MDE-Oriented Process to Manage Requirement Inconsistencies

Hatime Bencharqui, Adil ANWAR

Abstract: ***Requirements inconsistencies could arise for multiple reason and at all levels of hierarchy. in context of a complex system detecting inconsistencies among requirements is important and needs a rigorous practice. It is all the more crucial task when they are specified in informal language. Thus, managing inconsistencies is indispensable to success any system design. Constraint specification language (CSP) formalism is used to define rules to spot conflicted requirements. In this paper, we present a process founded on ReqDL language in order to manage requirements inconsistencies at all level of hierarchy. we describe an approach to support inconsistencies identification and resolution among requirement.***

Index Terms: requirement engineering, Sysml, DSL, ReqDL , Complex System

I. INTRODUCTION

Requirements engineering (RE) is subset of systems engineering consists of specification, documentation, validation and management requirements of stakeholders [22]. In context of the complexity and size of System we face today, these activities become more challenging. Smart Grid and AAL are examples of complex system whose requirements are difficult to manage. This is even harder in the context of complex system which may interact with each other. MBSE is one of the most efficient and useful approach that is relied on to develop systems and that is regarded as a primary communicative tool while developing especially complex systems. It is based on models to cover several aspects such as specifying requirements, designing systems and developing and verifying any kind of elements be it hardware or software. MBSE opted for System modeling language (SysML) as the de-facto language for modelling projects. It is a choice that was made on the basis of the number of the pragmatic diagrams that SysML provides and that make it possible to overcome systems engineering problems. Sysml is a general-purpose graphical modeling language for complex systems designs that may include hardware, software [7]. Concerning the SysML requirements diagram, it is the one used to deal with requirements specification and relationships. However, its informal text-based requirement presents certain drawbacks which supposes that there would be some difficulties to spot the stakeholder requirements consistencies while developing those complex systems.

Revised Manuscript Received on October 10, 2019

Mr. H Bencharqui, Department of Computer Science and Engineering, Mohammadia Engineering School, Rabat, Morocco.

Mr. A. ANWAR PHD Teacher, Computer Science at Mohammadia School of engineering in Rabat..

It is a drawback that is mostly related to the lack of defining rules or constraints for writing requirements text and coming up with rigorous methods for detecting inconsistencies using requirement text property.

Many rigorous modeling MBSE tools that support sysml and adopted by the Object Management Group (OMG) (e g MagicDrawn, Rational Rhapsody, Visual Paradigm, Sparx Enterprise Architect…). Other solutions are solely dedicated to requirements management; They offer mechanisms such structuring, elucidating and tracing requirements artifacts. In spate of that, manage inconsistencies among requirements is still one of the difficult tasks in requirement engineering life cycle. Most produced requirements documents and models do not follow accurately the guidelines and common criteria of writing requirements [1]-[4].

During the requirement engineering process, inconsistencies may arise at different abstractions level. One requirement is inconsistent if it is in conflict with one or more other requirements. Therefore, they cannot be met in parallel by the system being developed. Decision should be taken on which detected requirements will be removed or modified. Inconsistency in RE deserves to be considered, the main reasons causing failure of a large number of projects are requirements related issues, including inconsistencies. Furthermore, according to the Standish Groupe research reports[23] , requirements inconsistencies concerning system requirements are considered as a crucial research area in RE. In this paper we will present requirements inconsistencies process. This process offers a new approach to reduce, detect and resolve inconsistencies in requirements models. This approach relies on ReqDL as language in which requirements model are written. Our process contributes at different level of hierarchy to detect and resolve inconsistencies.

II. BACKGROUND

A. Overview

This section is devoted to give an overview of the main concepts of the ReqDL language, leveraging previous works [11][24] that presented the basic semantics and concepts of the language. ReqDL has a form of a structured natural language, is designed to allow requirements engineers to explicitly write all kind of requirements. ReqDL includes attributes and connectors to capture and control how requirements are written.

The way ReqDL is designed allows it to identify traceable requirement, and also obtain some traceable information that might constitute a part of requirements statements like point of view , source, and author.

