A Blame-Based Approach to Generating Proposals for Handling Inconsistency in Software Requirements

Kedian Mu, Peking University, China

Weiru Liu, Queen's University Belfast, UK

Zhi Jin, Peking University, China

ABSTRACT

Inconsistency has been considered one of the main classes of defects in software requirements specification. Various logic-based techniques have been proposed to manage inconsistencies in requirements engineering. However, identifying an appropriate proposal for resolving inconsistencies in software requirements is still a challenging problem. This paper proposes a logic-based approach to generating appropriate proposals for handling inconsistency in software requirements. Informally speaking, given an inconsistent requirements specification, the authors identify which requirements should be given priority to be changed for resolving the inconsistency in that specification, by balancing the blame of each requirement for the inconsistency against its value for that requirements specification. The authors follow the viewpoint that minimal inconsistent subsets of a set of formulas are the purest forms of inconsistencies in that set. According to this viewpoint, a potential proposal for resolving inconsistencies can be described by a possible combination of some requirements to be changed that can eliminate minimal inconsistent subsets. Then a method is proposed of evaluating the degree of disputability of each requirement involved in the inconsistency in a requirements specification. Finally, an algorithm is provided of generating appropriate proposals for resolving the inconsistency in a given requirements specification based on the degree of disputability of requirements.

Keywords: *Degree of Disputability, Inconsistency Handling, Prioritized Knowledge Base, Software Requirements, The Blame for the Inconsistency*

1. INTRODUCTION

It has been increasingly recognized that inconsistency is inevitable during the requirements process (Easterbrook & Chechik, 2001a; Nuseibeh et al., 2001). Both general principles of managing inconsistency and special case-based approaches to handling inconsistency have recently been considered. In particular, it has been pointed out in Gervasi and Zowghi (2005) that the use of logic in managing inconsistency

DOI: 10.4018/jkss.2012010101

in requirements has been found to be effective in a number of studies. Various logic-based techniques have been proposed to manage inconsistencies in requirements engineering (Hunter & Nuseibeh, 1998; Gervasi & Zowghi, 2005; Martinez et al., 2008; Zowghi & Gervasi, 2003; Mu et al., 2005a, 2008, 2009). Most of these logic-based approaches focus on how to manage inconsistency by applying logical techniques such as paraconsistent reasoning and non-monotonic reasoning to requirements engineering. For example, Hunter and Nuseibeh (1998) developed the labeled quasi-classic logic to represent and reason about requirements specifications in the presence of inconsistency. Gervasi and Zowghi (2005) proposed methods for reasoning about inconsistencies in natural language requirements by combining natural language parsing techniques and non-monotonic reasoning. Easterbrook and Chechik (2001b) presented a framework termed χbel for merging inconsistent viewpoints using multi-valued logics. This framework was intended to highlight the source of inconsistency and to tolerate inconsistencies between viewpoints during model checking.

In contrast, there are relatively few logic-based techniques for generating appropriate proposals for inconsistency resolving actions in requirements engineering (Finkelstein et al., 1994; Gabbay & Hunter, 1993; Mu & Jin, 2007; Mu et al., 2008, 2009). Previously, we have argued that the relative priority of each requirement should play an important role in identifying appropriate proposals for resolving inconsistencies in requirement specifications (Mu & Jin, 2007; Mu et al., 2008, 2009), moreover, negotiation and combinatorial vote may be considered as two appropriate mechanisms of group decision making for identifying acceptable common proposals for handling inconsistent requirements specification (Mu et al., 2008, 2009). However, identifying appropriate actions for resolving inconsistency in requirements specification is still a challenging problem (Hunter & Nuseibeh, 1998). Generally, the choice of inconsistency handling actions is a rather context-sensitive issue (Finkelstein et al., 1994; Gabbay & Hunter, 1993). So, as pointed out in Mu et al. (2008), a feasible proposal for inconsistency resolving should focus on pointing out which requirements to be changed rather than how to change these requirements.

Roughly speaking, all the requirements involved in inconsistencies can be considered disputable. Each of such requirements is a candidate for requirements to be changed during the process of inconsistencies resolving. However, in many cases in requirements engineering, not all the requirements involved in inconsistencies need to be changed to resolve inconsistencies. Intuitively, the choice of requirements to be changed should depend on the evaluation of the blame of each requirement for inconsistencies in requirements specifications as well as the evaluation of the value of each requirement. To address this, in this paper, we present an approach to generating appropriate proposals for resolving inconsistencies in requirements specifications. This approach focuses on identifying requirements to be changed to resolve inconsistencies by balancing the blame of each requirement for inconsistencies against its value to the system-to-be. Informally, we formulate requirements specifications as prioritized knowledge bases in classical logic. Then we adopt the approach to measuring the blame of each formula for inconsistent prioritized knowledge bases presented in Mu et al. (2011) to evaluate the blame of each requirement for inconsistencies in an individual requirements set. Following this, we measure how disputable an individual requirement involved in inconsistency is by balancing the blame of that requirement against its priority. Finally, we propose an algorithm of choosing requirements to be changed based on this measurement.

The rest of this paper is organized as follows. Section 2 gives a brief introduction to the logical representation of requirements. We propose an approach to measuring how disputable a requirement involved in inconsistency is by balancing the blame of each requirement against its priority in Section 3. Section 4 proposes an algorithm of choosing requirements to be changed based on the degree of disputability.

We compare our approach with related work in Section 5. Finally, we conclude the paper in Section 6.

2. PRELIMINARIES

We use classical logic-based language to represent requirements in this paper. First order logic may be considered as a promising tool to represent requirements, since most tools and notations for representing requirements could be translated into formulas of first order logic (Hunter & Nuseibeh, 1998). Moreover, in a logic-based framework for representing requirements, consistency checking is always associated with certain scenarios with regard to the requirements specification (Hunter & Nuseibeh, 1998), or some specific domain knowledge. That is, we must add further relevant facts (e.g., domain knowledge) to model each scenario. Then reasoning about requirements is always based on these certain facts. It implies that checking the consistency of requirements considers only ground formulas. Furthermore, if we assume a universally quantified formula is just an abbreviation for the conjunction of formulas that can be formed by systematically instantiating the variables of the quantified formula with the constants in the language, then we may restrict the first order language to the propositional case. It will render consistency checking decidable. This gives some computational advantages. However, restricting first order logic to propositional logic in some way is a useful and practical way of balancing the computational advantages of propositional logic against its limited expressive power in requirements engineering as well as software engineering (Gervasi & Zowghi, 2005; Jackson, 2000). For these reasons, we assume a classical first order language without function symbols and existential quantifiers. This classical first order logic is the most convenient to illustrate our approach, as will be shown in the rest of the paper.

