Science of Computer Programming ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at SciVerse ScienceDirect

Science of Computer Programming

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

Measuring the significance of inconsistency in the Viewpoints framework

Kedian Mu[a,∗], Zhi Jin[b,c], Weiru Liu[d], Didar Zowghi[e], Bo Wei[f]

[a] *School of Mathematical Sciences, Peking University, Beijing 100871, PR China*
[b] *Key Laboratory of High Confidence Software Technologies (Peking University), Ministry of Education, Beijing 100871, PR China*
[c] *School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, PR China*
[d] *School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, BT7 1NN, UK*
[e] *School of Software, Faculty of Engineering and Information Technology, University of Technology, Sydney, Australia*
[f] *Academy of Mathematics and System Science, Chinese Academy of Sciences, Beijing 100190, PR China*

A R T I C L E   I N F O

Article history:
Received 27 December 2010
Received in revised form 11 October 2012
Accepted 10 December 2012
Available online xxxx

Keywords:
Requirements
Measure of inconsistency
Significance of inconsistency
Viewpoints

A B S T R A C T

Measuring inconsistency is crucial to effective inconsistency management in software development. A complete measurement of inconsistency should focus on not only the degree but also the significance of inconsistency. However, most of the approaches available only take the degree of inconsistency into account. The significance of inconsistency has not yet been given much needed consideration. This paper presents an approach for measuring the significance of inconsistency arising from different viewpoints in the Viewpoints framework. We call an individual set of requirements belonging to different viewpoints *a combined requirements collection* in this paper. We argue that the significance of inconsistency arising in a combined requirements collection is closely associated with global priority levels of requirements involved in the inconsistency. Here we assume that the global priority level of an individual requirement captures the relative importance of every viewpoint including this requirement as well as the local priority level of the requirement within the viewpoint. Then we use the synthesis of global priority levels of all the requirements in a combined collection to measure the significance of the collection. Following this, we present a scoring matrix function to measure the significance of inconsistency in an inconsistent combined requirements collection, which describes the contribution made by each subset of the requirements collection to the significance of the set of requirements involved in the inconsistency. An ordering relationship between inconsistencies of two combined requirements collections, termed *more significant than*, is also presented by comparing their significance scoring matrix functions. Finally, these techniques were implemented in a prototype tool called *IncMeasurer*, which we developed as a proof of concept.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

The development of any complex system always involves many different stakeholders with different background knowledge, technical expertise, responsibilities, and expectations of the benefit of the system [1,2]. Each stakeholder imagines the system-to-be from a particular perspective. The Viewpoints is a framework proposed to describe these different

∗ Corresponding author.
E-mail address: mukedian@math.pku.edu.cn (K. Mu).

0167-6423/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.scico.2012.12.006

perspectives and their relationships explicitly [1]. In the Viewpoints framework, the concerns about the system-to-be from a particular stakeholder, or a group of stakeholders are captured by a distinct viewpoint, and the requirements are represented by a collection of locally structured, loosely coupled and distributable viewpoints and constraints between them [2].

It seems to be inevitable to confront inconsistency in the Viewpoints framework. Different viewpoints invariably overlap, complement, or contradict each other [2]. Moreover, maintaining completeness and correctness of requirements specifications is always associated with the problem of inconsistency. For example, Zowghi et al. [3] argued that there is an important causal relationship between consistency, completeness and correctness of requirements. This also implies that inconsistency is not a self-contained problem in requirements engineering.

Contrary to viewing inconsistency as an absolute undesirable, it has been increasingly recognized that inconsistency may also play a positive role in facilitating requirements elicitation if management of inconsistency becomes more effective and flexible [4,5]. In particular, as pointed out in [6], the use of logic in managing inconsistency in requirements was proven to be effective in a number of studies. Various logic-based techniques have been proposed to manage inconsistencies in requirements engineering and software engineering [6–11]. Most of these techniques focus on a particular type of inconsistency in requirements engineering, i.e., logical contradictions: any situation in which some fact and its negation can be simultaneously derived from the same requirements set.

It has been recognized that measuring inconsistency is crucial for effective management of inconsistency in requirements engineering [4,5]. Appropriate assessment of inconsistencies in a requirements specification can provide sound bases for trade-off decision making on handling these inconsistencies. To achieve this, developers would like to know the degree of inconsistency arising from a requirements set. More importantly, developers also need to know the significance of inconsistency arising from that requirements set. Clearly, a stakeholder may consider that inconsistency $\alpha \wedge \neg\alpha$ is more important than $\beta \wedge \neg\beta$ if α is more important than β from his point of view. Therefore, a complete measurement of inconsistency in requirements should focus on the significance as well as the degree of inconsistency.

The degree of inconsistency and the significance of inconsistency are two essential aspects for articulating an individual inconsistent requirements collection (i.e., a set of requirements). The degree of inconsistency describes how inconsistent a requirements collection is, whilst the significance of inconsistency measures how important the inconsistency arising from a requirements collection is. Then they may be viewed as two distinct dimensions of measurement of an inconsistency.

Apart from requirements engineering, measuring inconsistency is one of the important topics in the Artificial Intelligence (AI for short) community [12,13]. Various techniques for measuring inconsistency have been proposed in the AI literature. The overwhelming majority of these techniques focus on different measures of the degree of inconsistency [14–18,13,19–21,12,22]. Hunter et al. have argued in [12] that these current proposals for measuring the degree of inconsistency may be classified into two categories: formula-based and variable-based. Roughly speaking, proposals such as [22,14,13] belong to the first category because these proposals involve counting the minimal number of formulas that are needed to cause inconsistency [12]. In contrast, proposals such as [15–18,13,19,20] belong to the second category because these approaches allow us to look inside the formulas involved in inconsistency [12]. Generally, the proposals of the first category are syntax sensitive, while the proposals of the second category are not. However, as argued in [13], the syntax sensitivity is necessary for formal representation of requirements. For example, $\{\alpha, \beta\}$ should be treated differently from $\{\alpha \wedge \beta\}$ since the first set states that there are two requirements α and β, whereas the second states that there is only one requirement $\alpha \wedge \beta$ [13]. In this sense, the proposals of the second category are less appropriate for measuring the inconsistency in requirements engineering. Among the proposals of the first category, the scoring function presented in [14] can be considered as one of the proposals appropriate for summarizing the degree of inconsistency, since it provides a succinct means for articulating the nature of inconsistency in a set of classical formulas [14].

In contrast, the significance of inconsistency has not yet been given much deserved consideration. There are relatively few investigations as how the significance of inconsistency can be measured [21,23]. The approach in the framework of Quasi-classical (QC for short) logic to evaluating the significance of inconsistency presented in [21] is one of the most representatives for measuring the significance of inconsistency in knowledge or belief bases. QC logic is a paraconsistent variation of the classical logic, which allows us to continue reasoning in the presence of inconsistency [9]. However, this approach is difficult to apply in RE, since it is based on specifying the relative significance of incoherent QC models using additional information, encoded as a mass assignment in the Dempster–Shafer theory [24]. In contrast, the scoring function-based approach presented in [23] can be considered as the first attempt to measuring the significance of inconsistency arising from prioritized requirements. It focused on measuring the inconsistency from a given viewpoint rather than multiple viewpoints. However, as illustrated later, this approach can be considered as a special case of our approach presented in this paper.

Intuitively, the significance of an inconsistency depends on the relative importance of requirements involved in the inconsistency. However, the same shared requirement among different viewpoints may have different relative importance within these different viewpoints. It may lead to a disagreement on the significance of a given inconsistency among these viewpoints. Thus developers need to reach an agreement on which inconsistencies are more important than others from a global perspective. That is, we should evaluate the significance of inconsistency based on the global priority of requirements involved in inconsistency, which considers the local priority of requirements within each viewpoint as well as the relative importance of each viewpoint. In addition, given an inconsistent requirements collection, it is desirable to identify most possible sources of inconsistency in the collection to know what specific requirements make more significant contribution to the inconsistency.

This paper concentrates on the problem of how to measure the significance of inconsistency arising from different viewpoints within the Viewpoints framework. This work extends Hunter's approach for dealing with the degree of inconsistency [14] by developing the measures of significance of inconsistency. We argue that the significance of a given requirements collection from the integrated perspective depends on the global relative importance of requirements in that collection. We thus present an innovative approach to measuring the significance of inconsistency arising from different viewpoints based on a novel scoring function. Roughly speaking, given an inconsistent requirements collection, the scoring matrix function is proposed to describe the contribution made by each subset of that collection to the inconsistency in that collection. An ordering relationship between inconsistencies of two requirements collections, termed *more significant than*, is also presented by comparing their significance scoring matrix functions. These techniques were implemented and tested in a proof of concept prototype tool called *IncMeasurer*. The effectiveness of the innovative techniques presented in this paper and the supporting tool are illustrated by a case study involving conflicting requirements from multiple viewpoints in an electronic health records system.

The rest of the paper is organized as follows. Section 2 introduces the theoretical foundations and formal details of the techniques developed. Section 3 provides a brief overview to our previous work on measuring the significance of inconsistency in a given viewpoint, which can be considered as a simple case of an approach presented in Section 4. Section 4 presents an approach to measuring the significance of inconsistency arising from different viewpoints. Section 5 gives a case study to illustrate the feasibility of our approach and its supporting tool in requirements engineering and Section 6 compares our work with other related research. Finally, we conclude this paper in Section 7.

2. Preliminaries

Throughout this paper, we use first order logic to formulate requirements in the Viewpoints framework. First of all, this paper aims to capture the underlying nature of conflicts between requirements, which should be independent of any individual representation of requirements. For example, the severity of conflicts between requirements should not depend on whether the requirements are in natural language or UML. Representation frameworks with better capability of articulating the nature of inconsistency strictly and unambiguously are more preferred. Moreover, heterogeneity of representation in the Viewpoints framework allows different viewpoints to use different notations to represent their requirements during the requirements stage. This requires that any chosen tool for formulating requirements in the Viewpoints framework may represent requirements expressed in different forms. However, most notations for representing requirements such as ER diagrams, dataflow diagrams, inheritance hierarchies, and much of the Z language, could be translated into formulas of first order logic [9]. These make first order logic more appealing for formulating requirements in general. Second, as pointed out in [6], a number of studies have shown that the use of logic in managing inconsistency in requirements is effective. This implies that some appropriate logic-based approaches to handling inconsistency are appropriate and applicable to requirements engineering. Third, as shown later, we characterize and measure inconsistency in terms of minimal inconsistent subsets, which captures the most essential aspects of characterization of conflicts, i.e., minimality and logic contradiction. This renders the approaches presented in this paper feasible to be adapted to other cases such as conflicts between requirements assertions presented in [8]. Allowing for these, first order logic is considered as a promising language to represent different viewpoints and their relationships uniformly.