REQDL is a language which is designed to aid requirement elicitation with

a friendly-user concrete syntax. ReqDL possesses a range of elements to get the various sorts of data. which involves the traceability of the requirements and others information are about the requirement text description of the requirements itself. we focused here on the requirement description property. The requirement description in ReqDL is composed of three parts:

- Modal verbs: need shall, should, must and will refers to the requirement's feature which might be either the obligatory, urgent or recommended in the future and also locate the hierarchy levels , namely system(subsystem), component and stakeholder requirements.
- Requirement action: so as to feature the system behavior, ReqDL semantic distinguish three kind of requirement actions: Autonomous action, interactive action and passive action.
- Conditions: ReqDL description make use of a couple of operators "*when*" and "*where*". the former specifies temporal condition whereas the latter expresses the logical condition. Hereafter an example of specification of a requirement with the REQDL language.

```
Model CCSRequirements {
ModelElement Block Thermostat ;
Traceable Client Requirement QuickTemp {
identifier:01
type:Functional
description :CCS should be able to "allow the engine to reach optimal temperature "
};
Traceable System Requirement EngineCirculation belongsTo QuickTemp {
identifier:02
type:Functional
description :CCS should "restrict coolant flow to circulate through radiator"
using Thermostat
};
}
```

III. MANAGEMENT REQUIREMENT WITH REQDL LANGUAGE

The permitted structure on which the requirement model should follows is expressed as meta-model. The meta model describes and explains all relations and syntax of the model. It should be made in accordance with the suggested meta-model describing the abstract of the language. This must be made according to the proposed meta-model which describes the Abstract Syntax of the language. We have added some attributes to ReqDL language to define requirement description with more granularity. Fig.8 illustrates the main part of our Metamodel. Listing 1 illustrates the ReqDL description grammar that matches to the metamodel. This grammar concerns the description structure (e g. connectors, modal verbs . type of actions …) wich is implemented using xtext framework . xtext is an open source tool for developing domain specific language [12].

In order to provide some relevant aspects of a specification in our metamodel, we enrich the model with additional constraints in ReqDL metamodel using object constraint language (OCL). OCL is a formal language that is dedicated to describe unambiguous constraints in conjunction with Ecore implementations [19]. With a textual metamodel view, we could add new constraints directly at metamodel level. For instance, in one requirement description constrains each attribute to have a unique name and do not contain duplicate

Req.description a: The CCS shall transport the coolant while reaching operating temperature
Req.description a: The CCS shall transport the coolant Before reaching operating temperature

attributes in the same description.

IV. MODELING PROCESS FOR MANAGING REQUIREMENT INCONSISTENCIES

A set of requirements is consistent if their specifications do not contain any ambiguity and they are not in conflict with each other. Therefore, Inconsistencies arise when a set of requirements have similar template, but differ on few requirement attributes values or on connectors terms. hereinafter a simple example of a conflicting requirement relied on car cooling system.

```
1  Description returns Description:
2     ('where' condition=[imported::Parameters]
3  tirggerterms= Tirggerterms cond = condreq)?
4  subjectName = [imported::Acotr]
5  verb = ModalVerb
6  action = ActionLog
7     ('using' modelElement = ModelElement)?;
8     condreq returns condreq :
9     condreq =[imported::Constraint]|value=Value ;
10    Value: "value" name= INT ;
11 enum Tirggerterms: lower='is lower than' |upper='is upper than'
   |equal='is equal' |in='in';
12 enum ModalVerb: will='will'|
   shall='shall'|should='should'|should='must';
13 tempcon: lower='before' |upper='after' |
   assoonas ='as soon as' | while ='while' | under ='under';
14 ActionLog : AUTONOMOUSACTION |
15 INTERACTIVEACTION | PASSIVEACTION;
16 AUTONOMOUSACTION returns AUTONOMOUSACTION :
      actionName=[imported::Action]
17    (tempcon= tempcon event = [imported::Event])? ;
18 INTERACTIVEACTION : 'provide'
   username=ID 'with the ability to' actionName=[imported::Action] ;
19 PASSIVEACTION returns PASSIVEACTION :'be able to '
20    actionName=[imported::Action];
```

Listing 1: ReqDL description Grammar

The Fig.1 presents a modeling process for managing requirement inconsistencies. This process concerns all kind of requirements at every stage of system development. The process requires two main roles; requirement engineer and the system tool. requirement engineer has an important task in analyzing requirement at particular level development process. He should be able to take a good decision and evaluate their impact. The system should detect inconsistencies among requirement, notify requirement engineer with a list of resolution rules.