Let P be a set of predicate symbols, V be a set of variable symbols, and C a set of constant symbols. We call $\mathsf{A} = \{p(q_1, \ldots, q_n) \mid p \in \mathsf{P}$ and $q_1, \ldots, q_n \in \mathsf{V} \cup \mathsf{C}\}$ the set of atoms. Let F be the set of classical formulas formed from a set of atoms A and logical connectives $\{\vee, \wedge, \rightarrow, \neg\}$. 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 A_0 be a set of ground atoms. Let F_0 be the set of classical formulas formed from a set of atoms A_0 and logical connectives $\{\vee, \wedge, \rightarrow, \neg\}$. Let G be the set of formulas formed from F, where if $\alpha \in \mathsf{F}$, and $X_1, \ldots, X_n$ are the free variables of α, then $\forall X_1, \ldots, \forall X_n\ \alpha \in \mathsf{G}$. Essentially, the set G contains only universally quantified formulas (in which the quantifiers are outermost) and ground formulas.

A classical knowledge base K is a finite set of formulas in F_0. K is inconsistent if there is a formula $\alpha \in \mathsf{F}_0$ such that $K \vdash \alpha$ and $K \vdash \alpha$. We abbreviate $\alpha \wedge \alpha$ as $\perp$ if there is no confusion. Then an inconsistent knowledge base K is denoted by $K \vdash \perp$. Moreover, an inconsistent knowledge base K is called a *minimal inconsistent set* if none of its proper subset is inconsistent. If $K' \subseteq K$ and K' is a minimal inconsistent set, then we call K' a *minimal inconsistent subset* of K.

Let $\mathrm{MI}(\mathrm{K})$ be the set of all the minimal inconsistent subsets of K, i.e.,

$$MI(K) = \left\{ \begin{array}{l} K' \subseteq K \mid K' \vdash \perp, \\ and\ \ \forall K'' \subset K', K'' \nvdash \perp \end{array} \right\}.$$

The minimal inconsistent subsets can be considered as the purest form of inconsistency for conflict resolution where the syntactic representation of the information is important, since removing one formula from each minimal inconsistent subset would be sufficient to resolve the inconsistency (Reiter, 1987). In contrast, *a free formula* of a knowledge base K is referred to as a formula of K that does not belong to any minimal inconsistent subset of K. In this paper, we use $\mathrm{FREE}(\mathrm{K})$ to denote the set of free formulas of K.

We can use formulas in 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

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

However, to check inconsistency of requirements collections, the universally quantified formulas are always instantiated by the constants in certain scenarios. For example, given the following facts: "*Alice is an authorized user, and she applies to borrow the book of software engineering*; *The book of software engineering is available*". Then we use the following ground formula as a substitute for the universally quantified formula above:

$$auth(Alice) \wedge requ(Alice, Soft\_eng) \wedge avai(Soft\_eng) \rightarrow borr(Alice, Soft\_eng)$$

Generally, if ground formulas $\alpha_1, \alpha_2, \ldots, \alpha_n$ are the instantiations of the universally quantified formula α by using different facts in a scenario, then we may use $\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n$ as a substitute for α in the scenario. Thus, we concentrate on the instantiated requirements in the rest of this paper. That is, we assume that an individual set of requirements can be formulated by a classical knowledge base. With this, we restrict the first order logical representation of requirements to the propositional case.

In particular, we call a knowledge base K a (partial) requirements specification if each formula of K represents a requirement. If there is no confusion we make no distinction between a classical knowledge base and a requirements specification in the rest of this paper.

On the other hand, it has been increasingly recognized that the relative importance of requirements can help stakeholders to make some necessary trade-off decisions for resolving inconsistency. To address this, we need to attach a weight or qualitative priority level to each formula that represents an individual requirement. For convenience and simplicity and without losing generality, we assume that the set of priorities used in this paper is $(0,1]$. Let K be a classical knowledge base, then a prioritization over K is a function f_{K} from K to $(0,1]$ such that the bigger the priority value of a formula, the more preferred is the formula. By this, we can use $\langle K, f_K \rangle$ to formulate prioritized requirements specification. For simplicity, we call K, f_K a prioritized knowledge base. Note that this kind of prioritized knowledge base is exactly Type-I prioritized knowledge base defined in Mu et al. (2011).

We use the following example to illustrate the formulation of requirements in the form of classical logic formulas.

Example 1

Consider the following requirements for updating an existing software system. A representative of the sellers of the new system, provides the following demands:

(a) The system-to-be should be open, that is, the system-to-be could be extended easily.
(b) The system-to-be should be secure.
(c) The user interface of the system-to-be should be fashionable.

A representative of the users of the existing system, provides the following demands:

(d) The system-to-be should be developed based on the techniques used in the existing system;
(e) The user interface of the system-to-be should maintain the style of the existing system.

The domain expert in requirements engineering provides the following constraint, which is a consequence of (b) above:

(f) To guarantee the security of the system-to-be, openness (or ease of extension) should not be considered.

With regard to the prioritization over these requirements, suppose that both (b) and (f) are assigned to 0.9. Both (a) and (c) are assigned to 0.6, and (e) is assigned to 0.4. (d) is assigned to 0.7.

If we

- Use the predicate $\mathrm{Open}(\mathrm{sys})$ to denote that the system is open;
- Use the predicate $\mathrm{Fash}(\mathrm{int\_f})$ to denote that the interface is fashionable;
- Use the predicate $\mathrm{Exis}(\mathrm{sys})$ to denote that the system will be developed based on the techniques used in the existing system;
- Use the predicate $\mathrm{Secu}(\mathrm{sys})$ to denote that the system is secure.

Then we have a prioritized knowledge base $\mathrm{K}, \mathrm{f}_{\mathrm{K}}$ for the requirements above, where

$$K = \{Open(sys), Secu(sys), \neg Fash(\mathrm{int\_}f), Exis(sys), Fash(\mathrm{int\_}f), Secu(sys) \to \neg Open(sys)\},$$

and $\mathrm{f}_{\mathrm{K}} : \mathrm{K} \mapsto (0,1]$ such that

$$f_K(Open(sys)) = 0.6,$$
$$f_K(Fash(\mathrm{int\_}f)) = 0.6,$$
$$f_K(\neg Fash(\mathrm{int\_}f)) = 0.4,$$

$$f_K(Exis(sys)) = 0.7, f_K(Secu(sys)) = f_K(Secu(sys) \to Open(\neg sys)) = 0.9.$$

Clearly, the following inconsistencies can be identified from these requirements:

$$K \vdash Open(sys) \wedge \neg Open(sys),$$
$$K \vdash Fash(\mathrm{int\_}f) \wedge \neg Fash(\mathrm{int\_}f).$$

And the set of minimal inconsistent subsets of K is

$$MI(K) = \{\{Fash(\mathrm{int\_}f) \wedge Fash(\mathrm{int\_}f)\}, \{Open(sys), Secu(sys), ecu(sys) \to \neg Open(sys)\}\}$$

The set of free formulas of K is $\mathrm{FREE(K)} = \{\ \mathrm{Exis(sys)}\}$.