Moreover, in a logic-based framework for representing requirements, consistency checking is always associated with certain scenarios with regard to the requirements specification [9,6], or some specific domain knowledge [8]. 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. This implies that checking the consistency of requirements collections only considers ground formulas[1] rather than unground formulas. Furthermore, if we assume a universally quantified formula is just an abbreviation for a conjunction of the formulas that can be formed by systematically instantiating the variables of the quantified formula with the constants in the language used in formulation of requirements, then we may restrict the first order language to the propositional case [9]. It is well known that consistency checking is decidable in the propositional case, then this restriction will bring some computational advantages, especially for tool support on validation of our approaches. Actually, 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 [9,6,25,26,11]. For these reasons, we adopt a classical first order language without function symbols and existential quantifiers used in [9] to represent viewpoints in this paper. This classical first order logic is convenient to illustrate our approach, as will be shown in the rest of the paper.

We use the following notations about first order logic used in [9]. Let $\mathscr{P}$ be a set of predicate symbols. Let $\mathscr{V}$ be a set of variable symbols, and $\mathscr{C}$ a set of constant symbols. We call $\mathscr{A} = \{p(q_1, \ldots, q_n) | p \in \mathscr{P} \text{ and } q_1, \ldots, q_n \in \mathscr{V} \cup \mathscr{C}\}$ the set of atoms. Let $\mathscr{F}$ be the set of classical formulas formed from $\mathscr{A}$ and logical connectives $\{\vee, \wedge, \neg, \rightarrow\}$. In particular, we call $p(q_1, \ldots, q_n)$ a ground atom if and only if $q_1, \ldots, q_n \in \mathscr{C}$. Let $\mathscr{A}_0$ be a set of ground atoms. Let $\mathscr{F}_0$ be the set of classical formulas formed from $\mathscr{A}_0$ and logical connectives $\{\vee, \wedge, \neg, \rightarrow\}$.

[1] There is no variable symbol appearing in the ground formula. For example, $user(John)$ is a ground atom, and $user(x)$ is not a ground atom.

ARTICLE IN PRESS

Let $\mathscr{G}$ be a set of ground formulas and universally quantified formulas (in which the quantifies are outermost) formed from $\mathscr{F}$. That is, if $\alpha \in \mathscr{F}$, and $X_1, \ldots, X_n$ are the free variables of α, then $\forall X_1, \ldots, \forall X_n \alpha \in \mathscr{G}$. Note that we use $\mathscr{G}$ to formulate requirements in this paper.

Let $\alpha \in \mathscr{G}$ be a first order formula and $\Delta \subseteq \mathscr{G}$ a set of formulas in $\mathscr{G}$. In this paper, we call Δ a set of requirements or a requirements collection (mostly referred to as a (partial) requirements specification) while each formula $\alpha \in \Delta$ represents a requirement.

For example, for a requirement: *if an authorized student requests to choose a course and the course is available, then the student can choose the course*, if we

- use $auth(X)$ to denote that X is authorized;
- use $requ(X, Y)$ to denote that X requests to choose Y;
- use $avai(Y)$ to denote that Y is available;
- use $choo(X, Y)$ to denote that X can choose Y;
- use variable *Stud* to denote a student;
- use variable *Cour* to denote a course;

Then we can represent the requirement by

$$\forall Stud \forall Cour\ (auth(Stud) \wedge requ(Stud, Cour) \wedge avai(Cour) \rightarrow choo(Stud, Cour)).$$

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 in a scenario:

- *Bob* is an authorized student ($auth(Bob)$);
- *Bob* applies to choose *Probability Theory* ($requ(Bob, Prob)$);
- *Probability Theory* is available ($avai(Prob)$);

Then we use the following ground formula as a substitute for the universally quantified formula above:

$$auth(Bob) \wedge requ(Bob, Prob) \wedge avai(Prob) \rightarrow choo(Bob, Prob).$$

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 $\alpha_1 \wedge \alpha_2 \wedge \cdots \wedge \alpha_n$ as a substitute for α in the scenario. Thus, we concentrate on the instantiated requirements collections in the rest of this paper. That is, we assume that all the requirements collections are subsets of $\mathscr{F}_0$. By this, we restrict the first order logical representation of requirements to the propositional case.

Based on these notations, we can formally define the logical contradiction (inconsistency) as follows:

Definition 2.1 (*Inconsistency*)**.** Let Δ be a set of requirements. Let $\vdash$ be the classical consequence relation. If there exists $\alpha \in \mathscr{F}_0$ such that $\Delta \vdash \alpha$ and $\Delta \vdash \neg\alpha$, then we call Δ inconsistent.

An inconsistent set Δ of requirements is denoted by $\Delta \vdash \bot$ if we abbreviate $\alpha \wedge \neg\alpha$ as $\bot$.

We use the following example to illustrate the inconsistency in requirements.

Example 2.1. Consider the following two requirements about choosing courses:

(r1) if an authorized student requests to choose a course and the course is available, then the student can choose the course;
(r2) if a student failed the prerequisite course of a course requested, then the student cannot choose the course.

We may use the following formulas to formulate the two requirements:

$$\forall Stud \forall Cour\ (auth(Stud) \wedge requ(Stud, Cour) \wedge avai(Cour) \rightarrow choo(Stud, Cour));$$
$$\forall Stud \forall Cour1 \forall Cour2 (requ(Stud, Cour1) \wedge prer(Cour2, Cour1) \wedge fail(Stud, Cour2)$$
$$\rightarrow \neg choo(Stud, Cour1)).$$

Moreover, we give the following facts in a scenario:

(f1) *Bob* is an authorized student ($auth(Bob)$);
(f2) *Bob* applies to choose *Probability Theory*($requ(Bob, Prob)$);
(f3) *Probability Theory* is available ($avai(Prob)$);
(f4) *Advanced Mathematics* is the prerequisite of *Probability Theory* ($prer(Adma, Prob)$);
(f5) *Bob* failed *Advanced Mathematics* ($fail(Bob, Adma)$).

Then we can get the following instantiated requirements in this scenario:

$$(r1)\ auth(Bob) \wedge requ(Bob, Prob) \wedge avai(Prob) \rightarrow choo(Bob, Prob);$$
$$(r2)\ requ(Bob, Prob) \wedge prer(Adma, Prob) \wedge fail(Bob, Adma) \rightarrow \neg choo(Bob, Prob).$$

Let $\Delta = \{(r1), (r2), (f1), f(2), f(3), f(4), f(5)\}$, then

$$\Delta \vdash choo(Bob, Prob), \text{ and } \Delta \vdash \neg choo(Bob, Prob),$$

i.e., Δ is inconsistent.

ARTICLE IN PRESS

2.1. The degree of inconsistency

Let Δ be a requirements collection. A *minimal inconsistent subset* of Δ, denoted Φ, is an inconsistent subset of Δ such that none of its proper subsets is inconsistent. Note that minimal inconsistent subsets are considered as the purest form of inconsistency for applications of syntax sensitive inconsistency handling, because one only needs to remove one formula from each minimal inconsistent subset to restore consistency [27]. In this sense, we may consider requirements in the union of all minimal inconsistent subsets of Δ as *problematical* ones, i.e., requirements involved in inconsistencies [9]. We use $\mathsf{CORE}(\Delta)$ to denote the set of all the requirements involved in inconsistencies in Δ. In contrast, a *maximal consistent subset* of Δ, denoted Γ, is a consistent subset of Δ such that no requirement in $\Delta - \Gamma$ can be added to it without creating an inconsistency. Roughly speaking, each maximal consistent subset of Δ may be considered as the reflection of one of many plausible views of the requirements collection [9]. Then the common subset of all maximal consistent subsets of Δ may be considered as the collection of all the *uncontroversial* requirements in Δ, i.e., the requirements free from minimal inconsistent subsets [9]. We call such requirements *free requirements*(*formulas*) in Δ, and use $\mathsf{FREE}(\Delta)$ to denote the set of all the free requirements in Δ [9].

We use $\mathsf{MI}(\Delta)$ and $\mathsf{MC}(\Delta)$ to denote the set of all the minimal inconsistent subsets of Δ and the set of all the maximal consistent subsets of Δ, respectively. That is,

$$\mathsf{MI}(\Delta) = \{\Phi \subseteq \Delta \mid \Phi \vdash \bot, \text{ and } \forall \alpha \in \Phi,\ \Phi \setminus \{\alpha\} \nvdash \bot\},$$
$$\mathsf{MC}(\Delta) = \{\Gamma \subseteq \Delta \mid \Gamma \nvdash \bot, \text{ and } \forall \alpha \in \Delta - \Gamma,\ \Gamma \cup \{\alpha\} \vdash \bot\}.$$

Then

$$\mathsf{FREE}(\Delta) = \bigcap_{\Gamma \in \mathsf{MC}(\Delta)} \Gamma, \text{ and } \mathsf{CORE}(\Delta) = \bigcup_{\Phi \in \mathsf{MI}(\Delta)} \Phi.$$

In particular, $\mathsf{CORE}(\Delta) = \Delta - \mathsf{FREE}(\Delta)$ [9]. Essentially, this means that we may divide a given inconsistent requirements collection Δ into two parts, i.e., $\mathsf{FREE}(\Delta)$ and $\mathsf{CORE}(\Delta)$.

Hunter's approach [14] to measuring the degree of inconsistency arising in a set of formulas based on scoring function can be stated as follows. For a set of formulas Δ, a scoring function S is defined from $\mathcal{P}(\Delta)$ (the power set of Δ) to the natural numbers so that for any $\Gamma \in \mathcal{P}(\Delta)$, $S(\Gamma)$ gives the number of minimal inconsistent subsets of Δ that would be eliminated if the subset Γ was removed from Δ [14]. That is, for $\Gamma \subseteq \Delta$,

$$S(\Gamma) = |\mathsf{MI}(\Delta)| - |\mathsf{MI}(\Delta - \Gamma)|.$$

As such, sets of formulas could be compared using their scoring functions so that an ordering relation $\leq$, which means *more inconsistent than*, over these sets can be defined [14]. That is, assume that Γ_i and Γ_j are of the same cardinality, S_i is the scoring function for Γ_i, and S_j the scoring function for Γ_j. $S_i \leq S_j$ holds iff there is a bijection $f : \mathcal{P}(\Gamma_i) \mapsto \mathcal{P}(\Gamma_j)$ such that the following condition can be satisfied:

$$\forall \Theta \in \mathcal{P}(\Gamma_i),\ S_i(\Theta) \leq S_j(f(\Theta)).$$

We say Γ_j is more inconsistent than Γ_i iff $S_i \leq S_j$ [14].

If we sort the scoring functions $\{S_i(\Theta) \mid \Theta \subseteq \Gamma_i\}$ and $\{S_j(\Theta') \mid \Theta' \subseteq \Gamma_j\}$, respectively, it is easy to find whether there is such a bijection.

Roughly speaking, the scoring function-based idea of measuring the inconsistency in Γ focuses on the contribution made by each subset of Γ to the inconsistency in Γ [14].

