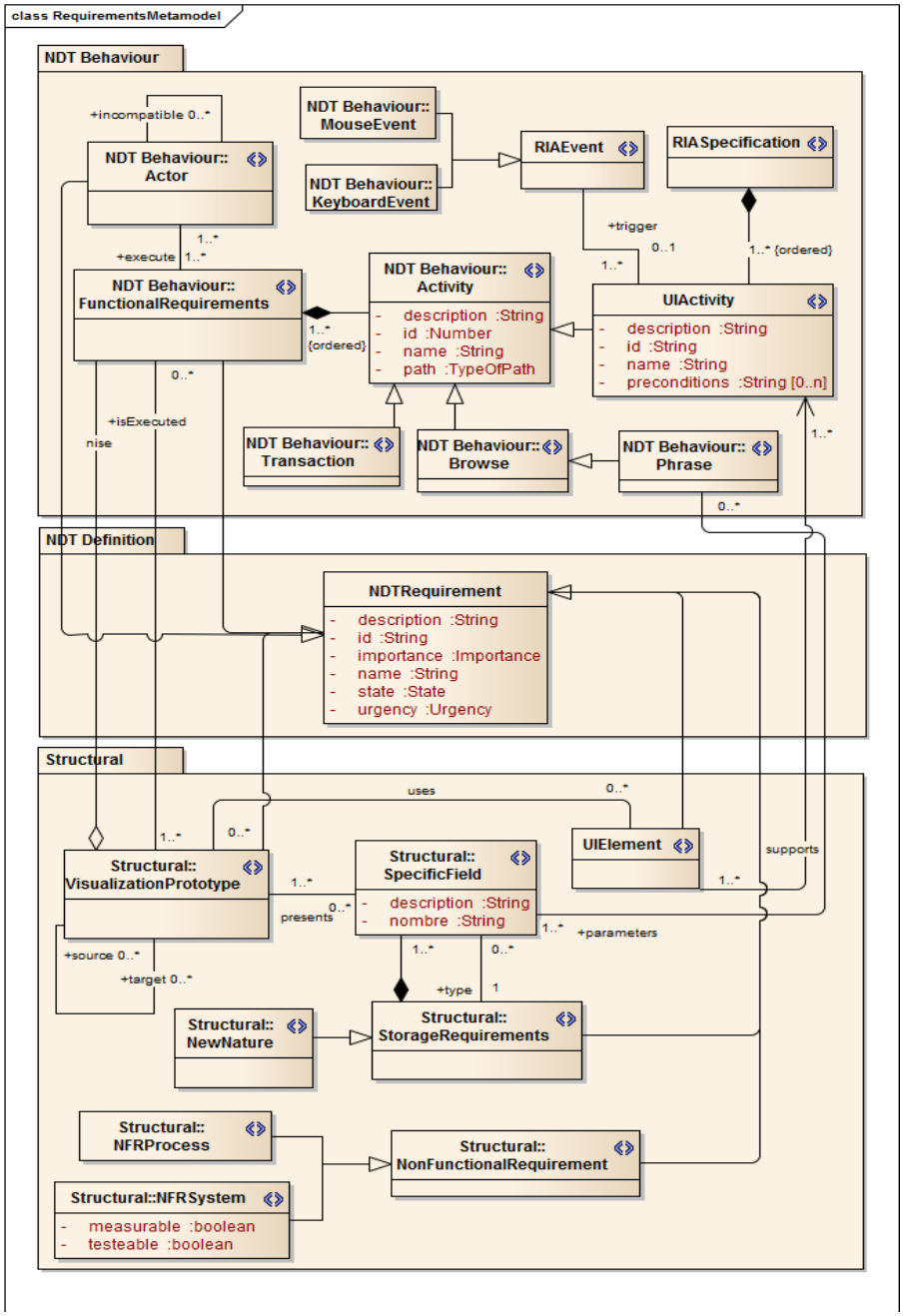

Fig. 2. NDT Requirement meta-model

Additionally, we include a new metaclass *UIActivity* which captures the actions that the user can perform over an element in the user interface of the application (relationship between *UIActivity* and *UIElement*). Instances of *UIActivity* are "click", "type keys", and execution of one of the actions may produce many events, e.g. when typing a key on a user interface element three events are fired, namely *onpressdown*, *onpresskey* and *onpressup*. *UIActivity* are grouped in concrete scenarios, which are defined as instances of the *RIASpecification* class.

4.3 Initial Hypothesis: Characterizing Requirements Conflicts in Web Applications

During requirement specification, there may be cases where two or more scenarios that reflect the same business logic differ subtly from each other producing an inconsistency. When these inconsistencies are based on contradictory behaviours, we are facing a requirements conflict [14]. Conflicts are characterized by differences in objects' features, logical (what is expected) or temporal (when it is expected) conflicts between actions, or even difference of terminology that creates ambiguity.

In this analysis we will emphasize Web application navigation, as well as users' interaction peculiarities that are not covered in the traditional characterization of requirement conflicts [14]. Consequently, we provide an interpretation of each conflict type on the Web application realm by means of simple but illustrative examples.

(1) **Structural conflicts:** They stand for a difference in the data expected to be presented on a Web page by different stakeholders. A stakeholder may demand a data to be shown on a Web page that contradicts other stakeholder's requirement. For example, none of them expects a product content description just as a read-only label, while another one may expect the content as a list of packaged items with an overall description contradicting the first requirement.

(2) **Navigational conflicts**: They take place when two Web application requirements may contradict the way in which links are traversed producing navigational conflicts, e.g. having a single source node but two targets. The target nodes are different, although the events that trigger the navigation and the context restrictions are the same, which poses an ambiguity of such requirement.

(3) **Semantic conflicts**: They occur when the same real-world object is described with different terms. This situation may generate a false negative in the conflict detection process since a conflict may not be detected and new terms are introduced into the system space thus increasing its complexity. As a consequence the same domain object is modelled in two entities with different terminology.