3. MEASURING THE DEGREE OF DISPUTABILITY

"Inconsistency Implies Actions" is recognized as a meta-rule for inconsistency handling (Gabbay & Hunter, 1993; Hunter & Nuseibeh, 1998) in many application domains. However, as mentioned earlier, identifying appropriate actions for resolving inc¬ onsistency is still a challenging issue in requirements engineering. A feasible general approach to handling inconsistencies in requirements should focus on identifying some potential requirements to be changed rather than identifying potential actions for changing them. That is, we need to know which requirements are disputable and how disputable these requirements are.

To characterize that some formulas are more disputable than others in an inconsistent knowledge base, we define the degree of disputability of a formula. Intuitively, given an inconsistent knowledge base, each of the formulas involved in minimal inconsistent subsets of that knowledge base may be considered disputable, since removing this formula can eliminate at least one minimal inconsistent subset. This motivates us to present the following general definition of the degree of disputability.

Definition 1

(The degree of disputability) Let $\mathrm{K}, \mathrm{f}_{\mathrm{K}}$ be a prioritized knowledge base. A degree of disputability function for K, denoted d_{K}, is a function from K to $[0, +\infty)$ such that

(1C) $d_K(\alpha) = 0$

If

$\alpha \in FREE(K)$.

(2C) $d_K(\alpha) > 0$

If

$\exists M \in MI(K)\, s.t.\ \alpha \in M$.

Note that this definition of the degree of disputability function provides only intuitive constraints on the degree of disputability. The first condition states that each free formula of a knowledge base has null degree of disputability. This accords with the viewpoint that free formulas have nothing to do with inconsistencies conveyed by minimal inconsistent subsets. The second condition ensures that any formula involved in minimal inconsistent subsets is disputable.

The simplest type of the degree of disputability function one can define is the drastic MinInc inconsistency value defined in Hunter and Konieczny (2008).

Definition 2

Let $\langle K, f_K \rangle$ be a prioritized knowledge base. d_K^1 is defined as:

$$\forall \alpha \in K, d_K^1(\alpha) = MIV_D(K,\alpha) = \begin{cases} 0, & \alpha \in FREE(K), \\ 1, & \exists M \in MI(K)\, s.t. \alpha \in M. \end{cases}$$

Note that d_K^1 allows us just to make a distinction between free formulas and disputable formulas. It cannot make a distinction between two formulas involved in minimal inconsistent subsets of a knowledge base, as shown in Hunter and Konieczny (2008). However, to identify desirable proposals for inconsistency resolving actions, we need to choose some formulas to be changed from these disputable formulas in some systematic way. Then it is necessary to make a distinction between these disputable formulas.

It is intuitive to use the blame of each formula for the inconsistency in a knowledge base to make a distinction between formulas in the knowledge base. For example, Hunter et al have argued that the blame of each formula in a flat (or classical) knowledge base can be used to stratify the knowledge base in Hunter and Konieczny (2010). However, in many practical software projects, developers need to balance the blame of each requirement for the inconsistency against its value for the system-to-be when making a necessary trade-off decision on inconsistency resolving. Then an intuitive measure for the degree of disputability of a requirement should take into account the relative importance of the requirement as well as the blame of the requirement for the inconsistency to be resolved. To address this, we refine the notation of the degree of disputability loosely defined by introducing the blame and the relative importance of requirements explicitly.

Definition 3

(The blame-based degree of disputability) Let $\langle K, f_K \rangle$ be a prioritized knowledge base. Let $Blame(K,\alpha)$ be the blame of α for inconsistencies in K. A blame-based degree of disputability for K, denoted d_K^B, is a function from K to $[0,+\infty)$ such that

$$d_K^B(\alpha) = 0 \tag{1}$$

if

$\alpha \in FREE(K)$.

$$d_K^B(\alpha) > 0 \tag{2}$$

if

$\exists M \in MI(K)\, s.t.\ \alpha \in M$.

$$\forall \alpha, \beta \in K \tag{3}$$

s.t.

$f_K(\alpha) = f_K(\beta)$,

if

$Blame(K,\alpha) \geq Blame(K,\beta)$

then

$d_K^B(\alpha) \geq d_K^B(\beta)$.
$\forall \alpha, \beta \in K$ (4)

s.t.

$Blame(K,\alpha) = Blame(K,\beta)$,

if

$f_K(\alpha) \geq f_K(\beta)$

then

$d_K^B(\alpha) \leq d_K^B(\beta)$.

Note that (1C) and (2C) are the two basic constraints for the degree of disputability mentioned above. The condition of (3C) requires that as the blame of a formula with a given priority increases, its degree of disputability cannot decrease. The last condition requires that as the priority of a formula with a given blame for inconsistency increases, its degree of disputable should decrease.

There are a number of functions appropriate for instantiating the blame-based degree of disputability defined. Now we give the following simple function as the blame-based degree of disputability.

Definition 4

Let $\mathrm{K}, \mathrm{f_K}$ be a prioritized knowledge base. Let $Blame(K,\alpha)$ be the blame of α for inconsistencies in K. The function $\mathrm{d_K^{B_0}}$ is defined as follows:

$$\forall \alpha \in K, d_K^{B_0}(\alpha) = \frac{Blame(K,\alpha)}{f_K(\alpha)}.$$

Note that $\mathrm{d_K^{B_0}}$ uses the ratio of the blame of a formula to the priority level of the formula to capture how disputable the formula is. This ensures that $\mathrm{d_K^{B_0}}$ satisfies the last two intuitive constraints about the blame-based degree of disputability.

Just for the simplicity of discussion, we can provide the following normalized version of $\mathrm{d_K^{B_0}}$ as follows.

Definition 5

Let $\mathrm{K}, \mathrm{f_K}$ be a prioritized knowledge base. Let $Blame(K,\alpha)$ be the blame of α for inconsistencies in K. The function $\mathrm{d_K^{B_1}}$ is defined as follows:

$$\forall \alpha \in K, d_K^{B_1}(\alpha) = \frac{Blame(K,\alpha)}{Blame(K,\alpha) + f_K(\alpha)}.$$

Note that $d_K^{B_1}(\alpha) = \dfrac{d_K^{B_0}(\alpha)}{d_K^{B_0}(\alpha)+1}$ and $0 \leq d_K^{B_1}(\alpha) < 1$ for all $\alpha \in K$. In essence, the degree of disputability function $d_K^{B_1}(\alpha)$ focuses on the ratio of the blame of α for inconsistencies in K to the relative importance of α, such that the most disputable formulas provide the largest fraction of the total blame for the inconsistency but have the smallest fraction of the total importance.