2.2. Logical representation of viewpoints

In the Viewpoints framework, a viewpoint is a description of concerns of a particular group of stakeholders. Let $\mathsf{Viewpoints} = \{v_1, \ldots, v_n\} (n \geq 2)$ be the set of viewpoints, then the requirements specification could be represented by an n-tuple $\mathcal{R} = \langle \Delta_1, \ldots, \Delta_n \rangle$, where Δ_i is the set of requirements of viewpoint v_i, for $i = 1, \ldots, n$. We call each subset of Δ_i a requirements collection in viewpoint v_i. In contrast, we call $\bigcup_{i=1}^{n} \Gamma_i$ a *combined requirements collection* if for each $i\,(1 \leq i \leq n)$, Γ_i is a requirements collection in viewpoint v_i (i.e., a subset of Δ_i). It may be viewed as a combination of requirements collections in different viewpoints.

The relative importance of requirements has gained attention in requirements process [28,29]. For example, Davis argued that the relative importance of requirements is one of the most important knowledge for balancing the desired requirements, available budget, and the desired schedule in requirements triage [29]. Generally, the relative importance of a requirement is represented by the priority of the statements. One way for doing this is to prioritize requirements specifications based on numerical estimations of value, cost and risk of each requirement, such as cost-value approach [30] and quality function deployment [31]. However, Wiegers pointed out that few software organizations are willing to undertake such rigorous numerical approaches in his experience [28]. In contrast, a popular approach for prioritizing a requirements collection is to group requirements into several priority categories, such as three-level priority scheme of *High*, *Medium*, and *Low* and five-way priority scheme presented by Davis [29]. For the simplicity of discussion below, we assume that all the viewpoints use the same scale of priority categories. Furthermore, all the viewpoints have an agreement on the meaning of each level of priority.

Let m, a natural number, be the scale of priority categories, and L be $\left\{l_0^m, l_1^m, \ldots, l_{m-1}^m\right\}$, a totally ordered finite set of m symbolic values of the priorities, i.e. $l_i^m < l_j^m$ iff $i < j$. Furthermore, each symbolic value in L could be associated with a linguistic value. For example, for a three-level priority set [32,33], we have a totally ordered set L as $L = \{l_0^3, l_1^3, l_2^3\}$, where

$$l_0^3 : \mathit{Optional},\ l_1^3 : \mathit{Conditional},\ l_2^3 : \mathit{Essential}.$$

According to the convention in software engineering, the intuitive meaning of *essential* is that *the software product could not be acceptable unless all of the essential requirements are satisfied*; the meaning of *conditional* is that *these requirements would enhance the software product, but it is not unacceptable if absent*; the meaning of *optional* is that *these requirements may or may not be worthwhile* [32,33]. In all of the examples in this paper, we adopt this three-level priority set, though it is not obligatory.

For each viewpoint $v_i \in \mathsf{Viewpoints}$, prioritizing the requirements of Δ_i is in essence to establish prioritization, denoted P_i, a mapping from Δ_i to L by balancing the business value of requirements against its cost and technique risk [32]. Then the requirements specification with prioritization may be represented by $(\mathcal{R}, P) = \langle(\Delta_1, P_1), \ldots, (\Delta_n, P_n)\rangle$, where P_i is the prioritization of the viewpoint v_i.

As mentioned earlier, for each given viewpoint v_i, any requirements collection Γ in v_i is stratified into m strata by prioritization P_i. For each k $(0 \le k \le m-1)$, the k-th stratum of Γ, denoted by Γ^k, is a subset of Γ in which each requirement has the level of priority l_k^m. That is, $\Gamma^k = \{\alpha | \alpha \in \Gamma, P_i(\alpha) = l_k^m\}$. Obviously, $\Gamma = \bigcup_{k=0}^{m-1} \Gamma^k$ and $\Gamma^k \cap \Gamma^{k'} = \emptyset$, for any $k \neq k'$. We call n-tuple $\langle \Gamma^0, \ldots, \Gamma^{m-1}\rangle$ a priority-based partition of Γ under L [23].

2.3. The significance of requirements collections

Intuitively, the relative importance of a requirements collection in a given viewpoint depends on the relative importance of individual requirements in that collection. In our previous paper [23], we gave the following intuitive assumptions about significance of a requirements collection in a given viewpoint:

- the requirements with the same priority have the same significance;
- any requirement with a higher priority is more significant than all of those with lower priorities;
- those requirements with higher priorities play dominant roles in measuring the significance of a requirements collection.

Given a requirements collection of viewpoint v_i, denoted Γ, these assumptions imply that the number of requirements in each stratum of Γ plays a role in measuring the significance of Γ. We introduce the cardinality vector of a requirements collection in a given viewpoint (also termed as priority-based cardinality vector in [23]).

Definition 2.2 (*Cardinality Vector*). Given a viewpoint v_i, let L be an m-level priority set. $\forall \Gamma \subseteq \Delta_i$, the cardinality vector of Γ, denoted $\overrightarrow{C}_i(\Gamma)$, is defined as $\overrightarrow{C}_i(\Gamma) = (|\Gamma^0|, \ldots, |\Gamma^{m-1}|)$, where $\langle\Gamma^0, \ldots, \Gamma^{m-1}\rangle$ is the priority-based partition of Γ under L and $|\Gamma^k|$ is the cardinality of Γ^k, for $k = 0, \ldots, m-1$.

Essentially, for $\Gamma \subseteq \Delta_i$, the k-th component of $\overrightarrow{C}_i(\Gamma)$ is the number of the requirements with the k-th level of priority in viewpoint v_i. Then the cardinality vector $\overrightarrow{C}_i(\Gamma)$ provides the number of requirements in each stratum of Γ. Moreover, the relative location of $|\Gamma^k|$ in the vector $\overrightarrow{C}_i(\Gamma)$ reflects the relative priority of requirements in Γ^k. In this sense, $\overrightarrow{C}_i(\Gamma)$ gives a priority-based measure of significance of Γ from the perspective of v_i.

In order to compare two requirements collections in a given viewpoint, we assume the following lexicographical ordering relationship between cardinality vectors.

Definition 2.3 (*Lexicographical Ordering*, $\preceq$). Let $\mathbf{R}$ be a set of real numbers and $\mathbf{R}^m$ an m-dimensional space. For $A = (a_1, \ldots, a_m)$, $B = (b_1, \ldots, b_m) \in \mathbf{R}^m$, the lexicographical ordering, denoted $\preceq$, is defined as: $A \preceq B$ iff $\exists k$ such that

(a) $a_k \le b_k$, and
(b) $\forall i > k, a_i = b_i$.

Furthermore, $A \prec B$ iff $A \preceq B$ and $B \npreceq A$; $A = B$ iff $A \preceq B$ and $B \preceq A$.

The lexicographical ordering relationship may be easily generalized to the lexicographical ordering relationship between two $n \times m$ matrices. Suppose that $\mathbf{A} = (a_{ij})_{n\times m}$ and $\mathbf{B} = (b_{ij})_{n\times m}$. Let $A_i = (a_{i1}, \ldots, a_{in})$ and $B_i = (b_{i1}, \ldots, b_{in})$, the $\mathbf{A} \preceq \mathbf{B}$ iff $\exists k$ such that

(a) $A_k \preceq B_k$ and
(b) $\forall i > k, A_i = B_i$.

For the simplicity of discussion, we use formulas such as α and β to stand for any unspecified requirements statements in the examples in subsequent sections.

In the next section, we will provide an overview of an approach for measuring the significance of inconsistency in a requirements collection within a given viewpoint which was proposed in our previous paper [23]. And then in Section 4, we will elaborate an approach to systematically measuring the significance of inconsistency arising from different viewpoints.

3. Measuring significance of inconsistency in a given viewpoint

This section briefly introduces our previous work [23], which is for measuring the significance of inconsistency within one viewpoint.

Intuitively, for a given set of requirements Γ, the significance of the inconsistency in Γ is associated with the significance of $\mathsf{CORE}(\Gamma)$. However, the scoring function-based approaches to measuring inconsistency argued that the contribution to the inconsistency made by each subset of Γ should also be considered. For an individual subset of Γ, denoted Θ, its contribution to the inconsistency should be described by the reduction of requirements involved in inconsistency after the requirements of Θ were removed from Γ. Then we define the significance scoring vector function as follows:

Definition 3.1 (*Significance Scoring Vector Function [23]*). Given a viewpoint v_i, let L be an m-level priority set. $\forall \Gamma \subseteq \Delta_i$, let $\mathcal{P}(\Gamma)$ be the power set of Γ, then the significance scoring vector function for Γ, $\overrightarrow{Sc}_i : \mathcal{P}(\Gamma) \mapsto \mathbf{R}^m$, is defined such that $\forall \Theta \in \mathcal{P}(\Gamma)$,

$$\overrightarrow{Sc}_i(\Theta) = \overrightarrow{C}_i(\mathsf{CORE}(\Gamma)) - \overrightarrow{C}_i(\mathsf{CORE}(\Gamma - \Theta)).$$

Essentially, for $\Theta \in \mathcal{P}(\Gamma)$, $\overrightarrow{Sc}_i(\Theta)$ captures the reduction of the significance of those problematical statements in Γ from viewpoint v_i after Θ were removed from Γ.

Definition 3.2 (*Significance Ordering, $\preceq_i^S$ [23]*). Given a viewpoint v_i, let L be an m-level priority set. $\forall \Gamma_1, \Gamma_2 \subseteq \Delta_i$ and $|\Gamma_1| = |\Gamma_2|$, let $\overrightarrow{Sc}_i^1$ and $\overrightarrow{Sc}_i^2$ be the significance scoring vector functions under L for Γ_1 and Γ_2 respectively. Then $\overrightarrow{Sc}_i^1 \preceq_i^S \overrightarrow{Sc}_i^2$ holds iff there is a bijection $f : \mathcal{P}(\Gamma_1) \mapsto \mathcal{P}(\Gamma_2)$ such that the following condition can be satisfied:

$$\forall \Theta \subseteq \Gamma_1, \quad \overrightarrow{Sc}_i^1(\Theta) \preceq \overrightarrow{Sc}_i^2(f(\Theta)).$$

We call $\preceq_i^S$ the significance ordering. We also say the inconsistency in Γ_2 is *more significant than* that in Γ_1 in viewpoint v_i iff $\overrightarrow{Sc}_i^1 \preceq_i^S \overrightarrow{Sc}_i^2$. Furthermore, $\overrightarrow{Sc}_i^1 \prec_i^S \overrightarrow{Sc}_i^2$ iff $\overrightarrow{Sc}_i^1 \preceq_i^S \overrightarrow{Sc}_i^2$ and $\overrightarrow{Sc}_i^2 \not\preceq_i^S \overrightarrow{Sc}_i^1$; $\overrightarrow{Sc}_i^1 \simeq_i^S \overrightarrow{Sc}_i^2$ iff $\overrightarrow{Sc}_i^1 \preceq_i^S \overrightarrow{Sc}_i^2$ and $\overrightarrow{Sc}_i^2 \preceq_i^S \overrightarrow{Sc}_i^1$.

