Detecting Functional Requirements Inconsistencies within Multi-teams Projects Framed into a Model-based Web Methodology

J. A. García-García[1], M. Urbieta[2], J. G. Enríquez[1] and M. J. Escalona[1]
[1]*Web Engineering and Early Testing (IWT2) Research Group, University of Seville, Seville, Spain*
[2]*LIFIA, Facultad de Informática, UNLP, Buenos Aires, Argentina*

Keywords: Functional Requirements, NDT, Consistency, Ambiguity, Requirement Gathering, Distributed Teams.

Abstract: One of the most essential processes within the software project life cycle is the REP (Requirements Engineering Process) because it allows specifying the software product requirements. This specification should be as consistent as possible because it allows estimating in a suitable manner the effort required to obtain the final product. REP is complex in itself, but this complexity is greatly increased in big, distributed and heterogeneous projects with multiple analyst teams and high integration between functional modules. This paper presents an approach for the systematic conciliation of functional requirements in big projects dealing with a web model-based approach and how this approach may be implemented in the context of the NDT (Navigational Development Techniques): a web methodology. This paper also describes the empirical evaluation in the CALIPSOneo project by analyzing the improvements obtained with our approach.

1 INTRODUCTION

Requirements Engineering Process (REP) is the process of eliciting, understanding, specifying and validating customers' and users' requirements. The elicitation and specification of these requirements is the most critical tasks in requirements engineering (Robles et al., 2010) and it is a complex, an iterative and co-operative process as it is necessary to analyze and identify the functionality that the system has to fulfill in order to satisfy the users' and customers' needs.

In projects, being for the software or not, where analysts teams carry out the application's requirements eliciting, it can appear ambiguities or inconsistencies due to different points of view of the same business concept (Kotonya and Sommerville, 1996). These ambiguities and/or inconsistencies can cause variations in the project scheduling which can overestimate the required effort. These problems may be exacerbated in projects where: (i) high integration between its modules is required; and (ii) big teams of analysts are simultaneously working but in different modules. Consequently, it's necessary to carry out a validation and conciliation process which may be composed by analysis and consistency checking tasks between requirements in order to eliminate requirements ambiguity and contradictions. After performing this process in an iterative manner, the quality in the requirements specification can be improved which may imply an important reduction of development cost.

Traditionally, conciliation tasks are performed through meeting-based techniques and tools (De Lucia and Qusef, 2010). However, a high number of requirement inconsistencies are not usually detected on time (being this one of the most severe reason of project cost overrun (Yang et al., 2008)). In this context, the effort to correct the faults is several orders of magnitude higher than correcting requirements at the early stages (Leffingwell, 1997).

These problems may be increased when any methodology is used. In this sense, we propose to extend a highly used methodology in business and academia environments in order to define model-based systematic mechanisms to detect inconsistencies between requirements in early stages. This methodology is NDT (Navigational Development Techniques) (Escalona and Aragon, 2008) which is briefly described in Section 4.

This paper proposes a formal and systematic technique to detect inconsistencies in the catalog of requirements and a software tool with which it is possible to automate the early detection of these ambiguities and conciliate software requirements in MDE (Model-driven Engineering (Schmidt, 2006))

García-García, J., Urbieta, M., Enríquez, J. and Escalona, M.
Detecting Functional Requirements Inconsistencies within Multi-teams Projects Framed into a Model-based Web Methodology.
In *Proceedings of the 12th International Conference on Web Information Systems and Technologies (WEBIST 2016) - Volume 1*, pages 327-335
ISBN: 978-989-758-186-1

environments.

To achieve these objectives, we continue a previous paper (García-García et al., 2012) where we laid the foundations of our formal technical and its supporting tool which is named NDT-Merge. This tool detects conflicts from a lexical point of view. Now, this paper aims to propose a process in order to detect inconsistencies from a structural perspective based on models and later, it proposes solutions to solve these conflicts. In addition, this paper aims to implement this process within NDT-Merge.

Once defined the improvements mentioned on the NDT methodology for detecting conflicts, we have applied it on a real project called CALIPSOneo (Section 3). This project has been launched by Airbus Military and some work-packages were executed by the IWT2 Research Group.

Although the NDT methodology is purely focused on software projects, it was possible to successfully apply this methodology on CALIPSOneo where a methodological framework was needed during the capture of requirements and the analysis of the project as well as the testing phase.