Evidently, both $\mathrm{d_K^{B_0}}(\alpha)$ and $\mathrm{d_K^{B_1}}(\alpha)$ satisfy the conditions (3C) and (4C). To be measures for the degree of disputability, both $\mathrm{d_K^{B_0}}(\alpha)$ and $\mathrm{d_K^{B_1}}(\alpha)$ need to satisfy the basic constraints (1C) and (2C). This also implies that the blame measure $\mathrm{Blame(K,\alpha)}$ we used should satisfy

$\mathrm{Blame}(\mathrm{K},\alpha) = 0$

if

$\alpha \in FREE(K)$.

$\mathrm{Blame(K,\alpha)} > 0$

if

$\exists M \in MI(K) s.t.\ \alpha \in M$.

However, these are exactly two of the essential properties of the measures for the blame of formulas for the inconsistency in a knowledge base (Mu et al., 2011). Therefore, given a blame measure, we can get a corresponding measure for the degree of disputability.

In this paper, we use two particular blame measures, i.e., $\mathrm{Blame}_{\mathrm{mean}}$ and $\mathrm{Blame}_{\mathrm{max}}$, to define the measures for the degree of disputability, respectively. Previously, we have proposed an approach to measuring the blame of each formula for inconsistencies in a prioritized knowledge base (Mu et al., 2011). Roughly speaking, this approach to measuring the blame of each formula for inconsistency is guided by the principle of proportionality, which insists on that the more important the formulas opposed to the formula are, the more severe the deserved blame of the formula should be. We make use of two particular measures of the blame of each formula for inconsistencies in a prioritized knowledge base presented in Mu et al. (2011).

Informally, for a given formula α of K, its blame for the inconsistency in a minimal inconsistent subset M is determined by the set of formulas of M that would be disengaged from the inconsistency if α was removed from M. We call the set of such formulas the set of opposed formulas to α w.r.t. M, and use $\mathrm{Opp(M,}\alpha)$ to denote it (Mu et al., 2011), i.e.,

$$Opp\left(M,\alpha\right) = \begin{cases} \alpha, & \textit{if } M = \{\alpha\}, \\ M - \{\alpha\}, & \textit{if } \{\alpha\} \subset M, \\ \varnothing, & \alpha \notin M. \end{cases}$$

Note that for a singleton set $\mathrm{M} = \{\alpha\}$, the opposed formula to α is α, since α is a self-contradictory formula.

Further, let $\mathrm{Sig(K)}$ be a particular measure for the relative importance of K. Given a minimal inconsistent subset M, let $\mathrm{Inc(M)}$ be the measure for the amount of inconsistency in M. Then under guidance of the principle of proportionality, the blame of a formula for the inconsistency in M, is defined as (Mu et al, 2011)

$$\forall\alpha \in K,\ Blame\left(M,\alpha\right) = \frac{Sig(Opp(M,\alpha))}{\sum_{\beta\in M} Sig(Opp(M,\beta))} \times Inc(M).$$

Moreover, the blame of a formula for the inconsistency in a knowledge base K is the sum of the blame of the formula for the inconsistency in each minimal inconsistent subset of the knowledge base (Mu et al., 2011), i.e.,

$$\forall\alpha \in K,\ Blame\left(K,\alpha\right) = \sum_{M\in MI(K)} Blame(M,\alpha)\ .$$

Based on different measures for the relative importance of a knowledge base and measures for the inconsistency in a minimal inconsistent subset, we can get different measures for the blame of each formula for the inconsistency. Previously, we have also proposed a set of properties to develop and to characterize such measures in Mu et al. (2011). In particular, if we use $Sig_{mean}(K) = \sum_{\gamma\in K} f_K(\gamma)$ to measure the relative importance of K in Mu et al. (2011). Then we define the blame of each formula for the inconsistency $\mathrm{Blame}_{\mathrm{mean}}$ as follows:

Definition 6

(The Blame of each formula for the Inconsistency $\mathrm{Blame}_{\mathrm{mean}}$) (Mu et al., 2011) Let $\mathrm{K, f_K}$ be a prioritized knowledge base. The blame of each formula belonging to K for the inconsistency of K, denoted $\mathrm{Blame}_{\mathrm{mean}}$, is a function such that

$$\forall\alpha \in K, Blame_{mean}\left(K,\alpha\right) = \sum_{M\in MI(K)} Blame_{mean}(K,\alpha),$$

where

$$Blame_{mean}(M,\alpha) = \frac{Sig_{mean}(Opp(M,\alpha))}{\sum_{\beta\in M} Sig_{mean}(Opp(M,\beta))} \times \frac{Sig_{mean}(M)}{|M|^2}$$

for each minimal inconsistent subset M of K.

Roughly speaking, the blame of α for the inconsistency of K is the accumulation of the blames of α for the inconsistency of each minimal inconsistent subset of K. Within a minimal inconsistent subset M, the amount of inconsistency in M is captured by $\frac{\mathrm{Sig}_{\mathrm{mean}}(\mathrm{M})}{|\mathrm{M}|^2}$, moreover, the blame of α for the inconsistency of M is proportionate to $Sig_{mean}(Opp(M,\alpha))$. We have shown in Mu et al. (2011) that $\mathrm{Blame}_{\mathrm{mean}}$ satisfies the set of intuitive properties an intuitive measure for the blame should have. In particular, $\mathrm{Blame}_{\mathrm{mean}}$ satisfies the properties of Innocence and Necessity, i.e.,

(B1) Innocence:

$$\forall M \in MI(K),$$

$$\forall \alpha \notin M,$$

$$\mathrm{Blame}_{\mathrm{mean}}(\mathrm{M},\alpha) = 0.$$

(B2) Necessity:

$$\forall M \in MI(K),$$

$$\forall \alpha \in M,$$

$$\mathrm{Blame}_{\mathrm{mean}}(\mathrm{M},\alpha) > 0.$$

We use the following example to illustrate this measure for the blame of each formula for inconsistency.