5 Our approach for detecting requirements conflicts in NDT

We propose a five-step approach for detecting requirements conflicts when NDT is used. Below, we explain each one of these steps together with some examples for clarifying our method. The examples described are based on the Mosaico project (see Section 2).

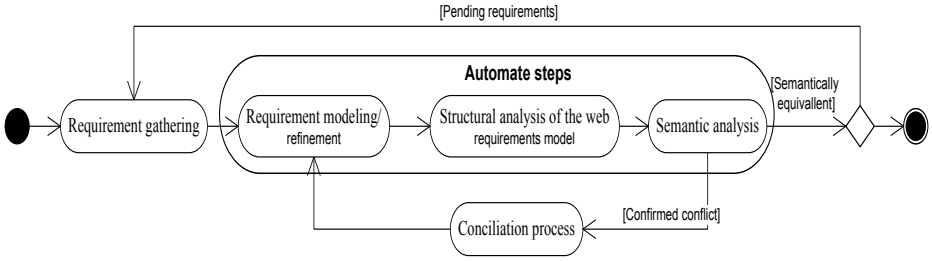

Fig. 3. The overall process for detecting requirement conflicts.

The process is applied iteratively each time a new set of requirement rises. The new incoming set of requirements is checked with each of the already consolidated requirements of the system space. In Figure 3, those steps that can be implemented to become automated are grouped in a dashed ellipse.

These steps could be integrated in different requirements meta-models. In [30] we have applied it with WebSpec.

Step 1 and 2. Requirement Gathering and Requirement Modelling

We propose to combine classical capture requirements techniques such as interviews or brainstorming (see [8]) for the requirements gathering; for the requirements modelling, we propose NDT-Profile. When analysts have completed the requirements catalogue represented in NDT-Profile, they should execute the next steps with the aim of detecting requirements inconsistencies.

Step 3. Detecting Syntactic Differences

A candidate conflict arises when the set of syntactic differences among requirements appear. These differences may appear as a consequence of: (*i*) the absence of an element in one model that is present in the other; (*ii*) the usage of two different artefacts for describing the same information; or (*iii*) a configuration difference in an element such as the properties values of an artefact. This situation may arise when two different stakeholders have different views of a single functionality, or when an evolution requirement contradicts an original one.

Structural conflicts detection can be implemented by a comparative operation between interactions, represented in NDT meta-model under *VisualizationPrototype* metaclass (see Figure 2), in order to detect the absence of elements or elements configurations differences. For instance, in Figure 4.a and 4.b, different mockups so called *Option A* and *Option B* are illustrated for the same set of requirements of the Mosaico project. These mockups are concrete instances of the *VisualizationPrototype* metaclass generated by the tool NDT-Suite. They present the same necessity of looking for concrete moveable heritage of a specific historical period by using the artefact FR of NDT[6], which represents a kind of interaction requirements. *Option A*

[6] The original prototype of Mosaico contains more fields for searching. In this figure, we only present a simple example to explain the approach.

was proposed by archaeologists and the *Option B* was proposed by art historians. For archaeologists the grade of certainty of the date of the moveable heritage is essential in the search because they mainly work with very old pieces, which are very difficult to date. However, it is not so relevant for art historians because their work in pieces that are more recent. Despite art historians use the grade of certainty as an attribute for moveable heritage, it is not a relevant field for searching for them. However, for art historians, they need to include the name of the author in the search, which is completely irrelevant for the archaeologists.

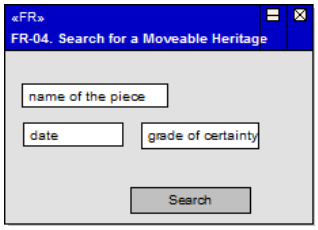

Fig. 4.a. Mockup for archaeologists - Option A

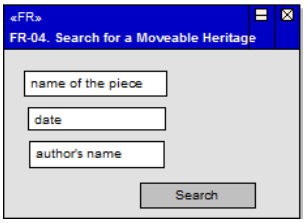

Fig. 4.b. Mockip for historians - Option B

$FR\text{-}04^{A}_{diff} = FR\text{-}04^{A} - FR\text{-}04^{B} =$ {grade of certainty}

$FR\text{-}04^{B}_{diff} = FR\text{-}04^{B} - FR\text{-}04^{A} =$ {author's name}

Inconsistencies= $FR\text{-}04^{A}_{diff} + FR\text{-}04^{B}_{diff} =$ {grade of certainty, author's name }

Fig. 4.c. Mockup differences detection algebra

Since NDT *interaction requirements* are defined with concrete artefacts that are stored in this tool, we can apply set's difference operations in order to detect inconsistencies. In *Option A* of Figure 4.a, a Search for Moveable Heritage version called $FR\text{-}04^{A}$ includes *name of the piece, date* and *grade of certainty*, and a *Search* Button. On Option B of Figure 4.b, a different version called $FR\text{-}04^{B}$ comprises a *name of the piece, date, author's name* and *Search* Button. In Figure 4.c, *Algebra* section shows how these differences are detected by means of set's difference operation.

It can be noticed that for the comparative operation, two elements are equal if and only if they have the same identifier, the same artefact type and compatible configuration.

Outgoing navigations from a given node with identical triggering events but different targets must be checked in order to detect navigational conflicts. The task is pretty straightforward; since *navigations* are described by an artefact with a set of context constraints and a set of actions that trigger them, the *navigations* for a given *interaction requirements* must be compared to each other taking into account their context constraints and set of actions. The main challenge of this procedure is to check whether the sets of actions that correspond to navigations are semantically equivalent, given that the actions can be syntactically different.

Bellow, we introduce an analysis process that helps avoiding false positives.