The rest of this paper is structured as follows. Section 2 offers a global vision of NDT. Section 3 presents the problem which has been our catalyst to carry out this research. Section 4 presents some related work in requirements validation. Section 5 describes our previous work based on systematic conflicts and presents our model-based approach for detecting inconsistencies and dealing with them based on NDT by means of an illustrative example. In section 5, it is expliained the requirement quality impact in budget And, finally, Section 6 concludes by discussing the lessons learned, our main conclusions and some further work on this subject.

2 NDT

NDT (Navigational Development Techniques) (Escalona and Aragon, 2008) is a Model-Driven methodology that was initially defined to deal with requirements in Web development. NDT has evolved in the last years and it offers a complete MDE-based support for each phase of the software development lifecycle: the feasibility study, requirements, analysis, design, construction, implementation, as well as the maintenance and testing phases.

At present, NDT defines formally a set of metamodels for each phase of its lifecycle and it uses OCL (International Organization for Standardization, 2012) in order to define semantic constraints which ensure the definition of well-defined models. In addition, NDT defines transformation rules (using QVT (Object Management Group, 2008)) with which it is possible to generate models from others systematically. This implies a lower cost and a quality improvement for the software development.

However, this paper is mainly focused on the requirements phase. In this phase, NDT offers a set of techniques to capture, define and validate different kinds of requirements that are formally defined by a metamodel and they can be traced to the remaining artifacts of the lifecycle by managing them in a suitable manner. NDT uses information previously captured, defined and validated in the requirements phase as the basis for the analysis phase.

3 MOTIVATING SCENARIO: CALIPSOneo PROJECT

CALIPSOneo (advanCed Aeronautical soLutIons using Plm proceSses & tOols) (Escalona et al., 2013) has been developed in Airbus Military and where multiple teams have worked. From the experience of this project we know requirements are difficult to conciliate in projects involving multiple teams. This paper proposes improving the NDT methodology using in CALIPSOneo project to solve these problems during requirements' analysis phase.

In CALIPSOneo we have three interrelated subprojects (MARS, ELARA and PROTEUS) and independent teams (you can find more information in (Escalona et al., 2013)). However, subprojects have to be coordinated and they have to be correctly integrated because they have common actors who demand common functionality. Consequently, a flexible software methodology was needed to ensure the success of the CALIPSOneo project and the communication between different teams that were working on CALIPSOneo.

In this sense, NDT and NDT-Suite were adapted to work on this project where a collaborative and distributed environment was necessary taking into account the quality assurance during the specification and development of the project. In this context, NDT-Profile was adapted to provide a collaborative environment according to NDT; NDT-Quality was used to measure and ensure the quality and traceability between project results and NDT-Driver was used to automate the systematic generation of analysis models and testing models from the requirements phase.

For instance, in a particular way, NDT-Profile was adapted to enable use functional requirements of a specific subproject (e.g., MARS) in the

specification of other subproject (e.g., ELARA). In this case, use case diagrams were used to identify and organize functional requirements.

In PROTEUS, a hierarchical structure with two levels was defined. In the first level, use case diagram, four use cases were defined: Process definition in DPE (DELMIA Process Engineering), Process validation in DPM (DELMIA digital Process for Manufacturing), Process simulation in DPM and Product import. Each of these use cases were decomposed into a second level use case diagram. For instance, the Process validation in DPM contained verification of product, resources and process, and verification of lifecycles. Each second level use case has an activity diagram to define the flow of activities to be carried out by the main actor, in this case the manufacturing planner shows an extract, in NDT-Profile, of the activity diagram of the use case Verification of product, resources and process.

The activity diagram has three objectives. The first one is to define the sequence of tasks, as a step-by-step guide, for the main actor. Such sequence defines the working process. The second objective is to define in detail the lower level tasks to be carried out, allowing to identify where it is necessary to conduct an application development to assist the main actor or to automate a task. Once an application development is identified, a class diagram is created to specify the concepts to be implemented by such application. The third objective is to be used as basis for the definition of the diagrams for the testing phase of CALIPSOneo.