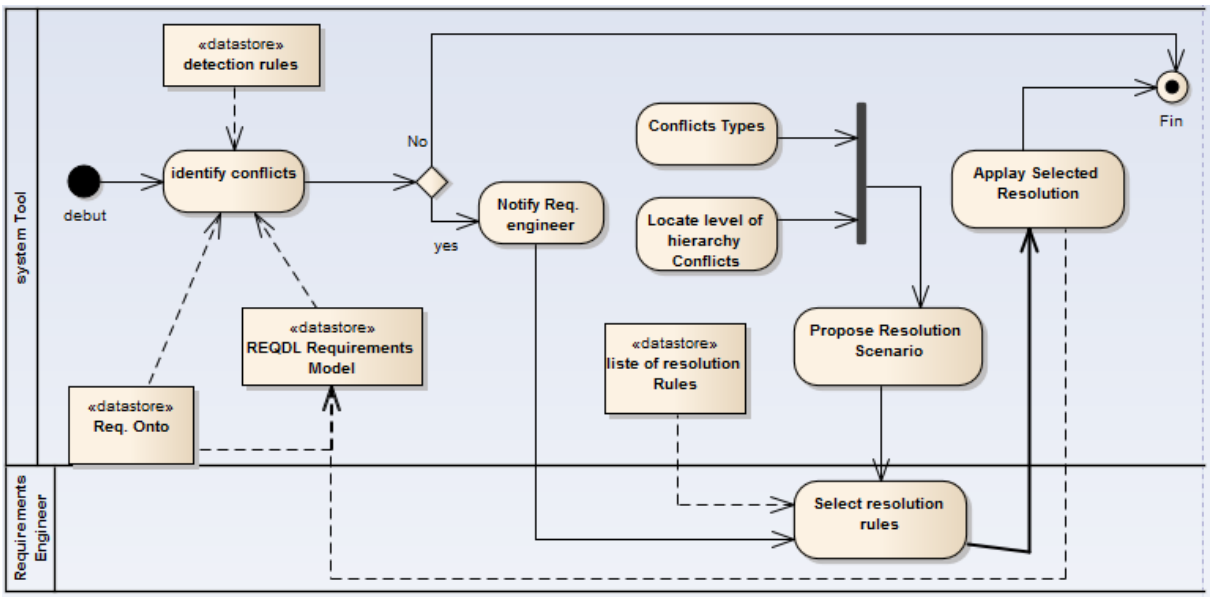

Fig. 1. Modelling Process for managing inconsistencies

To identify the inconsistencies among requirements, it is important to capture properties and concepts common among requirements. This is an indispensable activity throughout inconsistency management; for the reason that requirements without common pieces cannot be evaluated as inconsistent requirements. Therefore, identify inconsistencies consist of detection, which requirement properties are on conflict with other requirements properties. This is based on the use of attributes values or connectors terms to detect inconsistency.

Requirement ontology model: it is important to use a formal description of the ontology to guide the requirement engineer to elicited a requirement description. In terms of usability and maintainability the ontology is very advantageous because it contains all shared and agreed system terminology by all actors involved in the system development life cycle.

In order to be precise and unambiguous, ontology model should be the first step in the system development cycle. it captures the knowledge related to a specific domain that would reinforce terms and concepts. The following, some defined concepts:

- **Event**: Event that system intends to achieve
- **Object**: Object are used especially in the system requirement to specify elements that are subject to a function.
- **Function**: to express functions performed by the system
- **Interface**: define the external space that interact with the system boundaries.
- **Operational environment**: it is important to define some environmental aspects that the system should take into consideration to operate in.
- **Constraints**: define constraints that limit solution space
- **Temporal connectors**: before, after, while and when.

The second step is ReqDL model which uses the ontology model to define the requirement properties this consist of importing all necessary elements exist in the ontology model.

Level hierarchy: since the conflicts could arise any requirements engineering step. It is necessary to detect on which hierarchy level the inconsistency is spotted. Therefore, we associate a "modal verbs" at each different levels of system hierarchy namely: stakeholder requirement, system (subsystem) requirement or component requirement.