Step 4. Semantic Analysis

As the result of the structural analysis of models, a list of candidate conflicts is reported; this list must be verified in order to detect false positives (i.e. conflicts that actually are not conflicts since the compromised specifications describe the same

requirement). This issue has been already studied in [2] and [21] where models are analyzed in order to expose their underlying goals. When the underlying goals they are different, we are facing a confirmed conflict.

On the other hand, there are requirements that can be documented twice in different NDT diagrams duplicating specifications and injuring requirement traceability. These cases are also studied in this process.

We use an approach proposed in [1] which focuses on having an additional semantic view of requirements that complements the existing syntactic view. For achieving this, requirements models are downgraded in terms of abstraction, obtaining a simplified model formed only by semantically simple elements.

This approach is twofold: a meta-model called *semantic view*, in this case it is NDT requirement meta-model without those meta-model elements that give RIA support, and a transformation from the source model to one that obeys the *semantic view*.

For each detected conflict, the compromised models (the new and the stable one) are transformed into a semantic view where the derived models are finally compared syntactically. This approach avoids false positives because the semantically equivalent constructions compositions are disambiguated.

As a semantic view, we will use a reduced NDT requirement meta-model based on the meta-model shown in Figure 2 where no RIA behaviour is taken into account and, on the contrary, it focuses on allowing modelling basic user-interaction aspects. That is, the meta-model aims to provide a simplified view in which traditional navigation and RIA interaction are abstracted in a more generic interaction concept. The new model removes *Event´*s hierarchy, *RIASpecification* feature and *Activity's* hierarchy, but *Browser* class as well as any orphan relationship in the meta-model arise out of removing classes.

Finally a model transformation must turn a NDT model into a semantic one in order to provide a simpler understanding. In the transformation, a set of rules closely related to the Web requirement meta-model used is applied on the input model obtaining the semantic view.

Some of the rules for NDT meta-model comprised by the transformation are:

- Disabled *SpecificFields* that are presented in the *VisualizationPrototypes* are translated to Labels. As disabled TextFields do not allows user inputs these are replaced by simple Labels.
- *target* relation in *VisualizationPrototypes* are translated to Buttons. Links and Buttons are usually used for describing an action triggering. Therefore, this relation that expresses navigation is normalized to Buttons.
- *RIASpecification* are simplified into a single *Browse*. This rule makes the diagram focus more on the data itself instead of the way in which it is accessed. Finally, *Activity* specifications are removed.

If other Web requirement meta-model is used, a different set of rules must be defined where each one must increase the abstraction level in such a way the intent of the model is emphasized. For example, each disabled *TextField* is transformed to Label widget. In [30] a specific set of rules was developed for WebSpec meta-model.

A syntactic conflict is raised from two different views of the same requirement: one based on labels and links (Figure 5.a) and other based on disabled *TextFields* (no input is allowed) with buttons (Figure 5.b).

In order to detect if the syntactic conflict is in fact a conflict, the semantic transformation is applied over both requirement specifications. Both transformations produce the same model that is formed by *Labels* and a *Button*. Thus, as both semantic views are equal, there is not conflict at all.

The result of applying the transformation to both conflicted NDT diagrams is a pair of normalized diagrams that must be syntactically compared in order to detect differences.

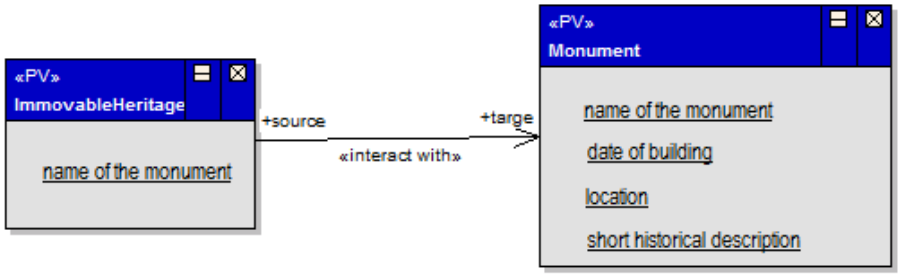

Fig. 5.a. Navigation from a ImmovableHeritage to Monument *VisualizationPrototype* based on *Labels* after clicking over a name item.

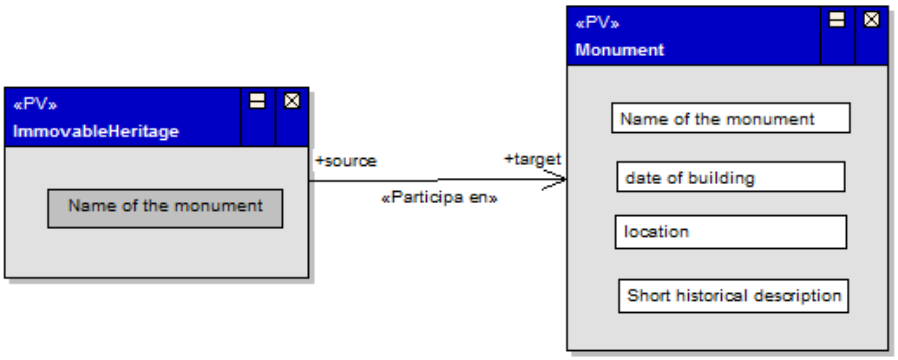

Fig. 5.b. Navigation from a ImmovableHeritage to Monument *VisualizationPrototype* based on *TextFields* after clicking over a name item.

Let's use standard UML models removing NDT enhancements and obtaining a graph of objects that will be used for describing the technique. Figure 8 shows the unique result of applying the transformation to the examples presented in Figure 6 and Figure 7 where *Phase* and *UIAction* were normalized into the more abstract *Activity*, and the *Home* link was removed because it is not referenced anymore.

Then a semantic conflict is detected because both models are semantically equivalent; in our tool a warning is produced in order to choose one of both models.