Considering these aspects and the collaborative nature of the CALIPSOneo project, we have considered necessary to research, propose and define formal technical with which is possible the early detection of ambiguities and inconsistencies when software requirements (specifically, functional requirements) are defined using use cases models and activity models. In addition, it is also necessary to conciliate different points of view of the same requirement once detection has been carried out.

4 RELATED WORK

Before developing our proposal to detect inconsistencies in functional requirements inconsistencies within multi-team projects, a Systematic Literature Review (SLR) has been carried out in order to know the state-of-the-art about this issue and take into account the proposals existing before tackling the indicated problem. This SLR is based on the protocol defined by (Kitchenham et al., 2007). Above, several works related to requirements validation are mentioned in this section.

Requirements Engineering (RE) is a coordinated effort to allow clients, users, and software engineers to jointly formulate assumptions, constraints, and goals about a software solution. However, one of the most challenging aspects of RE is the detection of inconsistencies between requirements in the requirements phase. Thus, this phase is considered the most critical tasks in RE (Robles et al., 2010). A global view presented in (Escalona and Koch, 2004) 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.

Focusing only on the detection of conflicts, over the last decade have been several proposals. Below, some of these proposals are described.

In (Brito et al., 2007), authors propose to detect concerned conflicts 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.

From a UML-based perspective, the conflict-detection process in other phases of the software life cycle has been deeply studied. In (Altmanninger, 2007), author proposes to detect conflicts in a twofold process: analyzing syntactic differences by raising candidate conflicts and understanding these differences from a semantic view.

In (Sardinha et al., 2009), authors 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.

Others authors propose to solve the detection of inconsistencies between requirements using knowledge-based techniques. In (Tuong Huan Nguyen et al., 2012), a knowledge-based requirements engineering tool, named REInDetector, is presented. This tool supports automatic detection of inconsistencies studying the semantic of requirements after capturing each requirement using its description logic language.

Moreover, UML has been widely studied for providing extensions and tools that allow modeling and developing high quality applications. In (Nugroho, 2008) it has been empirically analyzed the relation of level of detail of UML models and the resultant application's defect density. The outcome

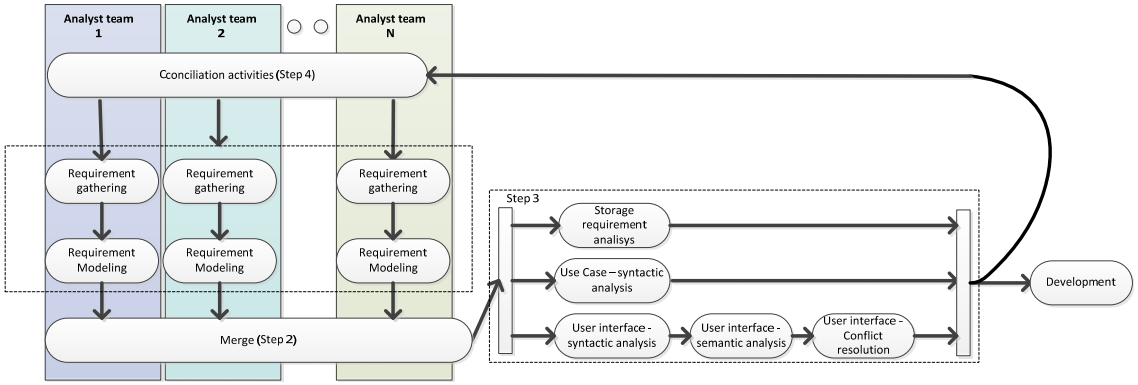

Figure 1: Distributed requirement gathering process.

showed that more detailed are models less defects report. Same authors provided a throughout empirical research about consequences of imperfect models. They point out, that although there are defects that are easily detected by developers, most of defects, such as Classes duplication, are hardly detected by them and therefore propagated in the solution. Duplicated element definitions in models such as Classes or Business Processes are named clones. Different technics has been addressed for detecting Clones in UML models (Störrle, 2010) or models repositories (Ekanayake et al., 2012).

5 OUR APPROACH FOR DETECTING CONFLICTS WITH NDT

Our approach to detect conflicts in the Software Requirement Specification (SRS) extends our previous proposal (Escalona et al., 2008). With this paper, we aim to extend our proposal introduced in (Escalona et al., 2008) taking into account the peculiarities of the functional requirements within distributed and heterogeneous projects where the NDT methodology is applied.