Actually, for Γ_1 and Γ_2, if the inconsistency in Γ_2 is more significant than that in Γ_1, then there is a bijection f from $\mathcal{P}(\Gamma_1)$ to $\mathcal{P}(\Gamma_2)$ such that for any subset of Γ_1, denoted Θ, the contribution to the inconsistency in Γ_2 made by $f(\Theta)$ is more significant than the contribution to the inconsistency in Γ_1 made by Θ.

Another particular thing is the constraint of $|\Gamma_1| = |\Gamma_2|$. It is only a necessary condition to construct a mapping from the subsets of Γ_1 to the subsets of Γ_2 conveniently. In practice, if this constraint is not satisfied, e.g., $|\Gamma_1| < |\Gamma_2|$, we may add some formulas being free from the inconsistency in Γ_1 to Γ_1 such that the constraint can be satisfied [14]. Actually, these formulas added to Γ_1 make no contribution to the significance of the original inconsistency in Γ_1.

Now we give the following example presented in [23] to show how to compare two inconsistent requirements collections in a given viewpoint in the sense of significance of inconsistency.

Example 3.1. Assume that $\Delta_1 = \{\alpha, \neg\alpha, \beta, \neg\beta\}$ and

$$P_1(\alpha) = l_0^3, \qquad P_1(\neg\alpha) = l_1^3, \qquad P_1(\beta) = l_1^3, \qquad P_1(\neg\beta) = l_2^3.$$

Consider $\Gamma_1 = \{\alpha, \neg\alpha\}$ and $\Gamma_2 = \{\beta, \neg\beta\}$. Let $\overrightarrow{Sc}_1^1$ and $\overrightarrow{Sc}_1^2$ be significance scoring vector functions for Γ_1 and Γ_2, respectively. Then

$$\overrightarrow{Sc}_1^1(\Gamma_1) = (1,1,0), \qquad \overrightarrow{Sc}_1^1(\{\alpha\}) = (1,1,0), \qquad \overrightarrow{Sc}_1^1(\{\neg\alpha\}) = (1,1,0)$$
$$\overrightarrow{Sc}_1^2(\Gamma_2) = (0,1,1), \qquad \overrightarrow{Sc}_1^2(\{\beta\}) = (0,1,1), \qquad \overrightarrow{Sc}_1^2(\{\neg\beta\}) = (0,1,1).$$

Therefore, $\overrightarrow{Sc}_1^1 \prec_1^S \overrightarrow{Sc}_1^2$, and we conclude that the inconsistency in Γ_2 is more significant than that in Γ_1 from the perspective of v_1.

4. Measuring significance of inconsistency in a combined requirements collection

The significance scoring vector function presented in [23] focuses on the significance of inconsistency from a given viewpoint. However, in the Viewpoints framework, many inconsistencies arise from overlaps of different viewpoints. The requirements involved in such an inconsistency may belong to different viewpoints. Moreover, for a shared requirement, different viewpoints may assign different local priorities to it. As a result, for the same inconsistency, different viewpoints may have different priority-based measures of its significance from their own perspectives, as shown in the following example.

Example 4.1. Let $\mathsf{Viewpoints} = \{v_1, v_2\}$. Consider $(\mathcal{R}, P) = \langle(\Delta_1, P_1), (\Delta_2, P_2)\rangle$, where

$$\Delta_1 = \{\alpha, \beta, \gamma\}, \qquad \Delta_2 = \{\neg\alpha, \neg\beta, \phi\};$$
$$P_1(\alpha) = l_0^3, \quad P_1(\beta) = l_1^3, \quad P_1(\gamma) = l_2^3; \quad P_2(\neg\alpha) = l_2^3, \quad P_2(\neg\beta) = l_1^3, \quad P_2(\phi) = l_2^3.$$

Then $\mathsf{MI}(\Delta_1 \cup \Delta_2) = \{\Gamma_1, \Gamma_2\}$, where $\Gamma_1 = \{\alpha, \neg\alpha\}$, $\Gamma_2 = \{\beta, \neg\beta\}$.

Intuitively, from the perspective of v_1 (resp. v_2), we should assign a priority lower than l_0^3 to each requirements statement of $\Delta_2 - \Delta_1$ (resp. $\Delta_1 - \Delta_2$). For a combined requirements collection $\Gamma \subseteq (\Delta_1 \cup \Delta_2)$, the cardinality vector of Γ in viewpoint v_i, $\overrightarrow{C}_i(\Gamma)$ could be defined as

$$\overrightarrow{C}_i(\Gamma) = (|\Gamma - (\Gamma \cap \Delta_i)|, |(\Gamma \cap \Delta_i)^0|, \ldots, |(\Gamma \cap \Delta_i)^2|),$$

where $\langle(\Gamma \cap \Delta_i)^0, \ldots, (\Gamma \cap \Delta_i)^2\rangle$ is the priority-based partition of $\Gamma \cap \Delta_i$ under L in viewpoint $v_i (i = 1, 2)$. We may also define the significance vector function for Γ in each viewpoint in the similar way.

Let $\overrightarrow{Sc}_1^1$ (resp. $\overrightarrow{Sc}_2^1$) and $\overrightarrow{Sc}_1^2$ (resp. $\overrightarrow{Sc}_2^2$) be the significance scoring vector functions for Γ_1 and Γ_2 from viewpoint v_1 (resp. v_2), respectively. Then

$$\begin{aligned}
&\overrightarrow{Sc}_1^1(\{\alpha, \neg\alpha\}) = (1, 1, 0, 0), && \overrightarrow{Sc}_1^2(\{\beta, \neg\beta\}) = (1, 0, 1, 0);\\
&\overrightarrow{Sc}_1^1(\{\alpha\}) = (1, 1, 0, 0), && \overrightarrow{Sc}_1^2(\{\beta\}) = (1, 0, 1, 0);\\
&\overrightarrow{Sc}_1^1(\{\neg\alpha\}) = (1, 1, 0, 0), && \overrightarrow{Sc}_1^2(\{\neg\beta\}) = (1, 0, 1, 0);\\
&\overrightarrow{Sc}_2^1(\{\alpha, \neg\alpha\}) = (1, 0, 0, 1), && \overrightarrow{Sc}_2^2(\{\beta, \neg\beta\}) = (1, 0, 1, 0);\\
&\overrightarrow{Sc}_2^1(\{\alpha\}) = (1, 0, 0, 1), && \overrightarrow{Sc}_2^2(\{\beta\}) = (1, 0, 1, 0);\\
&\overrightarrow{Sc}_2^1(\{\neg\alpha\}) = (1, 0, 0, 1), && \overrightarrow{Sc}_2^2(\{\neg\beta\}) = (1, 0, 1, 0).\\
&\overrightarrow{Sc}_1^1 \prec_1^S \overrightarrow{Sc}_1^2, && \overrightarrow{Sc}_2^2 \prec_2^S \overrightarrow{Sc}_2^1.
\end{aligned}$$

That is, from viewpoint v_1, inconsistency $\{\beta, \neg\beta\}$ is more significant than $\{\alpha, \neg\alpha\}$. In contrast, from viewpoint v_2, inconsistency $\{\alpha, \neg\alpha\}$ is more significant than $\{\beta, \neg\beta\}$.

As mentioned above, the disagreement between the two viewpoints on the significance of inconsistencies is due to different prioritization functions used in different viewpoints. To establish an acceptable common measurement of significance of inconsistency in a combined requirements collection to all the viewpoints, it is necessary to re-prioritize requirements from an integrated perspective.

4.1. Prioritizing requirements from the integrated perspective

Intuitively, to establish a global prioritization scheme over the requirements specification from a set of local priority levels within different viewpoints, the following factors should be taken into account [11]:

- the number of viewpoints that include a particular requirement statement;
- the relative importance of each viewpoint which the requirement statement belongs to;
- the relative importance of the requirement statement within the viewpoint that includes it.

Essentially, these are important concerns in evaluating the relative importance of an individual requirement from the global perspective.

Most of the current proposals for globally prioritizing requirements from multiple perspectives such as [34,35] concentrated on numerical (or weighted) priorities rather than qualitative priority levels used in this paper. However, in our previous paper [11], we have presented two approaches to globally prioritizing the requirements based on a set of local priority levels within different viewpoints, i.e., the merging-based approach and the priority vector-based approach. Roughly speaking, the merging-based approach considers each locally prioritized requirements collection as a stratified knowledge base, whilst some extension of the merged result of these stratified knowledge bases is considered as a globally prioritized requirements collection [11]. This kind of global prioritization always depends on the choice of merging operators. In contrast, the priority vector-based approach transforms each local priority level into a 0-1 vector (i.e., each element of this vector is either 0 or 1). Further, it transforms the process of global prioritization into some integrated operation over these vectors, which takes the factors above into account explicitly. Compared to transformation of qualitative levels into weighted values, the 0-1 vectors are more appropriate to articulating the nature of qualitative prioritization [11]. For these reasons, in this paper, we adopt a slight variation of the priority vector-based approach to construct a global or integrated prioritization over requirements based on the local priority of requirements within each viewpoint and the priority level of viewpoints. We start with the notation of supporting viewpoints of a given requirement defined in [11], i.e. the viewpoints which include the requirement.

Definition 4.1 (*Supporting Viewpoints*). Let $(\mathcal{R}, P)$ be a requirements specification comprising viewpoints $\{v_1, \ldots, v_n\}$ and $\Delta = \bigcup_{i=1}^{n} \Delta_i$. Let L be an m-level priority set. $\forall \alpha \in \Delta$, for $0 \leq k \leq m-1$, we define the set of supporting viewpoints of α with the priority l_k^m, denoted $Sup^k(\alpha)$, as follows:

$$Sup^k(\alpha) = \{v_i | \alpha \in \Delta_i, P_i(\alpha) = l_k^m\}.$$

We call $Sup(\alpha) = \bigcup_{k=0}^{m-1} Sup^k(\alpha)$ the set of supporting viewpoints of α. In particular, for each viewpoint $v_i (1 \leq i \leq n)$, we define the projection of $Sup^k(\alpha)$ onto v_i as

$$Sup_i^k(\alpha) = Sup^k(\alpha) \cap \{v_i\}.$$

Note that for the same requirements statement, different supporting viewpoints may assign it different priorities. On the other hand, as stated in [36,11], different stakeholders play different roles during the software system development, e.g., some stakeholders are more important than others. Therefore, it is not surprising that viewpoints are also prioritized during software development. Similar to prioritization of requirements, a common approach to prioritizing viewpoints is to group the set of viewpoints, Viewpoints, into several priority categories. Let L_V be an r-level priority set used in prioritizing the set of viewpoints. Then prioritizing viewpoints is to establish a prioritization P_V from Viewpoints to L_V in essence [11]. From now on, we call a set of viewpoints with different levels of priority a set of stratified viewpoints.