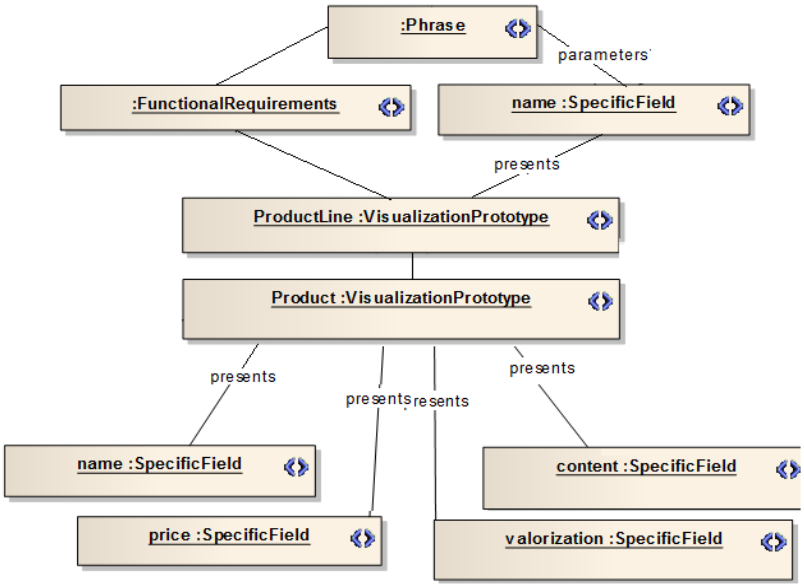

Fig. 6. Specification of conventional navigation requirement.

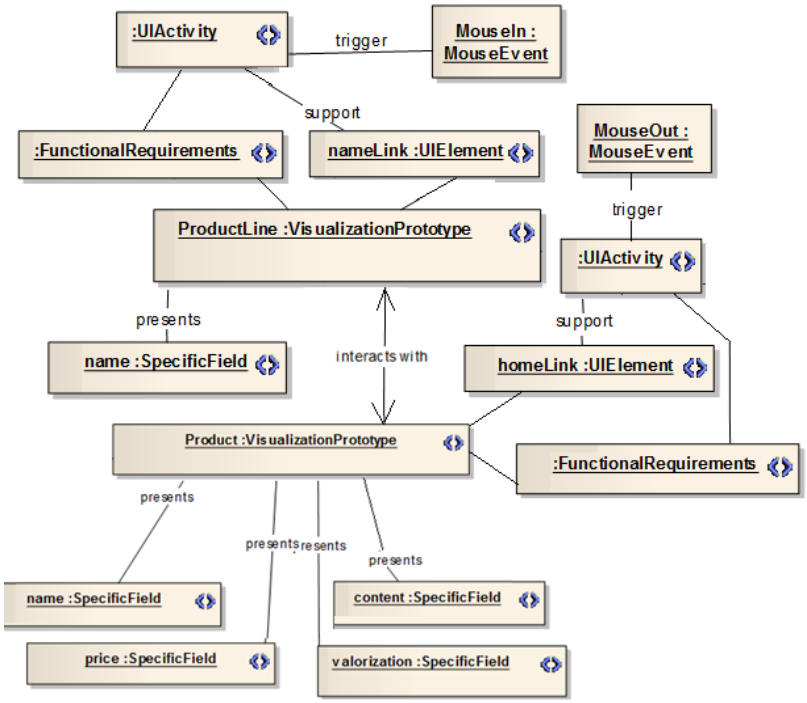

Fig. 7. Interaction based on a RIA feature.

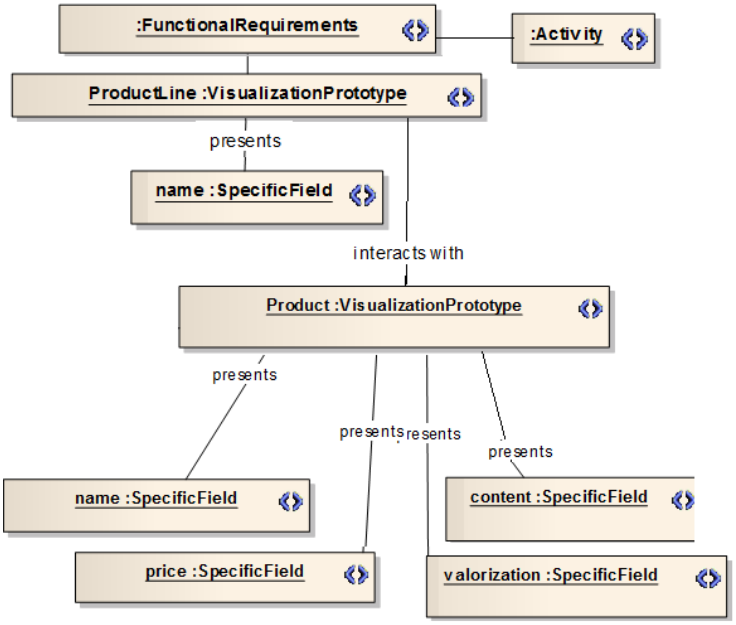

Fig. 8. Normalized diagram into Semantic view after transformation.

Step 5. Conciliation Process

So far, we have shown how to detect conflicts that must be resolved in order to keep the requirements document sound and complete. Next we will introduce a set of heuristics that helps resolving structural and navigation conflicts that have been implemented as suggested refactoring.

Structural Conflicts

When facing structural conflicts, there are NDT artefacts that may differ in their type or configuration. For example, an expected data can be realized as a read-only interaction element such a *Label* and, in another stakeholder's point of view, it may be a writable data modelled in a *TextField*.

In cases where a given artefact is absent in a model but present in the other, we can take an optimistic position understanding that the best solution is to include the construction as an improvement when it is not present. This idea comes from the fact that new requirements may improve others requirement´s functionality; therefore the new requirement artefact may enrich an existing interaction.