Due to large project complexity, our approach proposes a four steps process based on dividing and conquering by promoting different analyst teams that can focus in a specific subset of requirements. Each analyst team uses NDT approach for specifying application requirements.

Figure 1 shows an overall schema for our process and takes into account each NDT specification that comprises different models such as storage, functional and interaction requirements. The process is applied iteratively each time a new set of requirement rises (Harth and Koch, 2012). The new incoming set of requirements is checked with each of the already consolidated requirements of the system space.

Step 1. Requirement Gathering and Requirement Modelling. We propose to combine classical capture requirements techniques such as interviews or brainstorming 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 2. Requirement Merge. When analyst teams are specifying different functional modules with a high degree of interaction and integration among them, it is necessary to merge their works. This commitment is audited by a cross domain analyst who watches over the consistency of SRS. This kind of guardian is responsible keep SRS consistent and watches for the correct use of NDT guidelines. Each team's SRS is a perspective of the application meanwhile SRS is a stable view of the whole solution.

Step 3. Inconsistency Detection in Functional Requirements. When dealing with model oriented approaches like NDT, requirements are formalized using specific languages that provide facilities for describing system behavior. These models face same ambiguity issues when specifying requirements than traditional requirement gathering techniques does but have as advantage that reality has already been preprocessed by the analyst obtaining a simplified and clear problem to be solved.

Thus, once the "merge step" has been completed, a conciliation task start taking place where functional requirements are analyzed in order to look syntactic and structural inconsistencies. After that, if an issue is detected, it is reported to those analysts that their artifacts reported the issue.

As abovementioned, NDT specification is built of artifacts which compliant with diverse model schemas. Therefore, we had to define specific techniques for each kind of model for detecting ambiguities. Next we describe each technique.

1) **Storage Requirement Analysis.** Searching for syntactic conflicts aims to detect when two NDT artefacts are textually defining the same functional requirement. For this, we carried out the analysis of text using the technique described in (Salton and Buckley, 1988): «the vector space model technique». Both the use of this technique as its applications in NDT is described in (García-García et al., 2012). Thus, we are going to describe it briefly.

The identification of object' duplication depends on the analysis of the objects' description. Our paper has used a variation of «the vector space model technique» based on the statistic «term frequency-inverse document frequency technique». This technique associates a mathematical equivalence to any text, i.e., n-dimensional vector where n is the numbers of terms of the text. Each component stores the weight of each term. This weight of each word is calculated by the multiplication of two parameters: *tf * idf*.

On the one hand, tf indicates the frequency of the word in the text, i.e., the number of occurrences of the term in the text divided by the total number of terms in the text.

On the other hand, idf is the inverse document frequency, and it evaluates the importance of the considered term in the whole set of descriptions. Its definition allows giving a greater weight to the less frequent terms, which are considered as the most characteristic words. It is calculated by taking the logarithm of the quotient obtained by dividing the number of descriptions by the number of descriptions that contains the term. Then, the mathematical expression of idf is presented.

$$idf(t,D) = \log(|D| / (1+|\{d \in D : t \in d\}|)) \quad (1)$$

With:

- |D|. D is the corpus or set of descriptions analyzed and |D| is the number of descriptions in the corpus.
- (1 + | {d ∈ D : t ∈ d} | This mathematical expression represents the number of descriptions in which the term t appears. The number of descriptions where the term t appears. This expression avoids a division-by-zero in the case in which the term would be absent.

Finally, the mathematical expression of *tf * idf* is presented:

$$tf * idf(t,d,D) = tf(t,d) * idf(t,D) \quad (2)$$

The similarity of the descriptions is evaluated considering that descriptions are vectors of words. Since we consider vectors, we have to apply a single order of words. All the words of the whole set of descriptions have to be considered and each new one is a new dimension in the vector. Then, the description's original order is not relevant, it is only necessary to have all the words.

After building the two vectors, we can know what is the similarity between the two descriptions. For this, we apply the cosine to calculate of the angle between two vectors. The cosine with value 1 implies that the angle between the vectors is 0, which implies that the texts are similar.

$$cos(\alpha) = V1.V2 / (||V1||.||V2||), \quad (0 \leq cosine \leq 1) \quad (3)$$

To apply the technique described, first of all, the words are stemmed to their roots so that plurals, verbal forms or other forms are not considered. We also don't consider pronouns, articles and other connection terms. Then, the cosine similarity is applied; the algorithm calculates cosines between two vectors. Therefore, we understand that all the relevant words of the corpus have to be represented in the vectors.