For each given requirements α, any set of supporting viewpoints $Sup^k(\alpha)$ is also stratified into r strata by prioritization P_V. Let

$$\langle Sup^{(0,k)}(\alpha), \ldots, Sup^{(r-1,k)}(\alpha)\rangle$$

be priority-based partition of $Sup^k(\alpha)$, then the j-th stratum of $Sup^k(\alpha)$ is $Sup^{(j,k)}(\alpha) = \{v_i \in Sup^k(\alpha) | P_V(v_i) = l_j^r\}$. That is, $Sup^{(j,k)}(\alpha)$ is a subset of supporting viewpoints of α with the priority l_k^m in which each viewpoint has been given the priority of level l_j^r.

Intuitively, the integrated prioritization, denoted $\mathbf{P_I}$, should be characterized by the following assumptions:

(a) Given a requirement α, the priority levels of its supporting viewpoints rather than the priority levels of α in these viewpoints play a dominating role in prioritizing α from the integrated perspective [11]. It is similar to the role of suits in playing cards when we are sorting the cards.

(b) If two requirements α and β have the same set of supporting viewpoints (i.e., $Sup(\alpha) = Sup(\beta)$), then β should take priority over α if $\exists i$ such that

$$(|Sup^{(i,0)}(\alpha)|, \ldots, |Sup^{(i,m-1)}(\alpha)|) \prec (|Sup^{(i,0)}(\beta)|, \ldots, |Sup^{(i,m-1)}(\beta)|),$$

and $\forall j > i$,

$$(|Sup^{(j,0)}(\alpha)|, \ldots, |Sup^{(j,m-1)}(\alpha)|) = (|Sup^{(j,0)}(\beta)|, \ldots, |Sup^{(j,m-1)}(\beta)|).$$

holds. Roughly speaking, if the viewpoints with higher priority levels prefer β to α, then we may consider that β is more important than α from the integrated perspective.

Then we define the integrated prioritization as follows:

Definition 4.2 (*Integrated Prioritization*, $\mathbf{P_I}$). Let $(\mathcal{R}, P)$ be a requirements specification comprising stratified viewpoints $\{v_1, \ldots, v_n\}$ and $\Delta = \bigcup_{i=1}^{n} \Delta_i$. Let L be an m-level priority set used in all the viewpoints. Let L_V be an r-level priority set used in prioritizing viewpoints. From the integrated perspective, the prioritization $\mathbf{P_I}$ is defined as $\mathbf{P_I} : \Delta \mapsto \mathbf{R}^{r \times m}$ such that $\forall \alpha \in \Delta$,

$$\mathbf{P_I}(\alpha) = \begin{bmatrix} |Sup^{(0,0)}(\alpha)| & \cdots & |Sup^{(0,m-1)}(\alpha)| \\ \vdots & \ddots & \vdots \\ |Sup^{(r-1,0)}(\alpha)| & \cdots & |Sup^{(r-1,m-1)}(\alpha)| \end{bmatrix}.$$

Furthermore, $\forall \alpha, \beta \in \Delta, \alpha \prec_{\mathbf{P_I}} \beta$ iff $\mathbf{P_I}(\alpha) \prec \mathbf{P_I}(\beta)$, and we say that β takes priority over α from the integrated perspective.

Note that $\mathbf{P_I}$ satisfies the assumptions (a) and (b) mentioned above. Moreover, the j-th row of $\mathbf{P_I}(\alpha)$ is $(|Sup^{(j,0)}(\alpha)|, \ldots, |Sup^{(j,m-1)}(\alpha)|)$. For each k, $Sup^{(j,k)}(\alpha)$ is a subset of supporting viewpoints with priority level l_j^v. Then the relative location of the j-th row in $\mathbf{P_I}(\alpha)$ reflects the relative importance of these supporting viewpoints. Moreover, the relative location of $|Sup^{(j,k)}(\alpha)|$ in the j-th row reflects the relative importance of α with regard to these viewpoints since $|Sup^{(j,k)}(\alpha)|$ is the number of viewpoints with l_j^r which assign the priority of level l_k^m to α. In summary, the relative location of $|Sup^{(j,k)}(\alpha)|$ in $\mathbf{P_I}(\alpha)$ reflects the relative importance of viewpoints in $Sup^{(j,k)}(\alpha)$ as well as the relative importance of α with regard to these viewpoints. Thus, $\mathbf{P_I}(\alpha)$ captures the relative importance of α from the integrated perspective.

In particular, suppose that Viewpoints $= \{v_i\}$, i.e., there is only one viewpoint v_i, then $\forall \alpha \in \Delta_i, \mathbf{P_I}(\alpha) = (|Sup_i^0(\alpha)|, \ldots, |Sup_i^{m-1}(\alpha)|)$, where $|Sup_i^k(\alpha)| = 1$ and $|Sup_i^l(\alpha)| = 0$ for all $l \neq k$ if and only if $P_i(\alpha) = l_k^m$. From this sense, we present an alternative representation of prioritization P_i (also termed the priority vector function of v_i in [11]), denoted P_i^*, as follows:

$$\forall \alpha \in \Delta_i, \quad P_i^*(\alpha) = (a_0, \ldots, a_{m-1}) \quad \text{iff} \quad P_i = l_k^m,$$

ARTICLE IN PRESS

where $a_k = 1$ and $a_j = 0$ for all $j \neq k$. By this, we transform the qualitative prioritization P_i into a 0-1 vector representation of priorities P_i^* over Δ_i. Moreover, we provide an alternative representation of prioritization P_V, denoted P_V^*, as follows:

$$P_V^*(v_i) = (b_0, \dots, b_r) \text{ iff } P_V(v_i) = l_k^r,$$

where $b_k = 1$ and $b_j = 0$ for all $j \neq k$. Then we can get the following result:

$$\mathbf{P_I}(\alpha) = \sum_{i\ s.t. \alpha \in v_i} \left(P_V^*(v_i)\right)^\tau P_i^*(\alpha),$$

where $\left(P_V^*(v_i)\right)^\tau$ is the transpose of $P_V^*(v_i)$. It shows that we can compute the integrated prioritization $\mathbf{P_I}$ from the prioritization P_V^* and the set of locally prioritization with regard to each viewpoint $\left\{P_i^* | 1 \leq i \leq n.\right\}$ directly.

We give an example to illustrate how to get the integrated prioritization.

Example 4.2. Let $\mathsf{Viewpoints} = \{v_1, v_2, v_3\}$. Consider

$$(\mathcal{R}, P) = \langle (\Delta_1, P_1), (\Delta_2, P_2), (\Delta_3, P_3) \rangle,$$

where $\Delta_1 = \{\alpha, \neg\beta\}$, $\Delta_2 = \{\beta, \gamma\}$, $\Delta_3 = \{\neg\alpha, \beta\}$, and

$$P_1(\alpha) = P_1(\neg\beta) = l_2^3,$$
$$P_2(\beta) = l_1^3, \qquad P_2(\gamma) = l_2^3,$$
$$P_3(\neg\alpha) = l_0^3, \qquad P_3(\beta) = l_2^3.$$

Suppose that $P_V(v_1) = l_1^3$ and $P_V(v_2) = P_V(v_3) = l_2^3$. Then

$$P_V^*(v_1) = (0, 1, 0), \qquad P_V^*(v_2) = P_V^*(v_3) = (0, 0, 1);$$
$$P_1^*(\alpha) = P_1^*(\neg\beta) = (0, 0, 1),$$
$$P_2^*(\beta) = (0, 1, 0), \qquad P_2^*(\gamma) = (0, 0, 1),$$
$$P_3^*(\neg\alpha) = (1, 0, 0), \qquad P_3^*(\beta) = (0, 0, 1).$$

Furthermore, we get the integrated prioritization $\mathbf{P_I}$ as follows:

$$\mathbf{P_I}(\alpha) = P_V^*(v_1)^\tau P_1^*(\alpha) = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} (0, 0, 1) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix};$$

$$\mathbf{P_I}(\beta) = P_V^*(v_2)^\tau P_2^*(\beta) + P_V^*(v_3)^\tau P_3^*(\beta) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix};$$

$$\mathbf{P_I}(\neg\alpha) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}; \qquad \mathbf{P_I}(\neg\beta) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}; \qquad \mathbf{P_I}(\gamma) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Thus

$$\alpha,\ \neg\beta\ \prec_{\mathbf{P_I}}\ \neg\alpha\ \prec_{\mathbf{P_I}}\ \gamma\ \prec_{\mathbf{P_I}}\ \beta.$$

That is, the priority of β is higher than that of others from the integrated perspective.

4.2. Measuring significance of inconsistency based on stratified viewpoints

Given a viewpoint v_i, for any $\Gamma \subseteq \Delta_i$, $\overrightarrow{C}_i(\Gamma)$ describes the significance of requirements collection Γ from the perspective of v_i. However, it is easy to get that $\overrightarrow{C}_i(\Gamma) = \sum_{\alpha \in \Gamma} P_i^*(\alpha)$. It implies that the relative importance of Γ with regard to v_i equals to the sum of the relative importance of requirements of Γ. In this sense, we define the significance matrix, denoted **SiM**, to describe the relative importance of a combined requirements collection with regard to the integrated perspective.

Definition 4.3 (*Significance Matrix*). Let $(\mathcal{R}, P)$ be a requirements specification comprising stratified viewpoints $\{v_1, \dots, v_n\}$ and $\Delta = \bigcup_{i=1}^n \Delta_i$. Let L be an m-level priority set used in all the viewpoints. Let L_V be an r-level priority set used in prioritizing viewpoints. $\forall \Gamma \subseteq \Delta$, the significance matrix of Γ, denoted $\mathbf{SiM}(\Gamma)$ is defined as follows:

$$\mathbf{SiM}(\Gamma) = \sum_{\alpha \in \Gamma} \mathbf{P_I}(\alpha),$$

where $\mathbf{P_I}$ is the integrated prioritization over Δ.

Given a combined requirements collection Γ, suppose that $\mathbf{SiM}(\Gamma) = (s_{ij})_{r\times m}$. Then $s_{ij} = \sum_{\alpha\in\Gamma} |sup^{(i-1,j-1)}(\alpha)|$. It describes the total number of viewpoints in the $(i-1)$-th stratum of Viewpoints that assigned l_j^m to at least one requirement in Γ. Moreover, the relative location of s_{ij} in $\mathbf{SiM}(\Gamma)$ reflects the relative importance of these viewpoints as well as the relative importance of the related requirements with regard to these viewpoints. Then the significance matrix can be used to describe the significance of the combined requirements collection.

Furthermore, for a given combined requirements collection Γ, we concentrate on the contribution made by each subset of Γ to the significance of requirements involved in the inconsistency in Γ, which constitutes a scoring function-based measurement of the significance of inconsistency in Γ from the integrated perspective.

