International Journal of Approximate Reasoning 54 (2013) 109–131

Contents lists available at SciVerse ScienceDirect

International Journal of Approximate Reasoning

journal homepage: www.elsevier.com/locate/ijar

From inconsistency handling to non-canonical requirements management: A logical perspective

Kedian Mu [a,*], Jun Hong [b], Zhi Jin [c], Weiru Liu [b]

[a] School of Mathematical Sciences, Peking University, Beijing 100871, PR China
[b] School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, BT7 1NN, UK
[c] Key Laboratory of High Confidence Software Technologies (Peking University), Ministry of Education, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, PR China

ARTICLE INFO

Article history:
Received 21 July 2011
Revised 2 May 2012
Accepted 25 July 2012
Available online 1 August 2012

Keywords:
Inconsistency handling
Non-canonical requirements
Redundancy
Incompleteness
Requirements engineering

ABSTRACT

As a class of defects in software requirements specification, inconsistency has been widely studied in both requirements engineering and software engineering. It has been increasingly recognized that maintaining consistency alone often results in some other types of non-canonical requirements, including incompleteness of a requirements specification, vague requirements statements, and redundant requirements statements. It is therefore desirable for inconsistency handling to take into account the related non-canonical requirements in requirements engineering. To address this issue, we propose an intuitive generalization of logical techniques for handling inconsistency to those that are suitable for managing non-canonical requirements, which deals with incompleteness and redundancy, in addition to inconsistency. We first argue that measuring non-canonical requirements plays a crucial role in handling them effectively. We then present a measure-driven logic framework for managing non-canonical requirements. The framework consists of five main parts, identifying non-canonical requirements, measuring them, generating candidate proposals for handling them, choosing commonly acceptable proposals, and revising them according to the chosen proposals. This generalization can be considered as an attempt to handle non-canonical requirements along with logic-based inconsistency handling in requirements engineering.

1. Introduction

A good-quality software requirements specification is crucial for the success of a software development project. However, it can often be difficult to develop such a requirements specification. Problems associated with requirements are still potentially the major causes of software project failures, and there is a great need to facilitate software development process [1].

Inconsistency has been considered as a main class of defects in requirements specifications [1]. A number of techniques have been proposed to handle inconsistency in requirements engineering [2–12]. In particular, it has increasingly been recognized that it is effective to use logics to formulate management of inconsistent requirements specifications [1]. Various logic-based approaches to handling inconsistency in requirements specifications have recently been proposed [2,4,1,9–12]. Most of these approaches focus on how to apply non-classical reasoning techniques, such as paraconsistent reasoning and non-monotonic reasoning, to detecting and analyzing inconsistency in requirements specifications. For example, Hunter and Nuseibeh [2] developed labeled quasi-classic logic to represent and reason about requirements specifications in the presence of inconsistency. Gervasi and Zowghi [1] proposed methods for reasoning about inconsistency in natural language

* Corresponding author.
E-mail address: mukedian@math.pku.edu.cn

0888-613X/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ijar.2012.07.006

requirements by combining natural language parsing techniques and non-monotonic reasoning. Easterbrook and Chechik [4] presented a framework termed χbel for merging inconsistent viewpoints using multi-valued logics.

However, inconsistency is not an isolated problem in requirements engineering. The process of maintaining consistency often results in some other undesirable types of information about requirements. For example, Zowghi and Gervasi [12] argued that there is an important causal relationship between consistency, completeness and correctness of requirements in requirements evolution. Recently, Martinez et al. [11] took into account the interplay between inconsistency and incompleteness in merging multi-perspective requirements. Previously, we termed requirements that are incomplete, redundant, vague or inconsistent as non-canonical requirements [13].

Intuitively, a feasible proposal for inconsistency handling should take into account other non-canonical requirements that are also involved in inconsistency. Previously, we have developed some logical tools for managing other non-canonical requirements as well as inconsistency. For example, we have presented a logical tool for uniformly characterizing inconsistency, incompleteness, vagueness, and redundancy in requirements specifications [13]. We have also proposed an approach to detecting inconsistency, incompleteness and redundancy in requirements specifications based on Answer Set Programming [14]. These methods focus on how to identify non-canonical requirements when detecting inconsistency rather than on the problem of systematically managing non-canonical requirements.