On the other hand, the artefact type incompatibility demands a deeper analysis, if the context of difference is considered:

- Read-write over Read-only widgets. It may happen that the structural comparison exposes a contrast between read-only widget (or disabled *TextField*) and a *TextField*. In this case, we choose the most flexible one: use an enabled TextField allowing to show and edit data.
- Fixed data values range over wide values range. Two widgets may deal with the same data but differ in the manipulated range; masked text inputs and restricted set of options are examples. In this case, restrictive widget such as *Combobox*, *RadioButton* or masked *TextFields* are prioritized over less restrictive widgets.
- Container vs. atomic widgets: When having one *VisualizationPrototype* specifying a Container that defines an aggregation of data against a non container widget such as a *TextField*, Containers must be preserved because they establish a detailed information structure specification. These are instances of the *SpecificField* metaclass (see Figure 2).

Navigational Conflicts

Navigational conflicts express ambiguity in the way in which the Web application is browsed, when two stakeholders express different navigation in a same context. This situation is naturally resolved by enriching the scenario in such a way that the conflict is dissolved by increasing the scenario detail. In NDT context, there are two strategies available for disambiguating: either adding context constraints or extending the scenario path; both increase the scenario detail.

As we have previously seen, different stakeholders may provide slightly different specification for the same application goal. Nonetheless, there are scenarios that are prone to face inconsistencies such as the presence of business objects hierarchies. At the requirement elicitation stage, business objects hierarchies may not be clearly detected and defined, and as a consequence, several structurally different business objects are referenced with the same name.

Conciliation cycles are strictly related to the introduction of a new requirement in the system. As it is shown in Figure 3, each time a new requirement is identified, it is modelled, and later syntactically and semantically analyzed. When an inconsistency is detected, it must be resolved, if possible, following the previously proposed conciliation rules or by means of meetings with stakeholders for disambiguating the situation. Once there are no more conflicts, the requirement analysis process can start over again the analysis cycle for new requirements

The set of presented heuristic helps to easily modify a given model that tries to resolve a conflict; nonetheless, it can introduce inconsistencies when the outcome of chosen refactoring contradicts other requirement. In order to avoid any inconsistency, model checking process only finishes when there are no conflicts. Additionally, when there is no valid heuristic like this case, our approach suggests resolving conflict by means of meetings with stakeholders.

6 Experimentation and Validation

This section enables the experimentation and validation of our approach. According to the introduction presented in section 2, we executed an experiment to value the suitability of our approach in the real environment of Mosaico.

6.1 Improving Mosaico with requirements conciliation

Our approach tries to reduce the effort required by analysis tasks; that is 367 hours of dedication. The approach presented in Section 5 systematizes the detection of these conflicts. For that, we are going to analyse how this approach could have improved steps three and six, that is first and third conciliation, originally developed.
In order to carry out our experiment, we proposed two junior analysts to apply our approach in the context of Mosaico. For this experiment, we followed the next steps presented in Table 3 bellow.

1. We presented our analysts the approach of the conciliation approach, detailing each step.
2. As we cannot replicate the requirements gathering and requirements modelling, we provide them with the results of these phases, step 1 and 2, introduced in the original requirements elicitation of Mosaico. Thus, they started with the requirements modelled in NDT-Profile by each group of users.
3. Starting with these requirements, they had to apply the first conciliation of the process. This application is explained in detail in Section 6.1.1.
4. Expert analysts that participated in Mosaico requirements phase reviewed this conciliation, simulating users' review.
5. With the result of this review, we asked junior analysts to the third execution of our approach, which is presented in detail in Section 6.1.2.
6. The results of this conciliation were reviewed again for analyst experts.

Table 3. Experiment scenario

We measure the times of steps 3 and 5 of this scenario in order to have comparable results with the original conciliation of Mosaico. Besides, we collected results of reviews in steps 4 and 6 in order to compare the effectiveness of the application.

In the next sections, the first and third conciliation (steps 3 and 5 of the experiment scenario) will be presented in detail.

6.1.1 First Conciliation

During this step the analyst group was mainly focused on detecting structural and navigational conflicts. Each expert delivered the analyst group a set of prototypes, developed by users with the set of attributes for each group of historic heritage. Each group received more than 100 attributes and this provoked a high number of manual checking.

This revision could have been improved by applying requirement consistency checking in order to obtain, for instance, a structural conflict similar to the one presented in Figure 4.

During the review, for instance, our junior analysts detected a structural problem presented in Figures 9 and 10. These Figures offer prototypes developed by the analyst group following archaeologists' guidelines (Figure 9) with id $VP\text{-}01^A$ and artists' guidelines (Figure 10) with $VP\text{-}01^B$:

For archaeologist, the name of the authors is not relevant; authors of archaeological sites are normally unknown. However, for artists it is a basic aspect.
The application of our approach in this phase could reduce the number of hours executed to check structural consistencies. For that, we could calculate, for instance for the previous example:

$VP\text{-}01_{diff} = VP\text{-}01^A - VP\text{-}01^B$ = *{Activities, Authors, Copy of, Origin, Archaeological Context, History of the place}*