Definition 4.4 (*Significance Scoring Matrix Function*)**.** Let $(\mathcal{R}, P)$ be a requirements specification comprising stratified viewpoints $\{v_1, \dots, v_n\}$ and $\Delta = \bigcup_{i=1}^n \Delta_i$. Let L be an m-level priority set used in all the viewpoints. Let L_V be an r-level priority set used in prioritizing viewpoints. $\forall\Gamma \subseteq \Delta$, the significance scoring matrix function for Γ under L and L_V, $\mathbf{ScM} : \mathcal{P}(\Gamma) :\mapsto \mathbf{R}^{r\times m}$ is defined as that

$$\forall\Theta \subseteq \Gamma, \qquad \mathbf{ScM}(\Theta) = \mathbf{SiM}(\mathsf{CORE}(\Gamma)) - \mathbf{SiM}(\mathsf{CORE}(\Gamma - \Theta)),$$

where **SiM** is the significance matrix.

Note that the significance scoring vector function $\overrightarrow{Sc}_i$ may be considered as a simple case of the significance scoring matrix function **ScM** when Viewpoints $= \{v_i\}$.

For a combined requirements collection $\Theta \subseteq \Gamma$, $\mathbf{ScM}(\Theta)$ is the reduction of significance of those requirements involved in the inconsistency after Θ were removed from Γ. It describes the contribution made by Θ to the significance of inconsistency in Γ from the integrated perspective. In particular, $\mathbf{ScM}(\{\alpha\})$ captures how important the requirements disengaged from inconsistency are if we abandon the requirement α. So it can be considered as a measurement of the blame or responsibility of α for the inconsistencies in Γ for each $\alpha \in \Gamma$.

The scoring function-based thought for comparing two inconsistent requirements collections requires that we concentrate on comparing the contributions made by subsets of Γ_1 and Γ_2 to the inconsistencies in Γ_1 and Γ_2, respectively. If we can find a bijection f from $\mathcal{P}(\Gamma_1)$ to $\mathcal{P}(\Gamma_2)$ such that for any subset of Γ_1, denoted Θ, its contribution to the significance of the inconsistency in Γ_1 is less than the contribution made by $f(\Theta)$ to that in Γ_2, we may consider the inconsistency in Γ_2 is more significant than that in Γ_1. Then we present the following ordering relationship for comparing two inconsistent combined requirements collections in the stratified viewpoints framework.

Definition 4.5 (*The Significance Ordering* $\preceq_I^S$)**.** Let $(\mathcal{R}, P)$ be a requirements specification comprising stratified viewpoints $\{v_1, \dots, v_n\}$ and $\Delta = \bigcup_{i=1}^n \Delta_i$. Let L be an m-level priority set used in all the viewpoints. Let L_V be an r-level priority set used in prioritizing viewpoints. $\forall\Gamma_1, \Gamma_2 \subseteq \Delta$ and $|\Gamma_1| = |\Gamma_2|$, let $\mathbf{ScM}^1$ and $\mathbf{ScM}^2$ be the significance scoring matrix functions under L and L_V for Γ_1 and Γ_2 respectively. Then $\mathbf{ScM}^1 \preceq_I^S \mathbf{ScM}^2$ holds iff there is a bijection $f : \mathcal{P}(\Gamma_1) \mapsto \mathcal{P}(\Gamma_2)$ such that the following condition can be satisfied:

$$\forall\Theta \subseteq \Gamma_1, \mathbf{ScM}^1(\Theta) \preceq \mathbf{ScM}^2(f(\Theta)).$$

We call $\preceq_I^S$ the significance ordering from the integrated perspective. We also say the inconsistency in Γ_2 is more significant than that in Γ_1 iff $\mathbf{ScM}^1 \preceq_I^S \mathbf{ScM}^2$. Furthermore, $\mathbf{ScM}^1 \prec_I^S \mathbf{ScM}^2$ iff $\mathbf{ScM}^1 \preceq_I^S \mathbf{ScM}^2$ and $\mathbf{ScM}^2 \not\preceq_I^S \mathbf{ScM}^1$; $\mathbf{ScM}^1 \simeq_I^S \mathbf{ScM}^2$ iff $\mathbf{ScM}^1 \preceq_I^S \mathbf{ScM}^2$ and $\mathbf{ScM}^2 \preceq_I^S \mathbf{ScM}^1$.

Let us give an example to illustrate how to compare two inconsistent combined requirements collections in the stratified viewpoints framework.

Example 4.3. Let Viewpoints $= \{v_1, v_2, v_3\}$ be a set of stratified viewpoints. Consider $(\mathcal{R}, P) = \langle(\Delta_1, P_1), (\Delta_2, P_2), (\Delta_3, P_3)\rangle$, as the requirements specification of Viewpoints, where

$$\Delta_1 = \{\alpha, \beta, \neg\gamma\}, \qquad \Delta_2 = \{\alpha, \beta \to \gamma, \varphi\}, \qquad \Delta_3 = \{\beta, \alpha \to \neg\varphi, \neg\gamma\};$$
$$P_1(\alpha) = P_1(\beta) = l_2^3, \quad P_1(\neg\gamma) = l_1^3; \quad P_2(\alpha) = l_2^3, \quad P_2(\beta \to \gamma) = l_1^3, \quad P_2(\varphi) = l_2^3;$$
$$P_3(\beta) = l_1^3, \quad P_3(\alpha \to \neg\varphi) = l_2^3, \quad P_3(\neg\gamma) = l_1^3;$$
$$P_V(\{v_1\}) = l_2^3, \quad P_V(\{v_2\}) = l_2^3, \quad P_V(\{v_3\}) = l_1^3.$$

Then the integrated prioritization $\mathbf{P_I}$ is:

$$\mathbf{P_I}(\alpha) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 2 \end{bmatrix}, \qquad \mathbf{P_I}(\beta) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{P_I}(\neg\gamma) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix},$$

$$\mathbf{P_I}(\varphi) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{P_I}(\alpha \to \neg\varphi) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \qquad \mathbf{P_I}(\beta \to \gamma) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

For inconsistent combined requirements collections $\Gamma_1 = \{\alpha, \alpha \to \neg\varphi, \varphi\}$ and $\Gamma_2 = \{\beta, \beta \to \gamma, \neg\gamma\}$, let $\mathbf{ScM}^1$ and $\mathbf{ScM}^2$ be the significance scoring matrix functions under L for Γ_1 and Γ_2 respectively. Then

$$\mathbf{ScM}^1(\{\alpha, \alpha \to \neg\varphi, \varphi\}) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 3 \end{bmatrix}; \quad \mathbf{ScM}^2(\{\beta, \beta \to \gamma, \neg\gamma\}) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 2 & 1 \end{bmatrix};$$

$$\mathbf{ScM}^1(\{\alpha, \alpha \to \neg\varphi\}) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 3 \end{bmatrix}; \quad \mathbf{ScM}^2(\{\beta, \beta \to \gamma\}) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 2 & 1 \end{bmatrix};$$

$$\mathbf{ScM}^1(\{\alpha, \varphi\}) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 3 \end{bmatrix}; \quad \mathbf{ScM}^2(\{\beta, \neg\gamma\}) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 2 & 1 \end{bmatrix};$$

$$\mathbf{ScM}^1(\{\alpha \to \neg\varphi\}) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 3 \end{bmatrix}; \quad \mathbf{ScM}^2(\{\beta \to \gamma, \neg\gamma\}) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 2 & 1 \end{bmatrix};$$

$$\mathbf{ScM}^1(\{\alpha\}) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 3 \end{bmatrix}; \quad \mathbf{ScM}^2(\{\beta\}) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 2 & 1 \end{bmatrix};$$

$$\mathbf{ScM}^1(\{\alpha \to \neg\varphi\}) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 3 \end{bmatrix}; \quad \mathbf{ScM}^2(\{\beta \to \gamma\}) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 2 & 1 \end{bmatrix};$$

$$\mathbf{ScM}^1(\{\varphi\}) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 3 \end{bmatrix}; \quad \mathbf{ScM}^2(\{\neg\gamma\}) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 2 & 1 \end{bmatrix};$$

$$\mathbf{ScM}^2 \preceq_I^S \mathbf{ScM}^1.$$

Thus, inconsistency in Γ_1 is more significant than that in Γ_2.

As shown by the following propositions, the significance scoring matrix function is a concise and expressive articulation of inconsistencies that arise in a combined requirements collection in the context of significance.

If a combined requirements collection Γ is consistent, any rational measurement of the significance of the inconsistency in Γ should be zero. The proposition below shows that the significance scoring matrix function satisfies this intuition characteristic.

Proposition 4.1. *Let $(\mathcal{R}, P)$ be a requirements specification comprising stratified viewpoints $\{v_1, \ldots, v_n\}$ and $\Delta = \bigcup_{i=1}^n \Delta_i$. Let L be an m-level priority set used in all the viewpoints. Let L_V be an r-level priority set used in prioritizing viewpoints. $\forall \Gamma \subseteq \Delta$, if* **ScM** *is the significance scoring matrix function for Γ under L and L_V, then*

$$\Gamma \text{ is consistent if and only if } \forall \Theta \subseteq \Gamma,\ \mathbf{ScM}(\Theta) = \mathbf{0},$$

where **0** *is the zero matrix.*

Proof. Suppose first that $\Gamma \subseteq \Delta$ is consistent. Then

$$\mathsf{CORE}(\Gamma) = \mathsf{CORE}(\Gamma - \Theta) = \emptyset$$

for any $\Theta \subseteq \Gamma$, so that

$$\forall \Theta \subseteq \Gamma,\ \mathbf{ScM}(\Theta) = \mathbf{SiM}(\mathsf{CORE}(\Gamma)) - \mathbf{SiM}(\mathsf{CORE}(\Gamma - \Theta)) = \mathbf{0}.$$

Second, suppose that $\forall \Theta \subseteq \Gamma,\ \mathbf{ScM}(\Theta) = \mathbf{0}$ and that Γ is not consistent. Let $\Theta^* = \mathsf{CORE}(\Gamma)$. Then $\Theta^* \neq \emptyset$ and $\Gamma - \Theta^* = \mathsf{FREE}(\Gamma)$ is consistent. Thus,

$$\mathbf{0} \prec \mathbf{SiM}(\Theta^*), \quad \text{and} \quad \mathsf{CORE}(\Gamma - \Theta^*) = \emptyset.$$

Moreover,

$$\mathbf{ScM}(\Theta^*) = \mathbf{SiM}(\mathsf{CORE}(\Gamma)) - \mathbf{SiM}(\mathsf{CORE}(\Gamma - \Theta^*)) = \mathbf{SiM}(\Theta^*) \succ \mathbf{0}.$$

This contradicts $\mathbf{ScM}(\Theta^*) = \mathbf{0}$, and so Γ is consistent. □

Intuitively, if a requirement α is involved in the inconsistency in Γ, then the singleton $\{\alpha\}$ should make material (i.e. non-zero) contribution to the significance of inconsistency. The following proposition shows that the significance scoring matrix function is characterized by this assumption.