- Duplicate requirement description: all requirement that have exactly the same properties
- Performance capability: Locate all requirements that differ only in performance capability such as capacity of system to perform requirement within a specified range unit.
- Incompatible event: consist in finding a pair of requirements where share all elements but differ at event property
- Incompatible temporal condition: The fact that two different temporal condition are used can be identified as potential inconsistency before <event> vs after <event>.
- Functional inconsistency
- Environmental inconsistency
- Performance/time conflict

A. Detection rules

Through constraints satisfaction problem (CSP), the inconsistent requirement can be identified automatically. To manage the inconsistent requirements, we use Detection/Resolution rules.

With CSP formalism, the consistency check among requirement is realized with rules. We opted for a formal presentation of those rules. Fig.2 shows the formal description of a detection rule. The specification of this rule, which shows consistency in requirement attributes, contains conditions. Fig.3 shows another example of rules in requirement Connectors.

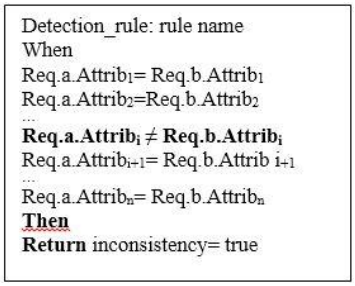

Fig. 2. Formal description rule

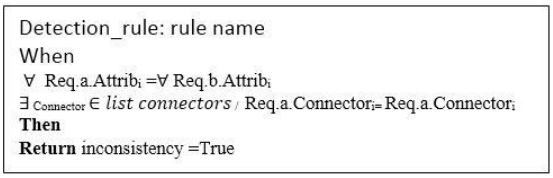

Fig. 3. Requirements connectors Rule

V. RELATED WORKS

Our work is motivated by the need to improve requirement description in order to describe it in a finer way. We provide a process that allow to detect inconsistencies among them.

Managing inconsistencies in RE activity still an issue for many researchers which can be categorized into two groups.one is interaction among requirements on software system and another concern requirement text description for detecting inconsistencies.

in the remainder of this section, we are discussing the main research works relied on approaches to tackle the requirements inconsistencies in "RE" area.

in the remainder of this section, we discuss the key related research approaches that have been proposed for dealing with requirements inconsistencies in RE.

- **Requirements Interaction**

Many previous works have been interested on the conflicting requirements management in the case of software systems. Those approaches focused mainly on interactions between functional and unfunctional requirement and consider this method as efficient to deal with inconsistency [16][17][18][19][20]. [21] proposed a method founded on quality attributes such as performance and maintainability and traceability techniques to identify conflicting requirements. Based on use case, Hausmann et al.[ref] introduce an approach to detect inconsistencies between functional requirements. Pre-condition and post-condition rules are used to express functional requirements. The authors proposed an approach for detecting dependencies that represent inconsistency to be decided by the modeler. Unlike our approach, which based on the capturing concepts into text property and afterward, searching similarity that might exist in the other requirements elements.

- **Requirement text description**

Few works focus on the management of requirement inconsistencies based on requirement statement. In [14] the authors presented a framework which permit identifying the conflicts among requirements in the case of system of systems. This method has been applied to resource-based consumption. To deal with this issue, the authors extend the Relax language [15] to express the requirements in case of the self-adaptive complex system. The author in [13] has presented a method for constructing a collection of inconsistent requirements and their categorization. This is relied on linguistic basis to mine the incoherencies between requirements. Nevertheless, it is limited on requirements in technical specification, doesn't concern different levels of abstraction in RE activity.

Finally, our approach and our requirements specification with Reqdl language allows requirements to be formalized in an accurate and a simple syntax, making it easy for diferent actors involved in the project to understand and communicate between each other.

VI. CONCLUSION

in view of the fact that systems are more growing and advanced, requirement activity requires new approaches to manage à large set of requirements along system development process.

In this paper, we have presented a new process to tackle requirements inconsistencies. Our contribution consists of the use in addition to the language ReqDL, an ontology which allow to specify all terminology shared by all protagonists of system. Thus, it would lead to capture the elements and attributes of requirement that are inconsistent with others in another requirement. In addition, the use of those elements and operators allows the identification of inconsistency type and level of the hierarchy in which the conflict is spotted. For next works, we have aimed to implement and validate our approach and to design a solution tool that help to check and tackle the issue of requirements consistency model.