Example 2

Consider $\langle K_1, f_{K_1}\rangle$,
where $K_1 = \{a,\neg a,\neg a \vee c,b,\neg c,d\}$ and

$$f_{K_1}(a) = 0.6,$$
$$f_{K_1}(\neg a) = 0.4,$$
$$f_{K_1}(\neg a \vee c) = 0.8,$$

$$f_{K_1}(b) = 0.5,$$
$$f_{K_1}(\neg c) = 0.1,$$
$$f_{K_1}(d) = 0.9.$$

Then $\mathrm{MI}(\mathrm{K}_1) = \{\mathrm{M}_1,\mathrm{M}_2\}$, where $M_1 = \{a,\neg a\}$, $M_2 = \{\neg a \vee c,a,\neg c\}$. So, the blame of each formula for inconsistency in M_1 is given as follows:

$$\mathrm{Blame}_{\mathrm{mean}}(\mathrm{M}_1,\mathrm{a}) = 0.1,$$

$$\mathrm{Blame}_{mean}(M_1,\neg a) = 0.15,$$

$$\mathrm{Blame}_{mean}(M_1,\neg c) = 0,$$

$$\mathrm{Blame}_{mean}(M_1,\neg a \vee c) = 0,$$

$$\mathrm{Blame}_{\mathrm{mean}}(\mathrm{M}_1,\mathrm{b}) = 0,$$

$$\mathrm{Blame}_{\mathrm{mean}}(\mathrm{M}_1,\mathrm{d}) = 0.$$

The blame of each formula for inconsistency in M_2 is given as follows:

$$\mathrm{Blame}_{\mathrm{mean}}(\mathrm{M}_2,\mathrm{a}) = 0.05,$$

$$\mathrm{Blame}_{mean}(M_2,\neg a) = 0,$$

$$\mathrm{Blame}_{mean}(M_2,\neg c) = 0.08,$$

$$\mathrm{Blame}_{mean}(M_2,\neg a \vee c) = 0.04,$$

$$\mathrm{Blame}_{\mathrm{mean}}(\mathrm{M}_2,\mathrm{b}) = 0,$$

$$\mathrm{Blame}_{\mathrm{mean}}(\mathrm{M}_2,\mathrm{d}) = 0.$$

The blame of each formula for inconsistency in K_1 is given as

$$\mathrm{Blame}_{\mathrm{mean}}(\mathrm{K}_1,\mathrm{a}) = 0.15,$$

$\text{Blame}_{mean}(K_1, \neg a) = 0.15,$

$\text{Blame}_{mean}(K_1, \neg c) = 0.08,$

$\text{Blame}_{mean}(K_1, \neg a \vee c) = 0.04,$

$\text{Blame}_{\text{mean}}(\text{K}_1, \text{b}) = 0,$

$\text{Blame}_{\text{mean}}(\text{K}_1, \text{d}) = 0.$

This example also shows that the blame of a formula for inconsistency is insufficient for characterizing how disputable a formula is. To illustrate this, consider the example above,

$$\text{Blame}_{mean}\left(K_1, a\right) = Blame_{mean}(K_1, \neg a) = 0.15,$$

but it is intuitive to consider that a is more disputable in this case since $f_{K_1}\left(\neg a\right) < f_{K_1}\left(a\right)$. It also implies that the degree of disputability of a formula should be determined by the blame of that formula together with the priority of that formula.

By using the blame measure $\text{Blame}_{\text{mean}}$, we define the degree of disputability function $\text{d}_{\text{K}}^{\text{B}_{\text{mean}}}$ as follows.

Definition 7

Let $\text{K}, \text{f}_{\text{K}}$ be a prioritized knowledge base. The function $\text{d}_{\text{K}}^{\text{B}_{\text{mean}}}$ is defined as follows:

$$\forall \alpha \in K, d_K^{B_{mean}}(\alpha) = \frac{Blame_{mean}(K, \alpha)}{Blame_{mean}\left(K, \alpha\right) + f_K(\alpha)}.$$

Note that the properties of Innocence and Necessity of $\text{Blame}_{\text{mean}}$ ensure that $\text{d}_{\text{K}}^{\text{B}_{\text{mean}}}$ satisfies (C1) and (C2), i.e., $\text{d}_{\text{K}}^{\text{B}_{\text{mean}}}$ is a degree of disputability function. The following proposition shows that $\text{d}_{\text{K}}^{\text{B}_{\text{mean}}}$ is an anticipated measure for the degree of disputability of each formula for a prioritized knowledge base.

Proposition 1

$\text{d}_{\text{K}}^{\text{B}_{\text{mean}}}$ is a blame-based degree of disputability function, i.e., $\text{d}_{\text{K}}^{\text{B}_{\text{mean}}}$ satisfies (1C)-(4C).

Example 3

Consider $\left\langle K_1, f_{K_1} \right\rangle$ again. Then

$$\text{d}_{\text{K}_1}^{\text{B}_{\text{mean}}}(\text{a}) = \frac{1}{5},$$

$$d_{K_1}^{B_{mean}}(\neg a) = \frac{3}{11},$$

$$d_{K_1}^{B_{mean}}\left(\neg c\right) = \frac{7}{16},$$

$$d_{K_1}^{B_{mean}}(\neg a \vee c) = \frac{7}{151},$$

$$\text{d}_{\text{K}_1}^{\text{B}_{\text{mean}}}(\text{b}) = 0,$$

$$\text{d}_{\text{K}_1}^{\text{B}_{\text{mean}}}(\text{d}) = 0.$$

Besides the blame measure $\text{Blame}_{\text{mean}}$, we may also use the other blame measures to define the blame-based degree of disputability function.

Note that Sig_{mean} considers the sum of the priority values of all the formulas in a knowledge base as the relative importance or the priority of the whole knowledge base. It essentially uses the average of the priority levels of all the formulas to evaluate the relative importance of the whole knowledge base. However, in some cases, the highest priority level of the formulas involved in inconsistency plays a dominant role in evaluating the relative importance of inconsistency. To address this, we may use the highest priority level of formulas in a knowledge base to evaluate the relative importance of the knowledge base.

If we use

$$Sig_{\max}(K) = |K| \cdot \max\{f_K(\alpha) \mid \alpha \in K\}$$

to measure the relative importance of K, then the inconsistency in a minimal inconsistent subset M can be captured by $\mathrm{Inc}_{\max}(\mathrm{M}) = \frac{\mathrm{Sig}_{\max}(\mathrm{M})}{|\mathrm{M}|^2}$ in Mu et al. (2011). We can define another blame measure along this line.

Definition 8

(The Blame of each formula for the Inconsistency $\mathrm{Blame}_{\max}$ (Mu et al., 2011)) Let $\mathrm{K}, \mathrm{f_K}$ be a prioritized knowledge base. The blame of each formula belonging to K for the inconsistency of K, denoted $\mathrm{Blame}_{\max}$, is a function such that

$$\forall \alpha \in K, Blame_{\max}(K, \alpha) = \sum_{M \in MI(K)} Blame_{\max}(K, \alpha),$$

where

$$Blame_{\max}(M, \alpha) = \frac{Sig_{\max}(Opp(M, \alpha))}{\sum_{\beta \in M} Sig_{\max}(Opp(M, \beta))} \times \frac{Sig_{\max}(M)}{|M|^2}$$

for each minimal inconsistent subset M of K.