Proposition 4.2. *Let $(\mathcal{R}, P)$ be a requirements specification comprising stratified viewpoints $\{v_1, \ldots, v_n\}$ and $\Delta = \bigcup_{i=1}^{n} \Delta_i$. Let L be an m-level priority set used in all the viewpoints. Let L_V be an r-level priority set used in prioritizing viewpoints. $\forall \Gamma \subseteq \Delta$, if* **ScM** *is the significance scoring matrix function for Γ under L and L_V, then $\forall \alpha \in \Gamma$,*

$$\alpha \in \mathsf{CORE}(\Gamma) \quad \textit{if and only if} \quad \mathbf{0} \prec \mathbf{ScM}(\{\alpha\});$$
$$\alpha \in \mathsf{FREE}(\Gamma) \quad \textit{if and only if} \quad \mathbf{ScM}(\{\alpha\}) = \mathbf{0},$$

where **0** *is the zero matrix.*

Proof. "$\Longrightarrow$" This part can be proved easily from the definition of significance scoring matrix function.

"$\Longleftarrow$" $\forall \alpha \in \Delta$, if $\mathbf{0} \prec \mathbf{ScM}(\{\alpha\})$, then

$$\mathbf{SiM}(\mathsf{CORE}(\Gamma - \{\alpha\})) \prec \mathbf{SiM}(\mathsf{CORE}(\Gamma)),$$

So, $\mathsf{CORE}(\Gamma - \{\alpha\}) \subset \mathsf{CORE}(\Gamma)$, that implies $\alpha \in \mathsf{CORE}(\Gamma)$.

If $\mathbf{ScM}(\{\alpha\}) = \mathbf{0}$, then

$$\mathbf{SiM}(\mathsf{CORE}(\Gamma - \{\alpha\})) = \mathbf{SiM}(\mathsf{CORE}(\Gamma)),$$

So, $\mathsf{CORE}(\Gamma - \{\alpha\}) = \mathsf{CORE}(\Gamma)$, that implies $\alpha \in \mathsf{FREE}(\Gamma)$. $\square$

Furthermore, given a combined requirements collection $\Theta \subseteq \Gamma$, the two propositions together demonstrate that $\mathbf{0} \prec \mathbf{ScM}(\Theta)$ means at least one requirement of Θ being involved in the inconsistency.

We argue that the significance of an inconsistency is associated with the relative importance of the requirements involved in the inconsistency earlier. The following proposition shows the relationship between the significance of the inconsistency in Γ and the significance of the set of requirements involved in the inconsistency.

Proposition 4.3. *Let $(\mathcal{R}, P)$ be a requirements specification comprising stratified viewpoints $\{v_1, \ldots, v_n\}$ and $\Delta = \bigcup_{i=1}^{n} \Delta_i$. Let L be an m-level priority set used in all the viewpoints. Let L_V be an r-level priority set used in prioritizing viewpoints. $\forall \Gamma_1, \Gamma_2 \subseteq \Delta$ and $|\Gamma_1| = |\Gamma_2|$, let $\mathbf{ScM}^1$ and $\mathbf{ScM}^2$ be the significance scoring matrix functions under L and L_V for Γ_1 and Γ_2 respectively. Then $\mathbf{ScM}^1 \preceq_I^S \mathbf{ScM}^2$ implies $\mathbf{SiM}(\mathsf{CORE}(\Gamma_1)) \preceq \mathbf{SiM}(\mathsf{CORE}(\Gamma_2))$. But the converse does not hold.*

Proof. Suppose that $\mathbf{ScM}^1 \preceq_I^S \mathbf{ScM}^2$. Then there is a bijection $f : \mathcal{P}(\Gamma_1) \mapsto \mathcal{P}(\Gamma_2)$ such that

$$\forall \Theta \subseteq \Gamma_1, \quad \mathbf{ScM}^1(\Theta) \preceq \mathbf{ScM}^2(f(\Theta)).$$

Let $\Theta^* = \mathsf{CORE}(\Gamma_1)$. Then

$$\mathbf{SiM}(\mathsf{CORE}(\Gamma_1)) = \mathbf{ScM}^1(\Theta^*) + \mathbf{SiM}(\mathsf{CORE}(\Gamma_1 - \Theta^*)) = \mathbf{ScM}^1(\Theta^*) \preceq \mathbf{ScM}^2(f(\Theta^*)).$$

On the other hand,

$$\mathbf{ScM}^2(f(\Theta^*)) = \mathbf{SiM}(\mathsf{CORE}(\Gamma_2)) - \mathbf{SiM}(\mathsf{CORE}(\Gamma_2 - f(\Theta^*))) \preceq \mathbf{SiM}(\mathsf{CORE}(\Gamma_l)).$$

So, $\mathbf{SiM}(\mathsf{CORE}(\Gamma_1)) \preceq \mathbf{SiM}(\mathsf{CORE}(\Gamma_2))$. $\square$

Example 4.4 (*A Counterexample For The Converse*). Assume that

$$\mathbf{P_I}(\alpha) = \mathbf{P_I}(\alpha \to \beta) = \mathbf{P_I}(\phi) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix};$$

$$\mathbf{P_I}(\neg\beta) = \mathbf{P_I}(\gamma) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}; \qquad \mathbf{P_I}(\neg\gamma) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Consider $\Gamma_1 = \{\alpha, \alpha \to \beta, \neg\beta\}$ and $\Gamma_2 = \{\gamma, \neg\gamma, \phi\}$. Let $\mathbf{ScM}^1$ and $\mathbf{ScM}^2$ be significance scoring matrix functions for Γ_1 and Γ_2, respectively. Then

$$\mathbf{SiM}(\mathsf{CORE}(\Gamma_1)) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 2 \\ 0 & 1 & 0 \end{bmatrix} \prec \mathbf{SiM}(\mathsf{CORE}(\Gamma_2)) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

Moreover, $\forall \Theta \subseteq \Gamma_1$, $\Theta \neq \emptyset$,

$$\mathbf{ScM}^1(\Theta) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 2 \\ 0 & 1 & 0 \end{bmatrix}.$$

But

$$\mathbf{ScM}^2(\{\phi\}) = \mathbf{0}.$$

Thus,

$$\mathbf{ScM}^1 \not\preceq_I^S \mathbf{ScM}^2$$

The proposition shows the significance of the inconsistency in Γ reflects the relative importance of requirements involved in the inconsistency in some sense. But the significance of the inconsistency is not equal to the significance of $\mathsf{CORE}(\Gamma)$. The significance ordering $\preceq_I^S$ is more strict.

However, the significance ordering concentrates on the uncontroversial requirements as well as the requirements involved in the inconsistency. The next proposition shows in part how a significance ordering from the integrated perspective can be also viewed as a reflection of the numbers of requirements being free from inconsistency.

Proposition 4.4. *Let $(\mathcal{R}, P)$ be a requirements specification comprising stratified viewpoints $\{v_1, \dots, v_n\}$ and $\Delta = \bigcup_{i=1}^{n} \Delta_i$. Let L be an m-level priority set used in all the viewpoints. Let L_V be an r-level priority set used in prioritizing viewpoints. $\forall \Gamma_1, \Gamma_2 \subseteq \Delta$ and $|\Gamma_1| = |\Gamma_2|$, let $\mathbf{ScM}^1$ and $\mathbf{ScM}^2$ be the significance scoring matrix functions under L and L_V for Γ_1 and Γ_2 respectively. Then $\mathbf{ScM}^1 \preceq_I^S \mathbf{ScM}^2$ implies $|\mathsf{FREE}(\Gamma_1)| \geq |\mathsf{FREE}(\Gamma_2)|$. But the converse does not hold.*

Proof. Suppose that $\mathbf{ScM}^1 \preceq_I^S \mathbf{ScM}^2$. Then there is a bijection $f : \mathcal{P}(\Gamma_1) \mapsto \mathcal{P}(\Gamma_2)$ such that

$$\forall \Theta \subseteq \Gamma_1, \quad \mathbf{ScM}^1(\Theta) \preceq \mathbf{ScM}^2(f(\Theta)).$$

Consider $\forall \alpha \in \mathsf{FREE}(\Gamma_2)$, then $\mathbf{ScM}^2(\{\alpha\}) = \mathbf{0}$, moreover $f^{-1}(\alpha) \in \Gamma_1$ such that

$$\mathbf{ScM}^1(\{f^{-1}(\alpha)\}) \preceq \mathbf{ScM}^2(\{\alpha\}),$$

Thus,

$$\mathbf{ScM}^1(\{f^{-1}(\alpha)\}) = \mathbf{0},$$

that is, $f^{-1}(\alpha) \in \mathsf{FREE}(\Gamma_1)$. Therefore, $|\mathsf{FREE}(\Gamma_1)| \geq |\mathsf{FREE}(\Gamma_2)|$. □

The proposition shows that the number of uncontroversial requirements also has some impact on the significance ordering. As shown by Proposition 4.2, for each $\alpha \in \mathsf{FREE}(\Gamma)$, $\{\alpha\}$ makes no material contribution to the significance of the inconsistency because of $\mathbf{ScM}(\{\alpha\}) = \mathbf{0}$. In this sense, $|\mathsf{FREE}(\Gamma_1)| \geq |\mathsf{FREE}(\Gamma_2)|$ is necessary to ensure that $\mathbf{ScM}^1 \preceq_I^S \mathbf{ScM}^2$ holds.

Example 4.5 (*A Counterexample For The Converse*)**.** Let $\mathsf{Viewpoints} = \{v_1, v_2\}$ be a set of viewpoints. Consider $(\mathcal{R}, P) = \langle(\Delta_1, P_1), (\Delta_2, P_2)\rangle$ as the requirements specification of $\mathsf{Viewpoints}$, where

$$\Delta_1 = \{\alpha, \beta, \alpha \to \neg\gamma\}, \qquad \Delta_2 = \{\neg\alpha, \beta, \gamma\},$$
$$P_V(v_1) = l_2^3, \qquad P_V(v_2) = l_1^3;$$
$$P_1(\alpha) = l_2^3, \qquad P_1(\beta) = l_2^3, \qquad P_1(\alpha \to \neg\gamma) = l_1^3;$$
$$P_2(\neg\alpha) = l_2^3, \qquad P_2(\beta) = l_2^3, \qquad P_2(\gamma) = l_1^3.$$

$\Gamma_2 = \{\alpha, \neg\alpha, \beta\}$ and $\Gamma_1 = \{\alpha \to \neg\gamma, \gamma, \alpha\}$ are two inconsistent combined requirements collections. Then

$$\mathsf{FREE}(\Gamma_1) = \{\beta\}, \qquad \mathsf{FREE}(\Gamma_2) = \emptyset,$$

and

$$\mathbf{SiM}^2(\mathsf{CORE}(\Gamma_1)) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \prec \mathbf{SiM}^1(\mathsf{CORE}(\Gamma_2)) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 2 \\ 0 & 0 & 2 \end{bmatrix};$$

Thus

$$|\mathsf{FREE}(\Gamma_1)| > |\mathsf{FREE}(\Gamma_2)|.$$

But

$$\mathbf{SiM}^1 \not\preceq_I^S \mathbf{SiM}^2$$

according to Proposition 4.3.