The algorithm returns a number for similarity ratio between 0 and 1 when comparing two texts. Zero means completely different texts and numbers near to one mean similar texts. When comparing texts in Spanish, using the Spanish stemmer, the algorithm returned lower values for unrelated texts and higher numbers for similar texts.

In order to find similar objects, we had to compare each requirement to each other, and then filter the results that returned the higher values. We defined filter threshold 0.5 to filter those results above such number.

In requirement gathering process we could detect two related definitions by analyzing storage requirements. Next we present both requirements that describe the same system need but they were defined differently. Analysts were warned by our tool because its similarity ratio 0.62856 was above defined threshold.

2) **Use Case Analysis.** Searching for use case conflict aims at detecting when two NDT functional requirements are defining the same functionality of the system in a different manner. For this, we are going to take advantage of this situation for introducing automatic analysis of modeled requirements using well-known graph theory. NDT uses UML Activity diagram as tool

for describing use cases in such a way models can be analyzed as graph using algorithms developed in graph similarity or graph isomorphism research line.

In order to avoid model duplications and inconsistencies, we profit from developments performed in UML field by (Störrle, 2010); (Kelter et al., 2005).

Our analysis tool takes each FR as UML Activity diagram modeled and stored in an Enterprise Architect document and builds an equivalent graph that represents each element of Activity Diagram but adds extra information that is not present in Enterprise Architect´s diagram. For instance, Activities in EA´s models are not related with swimlanes. Elements are placed over swimlanes in such a way they overlap but a relationship between them is not defined although it is perceivable by a designer. The translation is straightforward because object oriented models are by definition graph were objects are Vertex and relationships are Edges. In figure 2, we show two simple activity diagrams, their corresponding graph and how their differences are detected. The lower activity diagram has a start state, a simple states and a final state. Its graph representation shows five nodes. On the other hand, the upper activity diagram is a little bit more complex and has a start state, two linked states and a final state. Its graph representation has seven nodes; two additional nodes highlighted with grey. These last two nodes represent the difference between two activity diagrams.

Once activity diagrams have been derived to a navigable graph, firstly the tool takes two graphs and compares them looking for equality in graph definition (same vertexes and edges) and inclusion. By means of identifying equality, we can improve estimation of budget because a given requirement is not computed twice. By detecting inclusion, it allows defining reusable concepts that will simplify development and maintenance tasks.

In order to detect differences between models, we used well-known graph algorithms for isomorphism and equivalence analysis. Our tool was built on top of JGraphT library which provides a framework for graph computation.

We identified a couple common problems when modelling that can be identified using graph manipulation. Next we list two supported inconsistencies identification with a simplified graph operation.

- **Similarity on Elements Definition.** By means of comparing elements in model specification, it is defined a similarity ratio based on amount if different element over total graph elements.

Ratio values are in a range from 0 to 1 where 0 stand for totally different models and values closer to 1 means similar models.

$$ratio(ad,ad2) = (((graph_{ad1} \cup graph_{ad2}) - (\Delta(graph_{ad1}, graph_{ad2})))) / (graph_{ad1} \cup graph_{ad2}) \quad (4)$$

- **Duplications.** Occurrences of models duplications within other elements. This analysis is quite straightforward because, after removing redundant element such as initial and final state, it is checked whether a model is included within others.

$$1 < graph_{ad1} \cap (graph_{ad2} - \{initial\ state, final\ state\}) \quad (5)$$

- When the intersection result is not empty, it means that both diagrams share few elements definition.

3) **User Interface Models.** We propose a twofold process for analyzing NDT user interfaces: a syntactic step that compares each model in order to detect differences and a second step called semantic analysis that compares two models that show to be similar called conflict in order to evaluate if they are equivalent semantically.

A candidate conflict arises when the set of syntactic differences among requirements appear as a consequence of: (i) the absence of an element in one user interface 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.