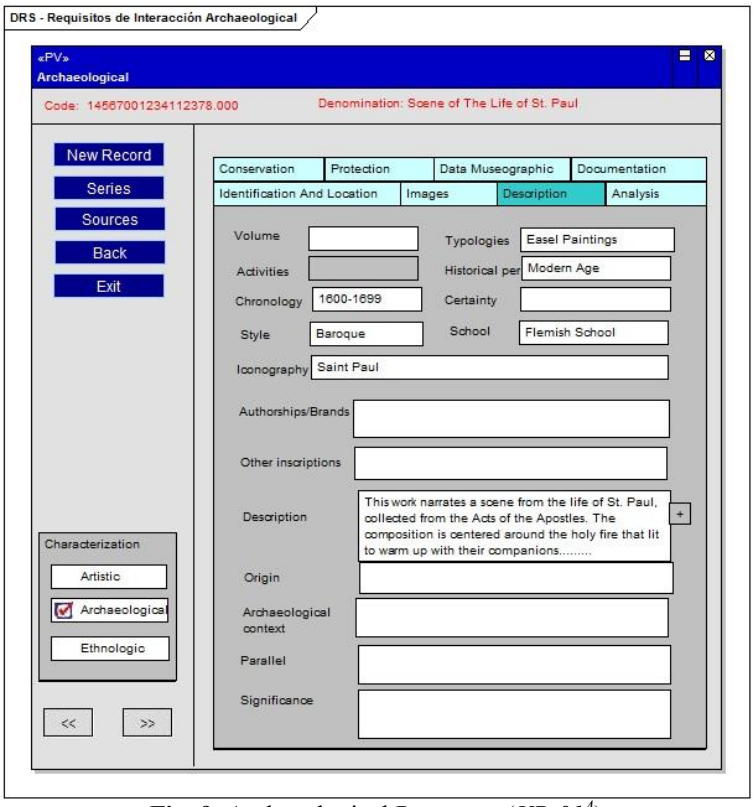

Fig. 9. Archaeological Prototype ($VP\text{-}01^A$)

By means of our approach, the junior analysts detected 100% of inconsistencies produced in Mosaico. Even, they detected some errors that in the original

requirements phase of Mosaico were not detected in the first review. In this sense, the approach succeeded.
Despite we presented in Figures 9 and 10 a prototype, our junior analyst did not work with them directly. It would produce a high cost for the manual review. The manual comparison of these prototypes is not easy; in fact, our junior analysts prepared a list with every prototype defined by the user and the attributes presented by them. This list was generated directly from the data base of NDT-Profile. NDT-Profile stores each artefact and element in an internal database. Thus, in fact, they compared a list of artefacts and attributes. It made easier the manual review. This work with the database directly is one of the mechanisms proposed in the conclusion as a future work because this comparison could be automatic using directly the database.

6.1.2 Third Conciliation

In the third conciliation, we asked experts to explain how information should be presented to the user, who can manage it and how an end-user could navigate spreading out this information. In this case, conciliation was oriented towards detecting navigational and semantic conflicts.

The junior analysts received the original set of visualization prototypes in NDT developed by each expert stating how information will be presented.

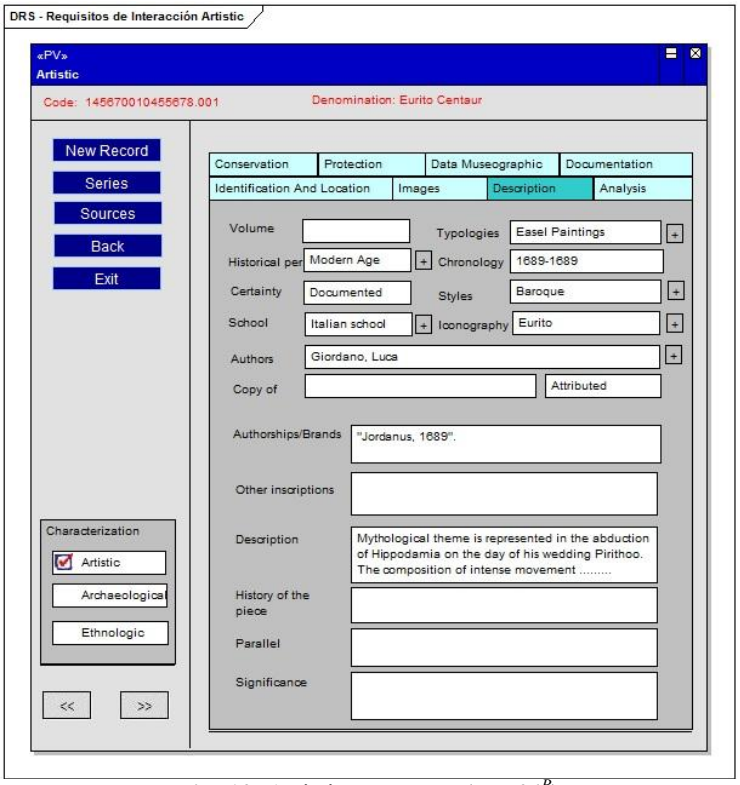

Fig. 10. Artistic Prototype ($VP\text{-}01^{B}$)

We could improve this phase if our approach is used to detect navigational and semantic conflicts. For instance, reviewing the navigation, we offer our junior analysts an initial prototype to the ethnological group based on prototypes developed for artists (Figure 9). However, for them, information was not presented in the same order. In fact, originally, they proposed a new set of attributes, represented by activities, which group information on how the monument was or is used, as it is presented in Figure 11.

Our analysts studied a total of 98 attributes, which were initially presented as three proposals, one for each specialist, with the attribute grouping and names that each of them selected. Again, they did not work with the visualization prototype directly, they worked with the information in the database. In the experiment, as it is comment, all syntactic and navigational inconsistencies were detected. However, the semantic problems were not automatically detected.

For instance, at the beginning, ethnologist named *activities* as *uses*, whereas archaeologists and artists referenced like *activities*. In our experiment, it was detected as a syntactic error in the first conciliation. Nevertheless, it did not seem different to the remainder differences.