REFERENCES

1. Gottschalk, M., Uslar, M., & Delfs, C. (2017). The Use Case and Smart Grid Architecture Model Approach: The IEC 62559-2 Use Case Template and the SGAM Applied in Various Domains. Springer.
2. Sage, A. P., & Rouse, W. B. (2014). Handbook of systems engineering and management. John Wiley & Sons
3. OMG System Modeling Language Specification Version 1.5", [online] Available: http://www.omg.org/spec/SysML/1.5/
4. INCOSE. 2017. Guide for Writing Requirements.
5. Wyner, A., Angelov, K., Barzdins, G., Damljanovic, D., Davis, B., Fuchs, N., ... & Luts, M. (2009, June). On controlled natural languages: Properties and prospects. In International Workshop on Controlled Natural Language (pp. 281-289). Springer, Berlin, Heidelberg..
6. Dick, J., Hull, E., & Jackson, K. (2017). Requirements engineering. Springer.
7. OMG System Modeling Language Specification Version 1.5", [online] Available: http://www.omg.org/spec/SysML/1.5/
8. Nguyen, T. H., Vo, B. Q., Lumpe, M., & Grundy, J. (2014). KBRE: a framework for knowledge-based requirements engineering. Software Quality Journal, 22(1), 87-119.
9. Doe, R. (2009). The standish group chaos report.
10. Cabot, J., & Gogolla, M. (2012, June). Object constraint language (OCL): a definitive guide. In *International School on Formal Methods for the Design of Computer, Communication and Software Systems* (pp. 58-90). Springer, Berlin, Heidelberg.
11. Haidrar, S., Bencharqui, H., Anwar, A., Bruel, J. M., & Roudies, O. (2017, September). REQDL: A requirements description language to support requirements traces generation. In *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)* (pp. 26-35). IEEE.

12. Bettini, L. (2016). *Implementing domain-specific languages with Xtext and Xtend*. Packt Publishing Ltd.
13. P. Saint-Dizier, "Mining Incoherent Requirements in Technical Specifications," 2017, pp. 71–83.
14. T. Viana, A. Zisman, and A. K. Bandara, "Identifying Conflicting Requirements in Systems of Systems," in 2017 IEEE 25th International Requirements Engineering Conference (RE), 2017, pp. 436–441.
15. J. Whittle, P. Sawyer, N. Bencomo, B. H. C. Cheng, and J.-M. Bruel, "RELAX: a language to address uncertainty in self-adaptive systems requirement," Requir. Eng., vol. 15, no. 2, pp. 177–196, Jun. 2010.
16. Hausmann, J. H., Heckel, R., & Taentzer, G. (2002, May). Detection of conflicting functional requirements in a use case-driven approach. In Proceedings of the 24th International Conference on Software Engineering. ICSE 2002 (pp. 105-115). IEEE.
17. "Heisel, M., Souquieres, J.: A heuristic algorithm to detect feature interactions in requirements. In: Language Constructs for Describing Features, pp. 143–162. Springer (2000)."
18. "Kim, M., Park, S., Sugumaran, V., Yang, H.: Managing requirements conflicts in software product lines: A goal and scenario based approach. Data Knowl. Eng. 61, 417–432 (2007)."
19. "Moore, C.W. 2003. The mediation process: Practical strategies for resolving conflict, 3rd ed. San Francisco, CA: Jossey-Bass."
20. "Robinson, W. N. (2004). Surfacing Requirements Interactions. In Perspectives on Software Requirements (pp. 69-90). Springer, Boston, MA."
21. "Egyed, A., Grunbacher, P.: Identifying requirements conflicts and cooperation: How quality attributes and automated traceability can help. IEEE Softw. 21(6), 50–58 (2004)."
22. Dick, J., Hull, E., & Jackson, K. (2017). Requirements engineering. Springer.
23. Standish Group, 2009. The CHAOS Report. http://www.standishgroup.com/
24. Bencharqui, H., Haidrar, S., & Anwar, A. (2019, April). Dealing with Requirement Inconsistencies Based on ReqDL Language. In 2019 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS) (pp. 1-6). IEEE.

AUTHORS PROFILE

Mr. H BENCHARQUI is PhD student in Department of Computer Science and Engineering, Mohammadia Engineering School, Rabat, Morocco. His area of interest is system engineering, Model Driven engineering, Requirement engineering and Sysml.

Mr. A. ANWAR is PHD Teacher in Computer Science at Mohammadia School of engineering in Rabat..his area of interest is system engineering, Model Driven engineering, Requirement engineering
.