It has increasingly been recognized that measuring inconsistency is crucial for effectively managing inconsistency in requirements [5,15,11]. Furthermore, appropriate measurements of non-canonical requirements could provide a good basis for making some trade-off decisions on handling non-canonical requirements. For example, software developers would need to know whether a requirements specification would become more incomplete if some inconsistent requirements are removed. They would also need to know whether some requirements changes for resolving inconsistency in a requirements specification would increase the degree of redundancy in the specification. On the other hand, the most redundant requirements involved in inconsistency should be given priority for consideration when there are options for requirements changes.

To address these issues, in this paper, we propose a logical framework for managing non-canonical requirements, including inconsistent, incomplete, and redundant requirements. First, we formulate a requirements specification as a set of logical formulas. Second, we present a framework for managing non-canonical requirements, which consists of five main parts, identifying non-canonical requirements, measuring them, generating candidate proposals for handling them, choosing commonly acceptable proposals, and revising them according to the chosen proposals. Third, we provide a family of measures for non-canonical requirements, including measures for inconsistency, incompleteness, and redundancy. Finally, we discuss some strategies for generating proposals for handling non-canonical requirements by using these measures.

The rest of this paper is organized as follows. Section 2 gives a brief introduction to the logical representation of requirements. We propose a general framework for managing non-canonical requirements in Section 3. Section 4 proposes measures for non-canonical requirements. Section 5 discusses strategies for generating proposals for handling non-canonical requirements. We compare our approach with related work in Section 6. We conclude the paper in Section 7.

2. Preliminaries

First-order logic has increasingly been considered as a promising tool for representing requirements [2,9,10]. Moreover, restricting first-order logic to propositional logic is a useful and practical way of balancing the computational advantages of propositional logic and its limited expressive power in requirements engineering as well as software engineering [1,9,10,16]. In this paper, we use a classical first-order language without function symbols and existential quantifiers. Classical first-order logic is the most convenient to illustrate our approach, as will be seen in the rest of the paper.

Let $\mathscr{P}$ be a set of predicate symbols, $\mathscr{V}$ be a set of variable symbols, and $\mathscr{C}$ a set of constant symbols. We call $\mathscr{A} = \{p(q_1, \ldots, q_n) | p \in \mathscr{P}$ and $q_1, \ldots, q_n \in \mathscr{V} \cup \mathscr{C}\}$ the set of atoms. Let $\mathscr{F}$ be the set of classical formulas formed from a set of atoms $\mathscr{A}$ and a set of logical connectives $\{\vee, \wedge, \neg, \rightarrow\}$.

In particular, we call $p(q_1, \ldots, q_n)$ a ground atom if and only if $q_1, \ldots, q_n$ are all constant symbols. Let $\mathscr{A}_0$ be a set of ground atoms and $\mathscr{F}_0$ be a set of classical formulas formed from a set of ground atoms $\mathscr{A}_0$ and a set of logical connectives $\{\vee, \wedge, \neg, \rightarrow\}$.

Let $\mathscr{G}$ be the set of ground formulas and prenex universally quantified formulas formed from $\mathscr{F}$, where $\forall X_1 \cdots \forall X_n \alpha \in \mathscr{G}$ if $\alpha \in \mathscr{F}$, and $X_1, \ldots, X_n$ are the free variables in α.

We can use formulas in $\mathscr{G}$ to formulate requirements expressed in natural language. For example, we can represent a requirement,"*If an authorized user requests to borrow a book and the book is available, then the user can borrow the book*", as

$$\forall \mathtt{User} \forall \mathtt{Book}\, (\mathtt{auth}(\mathtt{User}) \wedge \mathtt{requ}(\mathtt{User}, \mathtt{Book}) \wedge \mathtt{avai}(\mathtt{Book}) \rightarrow \mathtt{borr}(\mathtt{User}, \mathtt{Book})).$$

A scenario is a potential application setting of the system-to-be. For the sake of simplicity, we assume that a scenario consists of two parts, a set of facts to model the scenario, and a set of expected responses to model the expected behavior of the system-to-be. Moreover, facts and expected responses can be formulated by ground formulas in $\mathscr{F}_0$. We use $\langle S_I, S_E \rangle$ to denote a scenario, where S_I and S_E are the set of facts and the set of expected responses, respectively. Intuitively, we consider