Note that the measure $\mathrm{Blame}_{\max}$ also accords with the principle of proportionality. That is, within a minimal inconsistent subset M, the blame of a formula for the inconsistency in M is proportionate to the relative importance of the opposed formulas to that formula. Moreover, the blame of a formula for the inconsistencies in a knowledge base is the sum of the blame of the formula for the inconsistency in each minimal inconsistent subset of the knowledge base.

We use the following example to illustrate the measure $\mathrm{Blame}_{\max}$.

Example 4

Consider $\langle K_2, f_{K_2} \rangle$,
where $K_2 = \{a, \neg a, \neg a \vee c, b, \neg c, d\}$ and

$$f_{K_2}(a) = 0.6,$$
$$f_{K_2}(\neg a) = 0.4,$$
$$f_{K_2}(\neg a \vee c) = 0.9,$$

$$f_2(b) = 0.7,$$
$$f_{K_2}(\neg c) = 0.3,$$
$$f_{K_2}(d) = 0.7.$$

Then $\mathrm{MI}(\mathrm{K}_1) = \{\mathrm{M}_1, \mathrm{M}_2\}$, where $M_1 = \{a, \neg a\}$, $M_2 = \{\neg a \vee c, a, \neg c\}$. So, the blame of each formula for inconsistency in M_1 is given as follows:

$$\mathrm{Blame}_{\max}(\mathrm{M}_1, \mathrm{a}) = 0.12,$$

$$\mathrm{Blame}_{\max}(M_1, \neg a) = 0.18,$$

$$\mathrm{Blame}_{\max}(M_1, \neg c) = 0,$$

$$\mathrm{Blame}_{\max}(M_1, \neg a \vee c) = 0,$$

$$\mathrm{Blame}_{\max}(\mathrm{M}_1, \mathrm{b}) = 0,$$

$$\mathrm{Blame}_{\max}(\mathrm{M}_1, \mathrm{d}) = 0.$$

The blame of each formula for inconsistency in M_2 is given as follows:

$$\mathrm{Blame}_{\max}(\mathrm{M}_2, \mathrm{a}) = 0.11,$$

$$\mathrm{Blame}_{\max}(M_2, \neg a) = 0,$$

$\text{Blame}_{\max}(M_2, \neg c) = 0.11,$
$\text{Blame}_{\max}(M_2, \neg a \vee c) = 0.075,$

$\text{Blame}_{\max}(\text{M}_2, \text{b}) = 0,$

$\text{Blame}_{\max}(\text{M}_2, \text{d}) = 0.$

The blame of each formula for inconsistency in K_1 is given as

$\text{Blame}_{\max}(\text{K}_2, \text{a}) = 0.23,$

$\text{Blame}_{\max}(K_2, \neg a) = 0.18,$

$\text{Blame}_{\max}(K_2, \neg c) = 0.11,$

$\text{Blame}_{\max}(K_2, \neg a \vee c) = 0.075,$

$\text{Blame}_{\max}(\text{K}_2, \text{b}) = 0,$

$\text{Blame}_{\max}(\text{K}_2, \text{d}) = 0.$

Evidently, the measure $\text{Blame}_{\max}$ also satisfies the properties of Innocence and Necessity. By using the blame measure $\text{Blame}_{\max}$, we define the degree of disputability function $\text{d}_{\text{K}}^{\text{B}_{\max}}$ as follows:

Definition 9

Let $\text{K}, \text{f}_{\text{K}}$ be a prioritized knowledge base. The function $\text{d}_{\text{K}}^{\text{B}_{\max}}$ is defined as follows:

$$\forall \alpha \in K, d_K^{B_{\max}}(\alpha) = \frac{Blame_{\max}(K, \alpha)}{Blame_{\max}(K, \alpha) + f_K(\alpha)}.$$

Evidently, $\text{d}_{\text{K}}^{\text{B}_{\max}}$ is also an anticipated measure for the degree of disputability of each formula for a prioritized knowledge base.

Proposition 2

$\text{d}_{\text{K}}^{\text{B}_{\max}}$ is a blame-based degree of disputability function, i.e., $\text{d}_{\text{K}}^{\text{B}_{\max}}$ satisfies (C1)-(C4).

We use the following example to illustrate the degree of disputability function $\text{d}_{\text{K}}^{\text{B}_{\max}}$.

Example 5

Consider $\langle K_2, f_{K_2} \rangle$ again. Then

$$\text{d}_{\text{K}_2}^{\text{B}_{\max}}(\text{a}) = \frac{23}{83},$$

$$d_{K_2}^{B_{\max}}(\neg a) = \frac{9}{29},$$

$$d_{K_2}^{B_{\max}}(\neg c) = \frac{11}{41},$$

$$d_{K_2}^{B_{\max}}(\neg a \vee c) = \frac{1}{13},$$

$$\text{d}_{\text{K}_2}^{\text{B}_{\max}}(\text{b}) = 0,$$

$$\text{d}_{\text{K}_2}^{\text{B}_{\max}}(\text{d}) = 0.$$

To compare the formulas of a given prioritized knowledge base in terms of their degrees of disputability, we define a total ordering relation over the prioritized knowledge base termed as *more disputable than* as follows.

Definition 10

(The relation of more disputable than, $\geq_{\text{d}}$) Let $\text{K}, \text{f}_{\text{K}}$ be a prioritized knowledge base and $\text{d}_{\text{K}}^{\text{B}}$ the blame-based degree of disputability for K. A binary relation on K, denoted $\geq_{\text{d}}$, is defined as follows:

$\alpha \geq_d \beta$ i f and only if $d_K^B(\alpha) \geq d_K^B(\beta)$.

Further, $\alpha >_d \beta$ if $\alpha \geq_d \beta$ and $\beta \not\geq_d \alpha$. $\alpha \simeq_d \beta$ if $\alpha \geq_d \beta$ and $\beta \geq_d \alpha$. We say that α *is strictly more disputable than* β if $\alpha >_d \beta$.

Example 6

Consider $\langle K_1, f_{K_1} \rangle$ again. Suppose that we use $d_K^{B_{mean}}$ to measure the degree of disputability.

Then

$$\neg c >_d \neg a >_d a >_d \neg a \vee c >_d b \simeq_d d$$

According to this ordering relation, a is more disputable than $\neg a$, although they have the same blame for inconsistencies in K_1. This comparison is intuitive, since the priority of a is higher than that of a.

4. GENERATING APPROPRIATE PROPOSALS FOR HANDLING INCONSISTENCY

As mentioned above, a useful general approach to generating proposals for handling inconsistency should focus on pointing out which requirements to be changed rather than how to change these requirements. In this sense, generating proposals for handling inconsistency is a process of identifying potential requirements to be changed.