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 (Altmanninger, 2007) and (Li and Ling, 2004) where models are analyzed in order to expose their underlying goals. When the underlying goals 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 (Altmanninger, 2007) which focuses on having an additional seman-

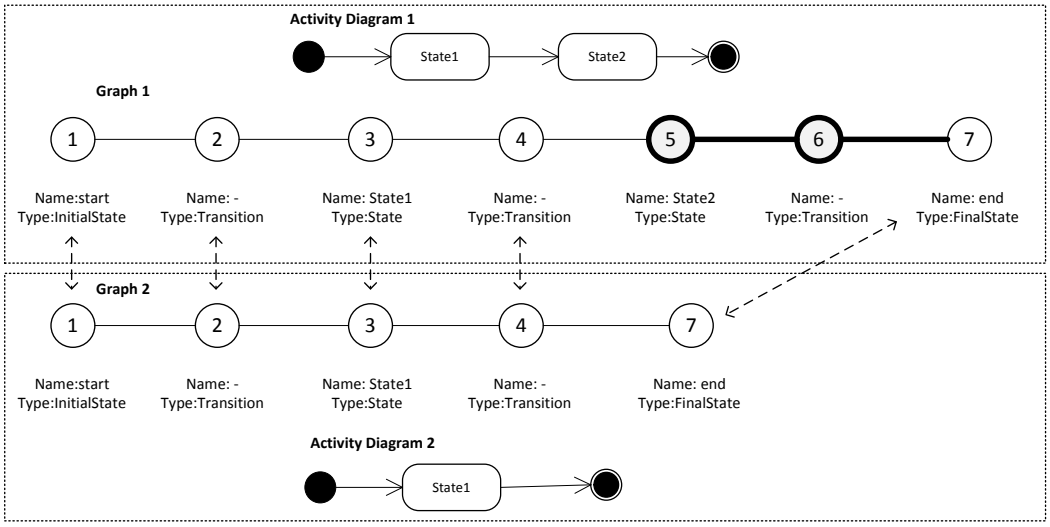

Figure 2: Graph representation of activity diagrams.

tic 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.

Step 4. 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.

6 IMPACT IN BUDGET

CALIPSOneo has already finished and we cannot replicate the whole process. However, we could do our experiment to measure the grade of efficiency and effectiveness reached when applying it.

This project was managed in a particular manner. In each subproject (MARS, PROTEUS and ELARA), meetings were held weekly during which, work teams discussed about possible integration problems when they were defining a catalog of requirements.

In this mechanism of reviewing two main problems were detected:

- Inconsistences were “discovered”, without any special mechanism or technique and their detection depended on the experience of the team.
- When an inconsistence was detected, the way to solve it was to discuss between teams. Depending of the kind of inconsistence, this discussion was in a local team (MARS, PROTEUS or ELARA) or even, if the inconsistence affected several subprojects, it provoked global meetings involving several teams.

Apart of “the luck” in the detection of inconsistences, the execution of the second point affects directly to the cost of the project. In this meetings, mainly if they affect to the three teams, resulted too expensive.

In these meetings analyst for each member, project leader of the affected subprojects and functional users had to participate and discuss about different solutions.

As no systematic mechanism were detected. Each subproject teams have weekly meeting to review the evolution of the requirements and monthly a global meeting was celebrated. Besides, the quality team of the project participated in each meeting. The cost of these meetings was too high and it could be reduced using approaches like proposed in this paper.

7 IMPACT IN BUDGET

In a software project, one of the most relevant phases

in the lifecycle is the requirements phase, which conditions the development through all the aspects of the project, mainly economic. With the increase of complexity of applications with big, distributed and heterogeneous projects, this phase acquires a more relevant role because often, these systems are specified by multiple analyst teams and in this context, it is necessary to perform an effective conciliation of requirements.

When there are different set of requirements, they have been merged in order to obtain conciliated requirements to initiate the system development. However, this task frequently depends on the analyst's experience or is done manually. Thus, it is necessary to establish formal mechanisms to combine different requirement specifications and detect conflicts among these requirements.

This paper extends a previous paper in which we had presented the application of a general MDE approach for the systematic detection of requirements inconsistencies and how that approach was extended to improve the NDT methodology.

Our proposal is based in techniques for detecting similarities between graphs and techniques for the detection of syntactic conflicting in a textual manner. This paper illustrates the application of our proposal on the CALIPSOneo project that originally was conciliated by hand without the use of any mechanism to check it.