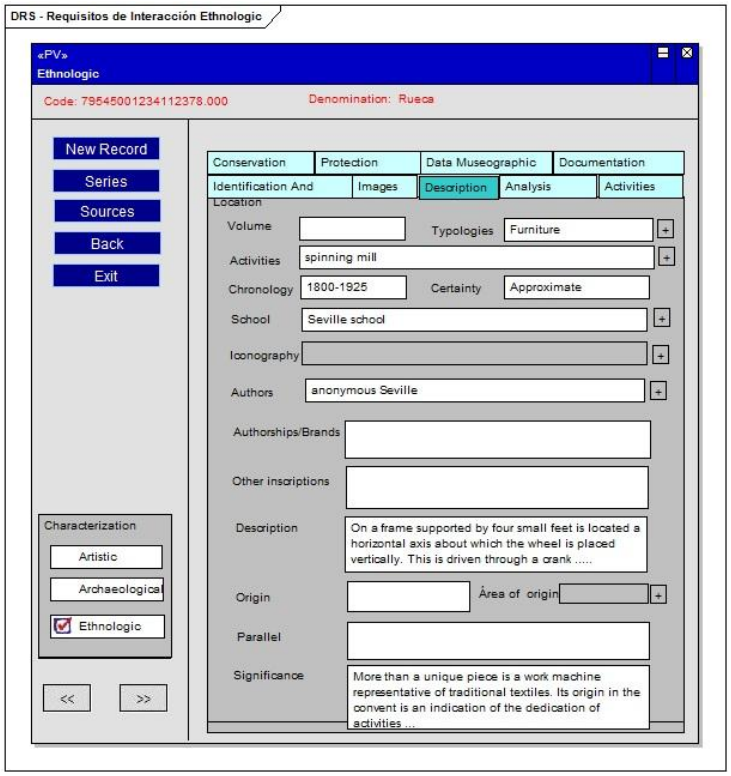

Fig. 11. Ethnologic Prototype

6.3 Measuring the improvement

Mosaico is now an implanted system that it is being used by a high number of users. For this reason, we cannot replicate the whole process. However, we could do our experiment to measure the grade of efficiency and effectiveness reached when applying it.

About the efficiency, the improvement was clear. In Table 4, we present hours used by each junior analyst for the first and third conciliation (step 3 and 5 of the scenario described in Table 3).

Task	Analyst 1	Analyst 2
First conciliation	16 hours	19 hours
Third conciliation	22 hours	27 hours

Table 4. Detailed cost by phases

Obviously the improvement only can be noticed in this two phases. In the first reduction, the time using the approach is reduced 78,4% and in the third conciliation, it supposes 74,4%. This could have saved in the budget 6.497,32 € in the first conciliation and 7.418,2 € in the third conciliation; this adds up 13.915,52 € (44% of the budget). Obviously, these measures are only a simulation but they offer very attractive results to continue with the incorporation of this approach in NDT.

About the efficiency, the detection of the syntactic and navigation inconsistencies was 100%; that is, our approach detected the same inconsistencies that the original analysis did. However, with the semantic inconsistencies, the approach only detected differences but not inconsistencies.

The performance difference can be argued with the fact of having a method that ruled the experience. Originally, inconsistencies have been analyzed untidy that led analysts to perform this task together with stakeholders. With the approach, most of the work was organized and, although it was tedious to analyst as it will be discussed next, the whole process was controlled and effective.

Additionally, because the lack of a case tool for this approach, analysts leaned on a spreadsheet derived from Enterprise Architect tool. This document stored different data structure obtained in requirement analysis. Using the embedded spreadsheet query engine, it was implemented the most important set of operations (described in section 5) needed to automate analysis tasks. This simple resource ease with the method already described decreased the effort and time spent originally.

6.4 Analysts' review

Obviously, the advantages in the cost described in Table 4 are quite encouraging. However, in the application of our approach, our junior analysts detected some relevant problems:

1. The manual application of the approach is completely a crazy. In fact, they have to check every prototype manually and quoting their words "*It is a very tired and bored task*". Even, using the list of attributes generated from the database, this work is impossible to be executed by hand.

2. The concept of meta-models profiles is in fact very complex for them. They recognized that they used the approach without understanding these concepts and only following the process.
3. They highlight that the application of the approach is quite complex and they think that it could be difficult to be applied with users.

Nevertheless, they also stand out that the approach guides them to establish an objective and structured way to review requirements. Even though they do not have too much experience; they recognize that they feel targeted and coordinated.

7 Concluding Remarks and Further Work

The requirements phase is one of the most relevant phases in the life cycle of a software project. With the increase of complexity of applications, this phase acquires a more relevant role. In Web application, where new characteristics like RIA aspects are incorporated, the situation is similar or even more complex.

One important aspect in this phase, mainly in big projects, is the conciliation of requirements. When there are different final users and different set of requirements, they have been merged in order to obtain conciliate requirements to initiate the system development of the system. However, this task frequently depends on the analyst's experience or is done manually, without a specific a normalized support to develop it.

In this paper, we present the application of a general model-driven approach for the systematic detection of requirements inconsistencies that was initially presented in [30]. This approach is adapted and extended to improve the NDT methodology. The paper presents how with the use of meta-models the initial approach can be adapted to a concrete methodological environment and illustrates it with a real example measuring the improvement that could offer with an empirical example named Mosaico that originally was conciliated by hand without the use of any mechanism to check it.

Results obtained after applying this approach in a project like of Mosacio, open a very attractive line for our research works. As NDT is applied in a high number of companies, our next step is the inclusion of this approach in NDT-Suite, particularly in NDT-Quality. This implementation will improve NDT-Quality. We want to test the implementation in a complementary set of real projects to try to measure, in an objective way, the number of consistencies that are detected. This work will be made with companies and analyst groups that use NDT for Web software development.

The analysts' review pointed out in section 6.4 the need of a CASE tool with a real-time validation that checks inconsistencies as analysts model requirements.

Therefore, we will work on an Enterprise Architect plug-in that helps with conflict detection and conciliation tasks. This solution will be based in the comparison of attributes stored in the database of NDT-Profile.

Besides, we are interested in the improvement of the set of heuristic to solve conflicts in the semantic conciliation process. The automation of this phase could be an important improvement because it consumes a high number of resources in Web development. Additionally, we are completing the approach with a set of ontology matching algorithms[12] in order to improve semantic conflicts detection.

Finally, catalogue of refactoring for recurrent conflicts can be specified. This will help producing a formal knowledge basis for analysts and, once consolidated, each refactoring can be automated in the CASE tool.

Acknowledgements

This research has been supported by the project QSimTest (TIN2007-67843-C06_03), by the Tempros project (TIN2010-20057-C03-02) and by the project NDTQ-Framework (TIC-5789) of the Junta de Andalucia, Spain.