Without loss of generality, we use $d_K^{B_{mean}}$ as the degree of disputability function in the following examples if there is no confusion.

Definition 11

Let K, f_K be an inconsistent prioritized knowledge base. A proposal for handling the inconsistency in K, denoted $\pi(K)$, is a subset of K such that $MI\left(K - \pi\left(K\right)\right) = \varnothing$.

Note that $\pi(K)$ is a subset of K such that all the minimal inconsistent subsets of K would be eliminated if formulas of $\pi(K)$ were removed from K. For example, both $\{a\}$ and $\{\neg a, \neg c\}$ are proposals for handling inconsistency in K_1.

How to evaluate the appropriateness of a proposal is also a difficult issue in requirements engineering. Intuitively, the more disputable requirements should be given a priority to be included in a proposal. We present an algorithm for generating proposals which provides support for this intuition.

Definition 12

Let K, f_K be an inconsistent prioritized knowledge base. $\forall \pi_1, \pi_2 \in \Pi$, π_1 is more appropriate than π_2 for handling inconsistency in K if $\forall M \in MI(K)$, $\exists \alpha \in M \bigcap \pi_1$ such that $\alpha \geq_d \beta$ for all $\beta \in M \bigcap \pi_2$.

For example, for $\langle K_1, f_{K_1} \rangle$, the proposal $\{\neg a, \neg c\}$ is more appropriate than $\{\neg a, \neg a \vee c\}$.

Let $MAX(K)$ be the set of formulas of K with the highest degree of disputability, i.e.,

$$MAX\left(K\right) = \{\alpha \mid \alpha \in K, \\ and \forall \beta \in K,\ d_K^B(\beta) \leq d_K^B(\alpha)\}.$$

For example, $MAX\left(K_1\right) = \{\neg a\}$ and $MAX(\{b, d\}) = \{b, d\}$. Let Q be a set of subsets of K, then we abbreviate $\bigcup_{K' \in Q} K'$ as $\bigcup Q$. Then an algorithm for generating proposals for handling inconsistency based on the degree of disputability is given as shown in Box 1.

Note that condition $Q = \varnothing$ ensures that each proposal π generated by the algorithm satisfies $MI\left(K - \pi\left(K\right)\right) = \varnothing$. In contrast, the part from Line 6 to Line 10 of the algorithm ensures that π is one of the most appropriate proposals for handling inconsistency in K.

Example 7

Consider $\langle K_1, f_{K_1} \rangle$ again. Evidently, $\Pi_1 = \{\{\neg c\}\}$, $\Pi_2 = \{\{\neg c, \neg a\}\}$. Then the proposal for inconsistency handling generated is $\pi(K_1) = \{\neg a, \neg c\}$.

We use the following example to illustrate the algorithm.

Box 1.

```
**Input**:   A prioritized knowledge base $\langle K, f_K \rangle$
**Output**:  A set of proposals  $\Pi$
[1]   $\Pi_0 \leftarrow \varnothing$
[2]   $T \leftarrow 0$
[3]   $Q \leftarrow MI(K)$
[4]   **while**  $Q \neq \varnothing$
[5]           **do**   $\Pi_{T+1} \leftarrow \varnothing$
[6]                **for each**  $\pi \in \Pi_T$
[7]                    $Q \leftarrow MI(K) - \{M \mid M \in MI(K),\ M \cap \pi \neq \varnothing\}$
[8]                    **for each**  $\alpha \in MAX(\bigcup Q)$
[9]                            $\pi \leftarrow \pi \cup \{\alpha\}$
[10]                           $\Pi_{T+1} \leftarrow \Pi_{T+1} \cup \{\pi\}$
[11]               $T \leftarrow T+1$
[12]  return $\Pi_T$
```

Example 8 (Example 1 Continued)

Then

$$d_K^{B_{mean}}\left(\neg Fash(\text{int}\_f)\right) = \frac{3}{11} > d_K^{B_{mean}}$$
$$\left(Fash\left(\text{int}\_f\right)\right) = d_K^{B_{mean}}\left(\neg Open(sys)\right) = \frac{1}{7} >$$

$$\mathrm{d}_{\mathrm{K}}^{\mathrm{B}_{\mathrm{mean}}}\left(\mathrm{Secu(sys)}\right)$$
$$= \quad \mathrm{d}_{\mathrm{K}}^{\mathrm{B}_{\mathrm{mean}}}\left(\mathrm{Secu(sys)} \rightarrow \mathrm{Open(sys)}\right)$$
$$= \mathrm{d}_{\mathrm{K}}^{\mathrm{B}_{\mathrm{mean}}}\left(\mathrm{Exis(sys)}\right) = 0.$$

So, based on the algorithm,

$$\Pi_1 = \{\{\neg Fash\left(\text{int}\_f\right)\}\},$$

$$\Pi_2 = \{\{\neg Fash\left(\text{int}\_f\right), Open(sys)\}\}.$$

Then

$$\pi(K) = \left\{\neg Fash\left(\text{int}\_f\right), Open(sys)\right\}$$

is the generated proposal, i.e., the requirements (a) and (e) are recommended to be changed for resolving inconsistency. This proposal is intuitive.

5. RELATED WORK

Handling inconsistent requirements is a pervasive issue in requirements engineering.

Most of logic-based techniques for managing inconsistent requirements are concerned with reasoning about requirements in the presence of inconsistency. There are relatively few logic-based approaches to identifying appropriate proposals for handling inconsistency actions. In this paper we proposed an approach to identifying requirements to be changed from the set of requirements involved in inconsistencies by using the measure of the degree of disputability for requirements. In the following, we compare our approach with some of closely related proposals.

Our previous paper (Mu et al., 2008, 2009; Mu & Jin, 2007) presented two approaches to identifying acceptable common proposals for handling inconsistency in distributed requirements. However, this paper focuses on generating proposals for handling inconsistent requirements with weighted or numerical priorities within one perspective. The blame-based degree of disputability plays an important role in identifying potential requirements to be changed. In contrast, our previous papers (Mu et al., 2008 2009; Mu & Jin, 2007) are concentrated on multi-perspective requirements with qualitative priority levels (such as High and Low). These approaches emphasized the importance of group decision making mechanisms such as negotiation among multiple perspectives (Mu et al., 2008, 2009) and combinatorial vote (Mu & Jin, 2007) in identifying requirements to be changed.

Note that the blames of formulas for inconsistency are crucial for measuring how a formula disputable is. We make use of a particular measure for the blames of formulas for inconsistency defined in Mu et al. (2011) in characterizing the degree of disputability of formulas involved in inconsistency. Rough speaking, the blame of a formula measures how bad that formula is, in contrast, the priority of a formula states describes how good that formula is. In this sense, the blame-based degree of disputability of a formula balances the advantages of remaining that formula against the disadvantages.