ACKNOWLEDGEMENTS

This research has been supported by the MeGUS project (TIN2013-46928-C3-3-R), by the SoftPLM Network (TIN2015-71938-REDT) of the Spanish Ministry of Economy and Competitiveness, and CALIPSOneo Project.

REFERENCES

International Organization for Standardization, 2012. ISO/IEC. ISO/IEC 19507:2012 Object Management Group Object Constraint Language (OCL). 2012.

Kitchenham, B., Charters, S. Guidelines for performing Systematic Literature Reviews in Software Engineering. Version 2.3. Department of Computer Science, University of Durham. *EBSE-2007-01. 2007*.

Kotonya, G.; Sommerville, I.: Requirements engineering with viewpoints. *Software Engineering Journal*, vol.11, no.1, pp.5-18 (1996).

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).

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).

Leffingwell, D.: Calculating the Return on Investment from More Effective Requirements Management. *American Programmer*, 1997, VOL 10; NUMBER 4, pages 13-16 (1997).

Escalona, M. J., Garcia-Garcia, J. A., Mas, F., Oliva, M., Del Valle, C. Applying model-driven paradigm: CALIPSOneo experience. *Conference on Advanced Information Systems Engineering 2013 (CAiSE'13)*, vol. 1017, pp 25-32. 2013.

Robles, E., Garrigós, I., Grigera, J., Winckler, M.: Capture and Evolution of Web Requirements Using WebSpec. *ICWE 2010*:173-188 (2010).

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

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

Escalona, M. J., Urbieta, M., Rossi, G., Garcia-Garcia, J., Luna, E.. Detecting Web requirements conflicts and inconsistencies under a model-based perspective. *Journal of Systems and Software*, 86(12), 2013.

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).

Altmanninger, K.: Models in Conflict - Towards a Semantically Enhanced Version Control System for Models. *MoDELS Workshops 2007*:293-304 (2007).

Sardinha A., Chitchyan R., Weston N., Greenwood P., Awais Rashid: EA-Analyzer: Automating Conflict Detection in Aspect-Oriented Requirements. *ASE 2009*: 530-534, (2009).

Tuong Huan Nguyen, Bao Quoc Vo, Markus Lumpe, and John Grundy. 2012. REInDetector: a framework for knowledge-based requirements engineering. DOI=10.1145/2351676.2351754. 2012.

García-García J. A., Escalona M. J., Ravel E., Rossi G., Urbieta M. NDT-merge: a future tool for conciliating software requirements in MDE environments. *iiWAS 2012:177-186 (2012)*. ISBN/ISSN: 978-1-4503-1306-3. Bali, Indonesia. 2012b.

Salton G., Buckley C. 1988. Term-Weighting approaches in automatic text retrieval. Department of Computer Science, Cornell University, Ithaca, NY 14853, USA.

A. Nugroho, B. Flaton, and M. R. Chaudron. Empirical Analysis of the Relation between Level of Detail in UML Models and Defect Density. In *Proceedings of the 11th int. conference on Model Driven Engineering Languages and Systems (MoDELS '08)*.

C. C. Ekanayake, M. Dumas, L. García-Bañuelos, M. La Rosa, and A. H. M. ter Hofstede. 2012. Approximate clone detection in repositories of business process models. In *Proceedings of the 10th int. conference on Business Process Management (BPM'12)*.

Li, C., Ling, T. W.: OWL-Based Semantic Conflicts Detection and Resolution for Data Interoperability. *ER (Workshops) 2004*:266-277 (2004).

Harth, A., & Koch, N. (2012). Current Trends in Web Engineering: Workshops, Doctoral Symposium, and Tutorials, Held at ICWE 2011, Paphos, Cyprus, June 20-21, 2011. Revised Selected Papers. Springer Science & Business Media.

Schmidt, D. C., 2006. Model-Driven Engineering. Published by the *IEEE Computer Society vol 39*.

Harald Störrle. 2010. Towards clone detection in UML domain models. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume (ECSA '10)*, Carlos E. Cuesta (Ed.). ACM, New York, NY, USA, 285-293.

Kelter, U., Wehren, J. & Niere, J. (2005). A Generic Difference Algorithm for UML Models. In P. Liggesmeyer, K. Pohl & M. Goedicke (eds.), *Software Engineering* (p./pp. 105-116).