References

[1] Altmanninger, K., Kotsis, G.: Towards Accurate Conflict Detection in a VCS for Model Artifacts: A Comparison of Two Semantically Enhanced Approaches. APCCM 2009:139-146 (2009).
[2] Altmanninger, K.: Models in Conflict - Towards a Semantically Enhanced Version Control System for Models. MoDELS Workshops 2007:293-304 (2007).
[3] Brito, I. S., Vieira, F., Moreira, A., Ribeiro, R. A.: Handling conflicts in aspectual requirements compositions. In Transactions on aspect-oriented software development III, LNCS, Vol. 4620. Springer-Verlag, Berlin, Heidelberg 144-166 (2007).
[4] Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.. Designing Data-Intensive Web Applications. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2002).
[5] De Lucia, A., Qusef, A.: Requirements Engineering in Agile Software Development. In Journal of Emerging Technologies in Web Intelligence, Vol. 2, No 3 (2010), 212-220 (2010).
[6] de Paula, M. G., da Silva, B. S., Barbosa, S. D.: Using an interaction model as a resource for communication in design. In CHI '05 Extended Abstracts on Human Factors in Computing Systems, pp 1713-1716, Portland, USA, April 02-07 (2005).
[7] Enterprise Architect 9.0. Available in www.sparxsystems.com.au. Accessed in April 2012.
[8] 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).
[9] Escalona, M.J., Equipo de Coordinación. Mosaico. El sistema de Información para la Gestión del Patrimonio Histórico Andaluz. Proceedings of XI International Congress on Project Engineering. (2007)
[10] Escalona, M.J., Koch, N.: Metamodeling Requirements of Web Systems. In Proc. International Conference on Web Information System and Technologies (WEBIST 2006), INSTICC, 310--317, Setubal, Portugal (2006).
[11] Escalona, M.J., Koch, N.: Requirements Engineering for Web Applications: A Survey. Journal of Web Engineering. Vol. II. Nº2. pp. 193-212 (2004).
[12] Euzenat, J., Shvaiko, P.: Ontology Matching. Springer, 1 edition, ISBN: 978-3540496113 (2007).
[13] García-García J. A., Alba M., García-Borgoñon L., Escalona M. J.: NDT-Suite: A Model-Based Suite for the Application of NDT, LNCS 7387, M. Brambilla, T. Tokuda, and R. Tolksdorf (Eds.): ICWE 2012, LNCS 7387, pp. 469–472, (2012)
[14] IEEE Recommended Practice for Software Requirements Specifications. IEEE Std 830-1998 (1998).
[15] Junta de Andalucia. Consejería de Empleo. Dirección General de Trabajo y Seguridad Social. Pliego de prescripciones técnicas para la contratación del análisis y

diseño de un sistema de gestión integral de expedientes del centro de mediación, arbitraje y conciliación. pp. 10-11. (2005)
[16] Katasonov, A., Shakkien, M. Requirements Quality Control. A Unifying Framework. Vol. 11. No. 1. Pp. 42-57. (2006).
[17] Kotonya, G.; Sommerville, I.: Requirements engineering with viewpoints. Software Engineering Journal , vol.11, no.1, pp.5-18 (1996).
[18] Leffingwell, D.: Calculating the Return on Investment From More Effective Requirements Management. AMERICAN PROGRAMMER, 1997, VOL 10; NUMBER 4, pages 13-16 (1997).
[19] 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).
[20] Leite, J.C.S.P., Requirements Validation through Viewpoint Resolution. IEEE Transaction on Software Engineering. vol. 17, No.12. pp.1253-1269. 1991.
[21] Li, C., Ling, T. W.: OWL-Based Semantic Conflicts Detection and Resolution for Data Interoperability. ER (Workshops) 2004:266-277 (2004).
[22] McConnell, S.: Rapid Development: Taming Wild Software Schedules. Microsoft Press. ISBN 1-55615-900-5 (1996).
[23] NDT-Suite. Available in www.iwt2.org. Accessed in April 2012.
[24] Object Management Group, Object Constraint Language, Version 2.2, http://www.omg.org/spec/OCL/2.2/. Accessed in April 2012.
[25] Query/View/Transformation. Available in www.omg.org/spec/QVT/1.1/. Release 1.1. 2011. Accessed in April 2012.
[26] Robles, E., Garrigós, I., Grigera, J., Winckler, M.: Capture and Evolution of Web Requirements Using WebSpec. ICWE 2010:173-188 (2010).
[27] Sardinha A., Chitchyan R., Weston N., Greenwood P., Awais Rashid: EA-Analyzer: Automating Conflict Detection in Aspect-Oriented Requirements. ASE 2009: 530-534, (2009).
[28] Silva, J.R., dos Santos, E.A., Applying Petri Nets to requirements validation. ABCM Symposium. Series in Mechatronics. Vol 1. pp. 508-517. (2004).
[29] Sommerville, I.: Software Engineering. Addisson Wesley (2002).
[30] Urbieta, M., Escalona, M.J., Robles-Luna, E., Rossi, G. Detecting Conflicts and Inconsistencies in Web Application Requirements. ICWE 2011 Workshops. Lecture Notes in Computer Science 7059, pp. 278–288 (2011).
[31] Valderas P., Pelechano V.: A Survey of Requirements Specification in Model-Driven Development of Web Applications. TWEB 5(2):10 (2011)
[32] Van Der Straeten, R., Mens, T., Simmonds, J., Jonckers, V.: Using Description Logic to Maintain Consistency between UML Models. UML 2003:326-340 (2003).
[33] WebSpec Language, http://code.google.com/p/webspec-language/. Accessed in April 2012.
[34] Yang, D., Wang, Q., Li, M., Yang, Y., Ye, K., Du, J.: A survey on software cost estimation in the chinese software industry. ESEM 2008:253-262 (2008).