Note that we translate a set of requirements with priority levels into a prioritized knowledge base in this paper. It is natural to translate requirements into a classical knowledge base when there is no prioritization over requirements. However, a classical knowledge base may be considered as a special kind of prioritized knowledge base if we consider each formula in the knowledge base has the highest priority value 1. Then the approach presented in this paper can also be applied to classical knowledge bases. But in such cases, $f_K(\alpha) = 1$ for all $\alpha \in K$, and then

$$d_K^{B_0}(\alpha) = \frac{Blame(K,\alpha)}{1}$$
$$= Blame(K,\alpha)$$

and

$$d_K^{B_1}(\alpha) = \frac{Blame(K,\alpha)}{Blame(K,\alpha)+1}.$$

This means such disputability functions are determined by only the blame of α for the inconsistency in K. That is, within the context of classical knowledge bases, the degree of disputability essentially accords with the idea of stratifying knowledge bases by the blame of formulas presented in Hunter and Konieczny (2010).

6. CONCLUSION

We have presented an approach to generating proposals for handling inconsistent requirements specification. This paper presented the following contributions to managing inconsistency in requirements engineering:

(a) We argued that how disputable a requirement is depends on the blame of that requirement for inconsistency as well as the priority of that requirement.
(b) We defined two blame-based degree of disputability functions for formulas of a

prioritized knowledge base by balancing the blames of formulas for inconsistency against the relative priorities of formulas. Also, we defined a total ordering relation termed *more disputable than* over a prioritized knowledge base.

(c) We presented an algorithm for generating appropriate proposals for handling inconsistent requirements by using the relation of *more disputable than* over inconsistent requirements specification.

Note that we assume that the priority level of each requirement is a real number in (0,1]. However, in many software projects, the priority level of requirements is a qualitative value such as High or Low. How to handle inconsistency in such prioritized requirements will be the main direction for our future work.

ACKNOWLEDGMENTS

This work was partly supported by the National Natural Science Foundation of China under Grant No. 61170300, the National Basic Research 973 program of China under Grant No. 2009CB320701, the Key Project of National Natural Science Foundation of China under Grant No. 90818026, and the NSFC and the British Royal Society China-UK Joint Project. This paper is a revised and extended version of the paper "An Approach to Generating Proposals for Handling Inconsistent Software Requirements" presented at KSEM2011.

REFERENCES

Easterbrook, S., & Chechik, M. (2001a). 2nd international workshop on living with inconsistency. *Software Engineering Notes*, *26*(6), 76–78. doi:10.1145/505532.505552

Easterbrook, S., & Chechik, M. (2001b). A framework for multi-valued reasoning over inconsistent viewpoints. In *Proceedings of the International Conference on Software Engineering*, Toronto, ON, Canada (pp. 411-420). Washington, DC: IEEE Computer Society.

Finkelstein, A., Gabbay, D., Hunter, A., Kramer, J., & Nuseibeh, B. (1994). Inconsistency handling in multiperspective specifications. *IEEE Transactions on Software Engineering*, *20*(8), 569–578. doi:10.1109/32.310667

Gabbay, D., & Hunter, A. (1993). Making inconsistency respectable 2: Meta-level handling of inconsistent data. In M. Clarke, R. Kruse, & S. Moral (Eds.), *Proceedings of the European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, Granada, Spain (LNCS 474, pp. 129-136).

Gervasi, V., & Zowghi, D. (2005). Reasoning about inconsistencies in natural language requirements. *ACM Transactions on Software Engineering and Methodology*, *14*(3), 277–330. doi:10.1145/1072997.1072999

Hunter, A., & Konieczny, S. (2008). Measuring inconsistency through minimal inconsistent sets. In *Proceedings of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning*, Sydney, Australia (pp. 358-366). Palo Alto, CA: AAAI Press.

Hunter, A., & Konieczny, S. (2010). On the measure of conflicts: Shapley inconsistency values. *Artificial Intelligence*, *174*(14), 1007–1026. doi:10.1016/j.artint.2010.06.001

Hunter, A., & Nuseibeh, B. (1998). Managing inconsistent specification. *ACM Transactions on Software Engineering and Methodology*, *7*(4), 335–367. doi:10.1145/292182.292187

Jackson, D. (2000). Automating first-order relational logic. *ACM SIGSOFT Software Engineering Notes*, *25*(6), 130–139. doi:10.1145/357474.355063

Martinez, A. B., Arias, J. P., Vilas, A. F., Duque, J. G., Norse, M. L., Redondo, R. P., & Fernandez, Y. B. (2008). On the interplay between inconsistency and incompleteness in multi-perspective requirements specifications. *Information and Software Technology*, *50*(4), 296–321. doi:10.1016/j.infsof.2007.02.001

Mu, K., & Jin, Z. (2007). Identifying acceptable common proposals for handling inconsistent software requirements. In J. Derrick & J. Vain (Eds.), *Proceedings of the 27th IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems*, Tallinn, Estonia (LNCS 4574, pp. 296-308).

Mu, K., Jin, Z., Lu, R., & Liu, W. (2005). Measuring inconsistency in requirements specifications. In L. Godo (Ed.), *Proceedings of the 8th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, Barcelona, Spain (LNCS 3571, pp. 440-451).

Mu, K., Jin, Z., & Zowghi, D. (2008). A priority-based negotiations approach for handling inconsistency in multi-perspective software requirements. *Journal of Systems Science and Complexity*, *21*(4), 574–596. doi:10.1007/s11424-008-9136-4

Mu, K., Liu, W., & Jin, Z. (2011). Measuring the blame of each formula for inconsistent prioritized knowledge bases. *Journal of Logic and Computation.*

Mu, K., Liu, W., Jin, Z., Yue, A., Lu, R., & Bell, D. (2009). Handling inconsistency in distributed software requirements specifications based on prioritized merging. *Fundamenta Informaticae*, *91*(3-4), 631–670.

Nuseibeh, B., Easterbrook, S., & Russo, A. (2001). Making inconsistency respectable in software development. *Journal of Systems and Software*, *58*(2), 171–180. doi:10.1016/S0164-1212(01)00036-X

Reiter, R. (1987). A theory of diagnosis from first principles. *Artificial Intelligence*, *32*(1), 57–95. doi:10.1016/0004-3702(87)90062-2

Zowghi, D., & Gervasi, V. (2003). On the interplay between consistency, completeness, and correctness in requirements evolution. *Information and Software Technology*, *45*(14), 993–1009. doi:10.1016/S0950-5849(03)00100-9