4.3. Computation of the significance scoring matrix function

We discuss some issues about the computation of the significance scoring matrix function and the scalability of this approach. Given a requirements specification $(\mathcal{R}, P)$ and a prioritization P_V over $\mathsf{Viewpoints}$, the computation of the significance scoring matrix function for any combined requirements collection Γ may be divided into two sub-problems, i.e., the computation of the integrated prioritization $\mathbf{P_I}$ and the computation of $\mathsf{CORE}(\Gamma - \Theta)$ for each $\Theta \subseteq \Gamma$. As mentioned earlier, for each $\alpha \in \Gamma$, $\mathbf{P_I}(\alpha) = \sum_{i\ s.t. \alpha \in v_i} \left(P_V^*(v_i)\right)^{\tau} P_i^*(\alpha)$. It means that the integrated prioritization can be computed from the P_V and $\{P_i | 1 \leq i \leq n\}$ directly.

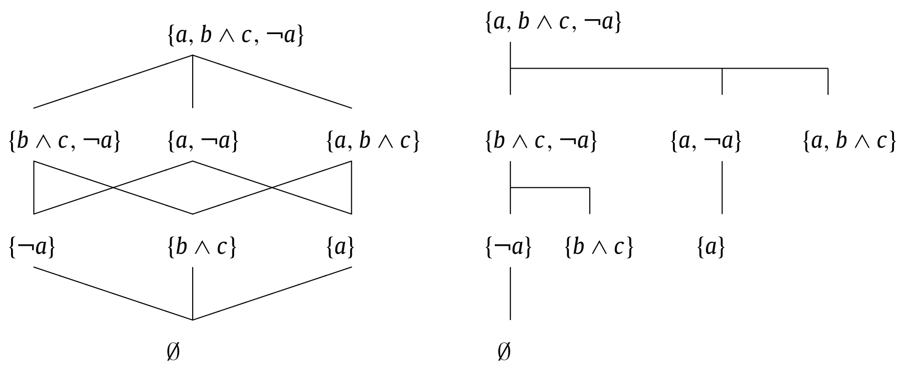


Fig. 1. Boolean lattice and its binomial tree for $\Gamma = \{a, b \wedge c, \neg a\}$.

With regard to the computation of $\mathsf{CORE}(\Gamma)$ given Γ, we may consider it as either a process of identification of all the maximal consistent subsets of Γ or a process of finding all the minimal inconsistent subsets of Γ. We will discuss these two possible methods, respectively.

However, for any subset of Γ, denoted Θ, both methods need to check whether Θ is consistent. Since we restrict the first order logical representation of requirements to the propositional case, consistency checking of Θ is a SAT problem. Although SAT is a NP-complete decision problem [37], many SAT solvers have been developed to solve the SAT problem efficiently for practical instances. For example, Conflict-Driven Clause Learning (CDCL for short) SAT Solvers are so effective that they have been successfully applied to many applications such as hardware and software model checking, planning, equivalence checking, bioinformatics, and cryptography [38]. Then we can utilize such an efficient SAT solver to check consistency of a requirements collection Θ.

The first method of computing $\mathsf{CORE}(\Gamma)$ checks whether Θ is maximal when Θ is consistent. Generally, to identify the maximal consistent subsets of Γ, each subset of Γ must be checked for consistency by a SAT solver. At worst, we need to check $2^{|\Gamma|}$ subsets. However, some important techniques for finding the maximal consistent subsets efficiently have been proposed [39,40]. We may utilize such techniques to find all the maximal consistent subsets of Γ. For example, the algorithm presented by Malouf [40] constructed a spanning tree of the Boolean lattice of subsets of Γ (which takes the form of binomial tree in [41]) firstly. Then a breadth-first search with root and leaf pruning of that binomial tree was performed for finding locally maximal consistent subsets [40]. Further, a final post-check for set inclusion can remove pseudo-maximal results from the set of locally maximal consistent subsets in their branch of the binomial tree [39,40]. As an important optimization presented in [39], the root pruning stated that a subtree rooted by a consistent subset can be pruned from the search space, because no subsets of the subset can be maximal. In contrast, the leaf pruning stated that a subtree can be skipped if the foot of that subtree is inconsistent [40]. In particular, it has been pointed out that keeping track of leftmost children allows us to avoid a substantial number of redundant consistency checks, moreover, as $|\Gamma|$ increases, leaf pruning can offer substantial improvements [40]. To illustrate this algorithm, consider $\Gamma = \{a, b \wedge c, \neg a\}$, then the Boolean lattice and binomial tree for Γ are shown in Fig. 1 according to [40,41]. Note that $\{b \wedge c, \neg a\}$ in Fig. 1 is consistent, then its subtree is pruned. The search finds that $\{b \wedge c, \neg a\}$, $\{a, b \wedge c\}$, and $\{a\}$ are the locally maximal consistent subsets. Note that $\{c\}$ is a pseudo-maximal consistent subset, and can be removed by final post-check.

Unfortunately, there is no explicit relationship between $\mathsf{MC}(\Gamma)$ and $\mathsf{MC}(\Gamma - \Theta)$ for an individual subset Θ. Consequently, to compute the scoring matrix function for Γ in terms of the maximal consistent subsets, we have to construct $2^{|\Gamma|}$ binomial trees. This makes the first method unpractical.

However, the good thing is that we have $\mathsf{MI}(\Gamma - \Theta) \subseteq \mathsf{MI}(\Gamma)$ and $\mathsf{MI}(\Gamma - \Theta) = \{\Phi \in \mathsf{MI}(\Gamma) | \Phi \cap \Theta = \emptyset\}$. Then if we take the second method to compute the scoring matrix function for Γ in terms of the minimal inconsistent subsets of Γ, we only need to construct a binomial tree corresponding to the inverse of the Boolean lattice.

The binomial tree corresponding to the inverse of the Boolean lattice is rooted by $\emptyset$ [41]. For example, the inverse of the Boolean lattice and the corresponding binomial tree for $\Gamma = \{a, b \wedge c, \neg a\}$ are shown in Fig. 2. Given a node of inconsistent subset Φ, there is no descendant of Φ that can be minimal since any subtree of Φ is rooted by a superset of Φ. Moreover, if the deepest leaf of a subset tree is consistent then there is no node in the tree that can be inconsistent. Then we adapt the root pruning and leaf pruning in [40] to the case as follows:

- root pruning: the branches rooted by an inconsistent subset can be pruned from the search space;
- leaf pruning: the subtrees whose deepest leaf is consistent can be skipped.

Suppose that $\Gamma = \{\gamma_1, \ldots, \gamma_{|\Gamma|}\}$, the algorithm of computing $\mathsf{MI}(\Gamma)$ may be described as follows:

Minimal_ Inconsistent_ Subsets (Γ)

```
1 $\mathrm{MI}(\Gamma) \longleftarrow \emptyset$
2 $\mathrm{Q} \longleftarrow \{\langle \emptyset, 0, \mathsf{FALSE} \rangle\}$
```

ARTICLE IN PRESS

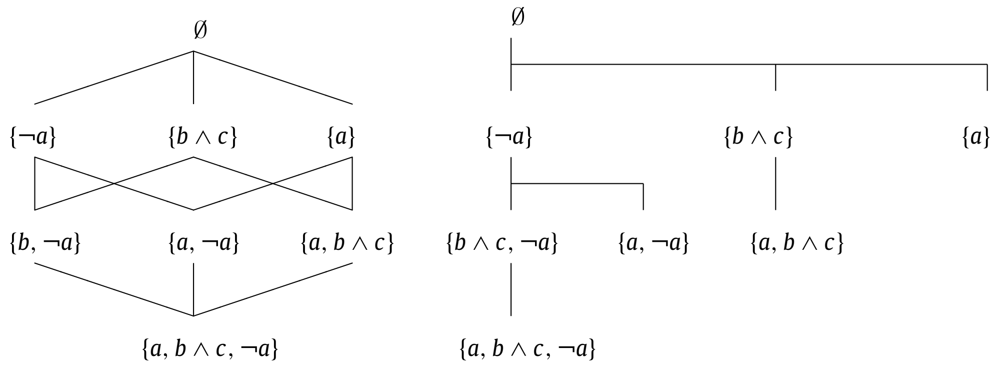


Fig. 2. The inverse of Boolean lattice and its binomial tree.

```
3  while $Q \neq \emptyset$
4    do $\langle \Phi, k, \mathsf{leftmost} \rangle \longleftarrow head(Q)$
5      if Inconsistent($\Phi$)
6        then if $\Phi \not\supset \Psi$ for all $\Psi$ in MI($\Gamma$)
7          then MI($\Gamma$) $\longleftarrow$ MI($\Gamma$) $\cup \{\Phi\}$
8      else $L \longleftarrow \Phi \cup \{\gamma_i : k < i \leq |\Gamma|\}$
9        if leftmost or Inconsistent(L)
10         then leftmost $\longleftarrow$ TRUE
11           for $i = k + 1$ to $|\Gamma|$
12             do ENQUEUE(Q, $\langle \Phi \cup \{\gamma_i\}, i, \mathsf{leftmost} \rangle$)
13               leftmost $\longleftarrow$ FALSE
14     DEQUEUE(Q)
15 return MI($\Gamma$)
```

This algorithm is a slight variation of the breadth-first search algorithm with root and leaf pruning presented by Malouf [40]. It coincides with the idea of inverting the search direction when $|\Gamma|$ is likely very large [40]. As usual, a queue is adopted to store unchecked subsets. The index *k* in $\langle \Phi, k, \mathsf{leftmost} \rangle$ is used to indicate the formula that was last added to Φ. The queue, together with the index *k*, allows us to keep track of unchecked subsets. Note that a flag leftmost is used to indicate whether that subset is the leftmost child. Because the deepest leaf node of the leftmost child is the same as the deepest leaf node of the parent in the binomial tree, then using the flag leftmost allows us to avoid a number of redundant inconsistency checks [40]. Of course, this algorithm also requires a post-check to remove pseudo-minimal subsets from MI(Γ). Note that Inconsistent(Φ) is a macrocommand. Its computational complexity depends on the underlying strategies used for checking whether Φ is inconsistent. Given a strategy for checking Inconsistent(Φ) with the maximal time cost T, the complexity of this algorithm is $O(2^{|\Gamma|} \cdot \mathsf{T})$ in the worst case in which no branch is pruned. In this sense, the efficiency of the **Minimal_ Inconsistent_ Subsets**(Γ) algorithm depends on the size of Γ, the concrete elements of Γ, and the efficiency of the selected SAT solver.

We use $\langle \Theta, \mathbf{ScM}(\Theta) \rangle$ to denote Θ and its significance scoring matrix function. Let $\mathbf{P_I}[i]$ be the integrated priority of γ_i for each *i*. Then we can describe an algorithm for computing the scoring matrix function for Γ as follows:

```
Scoring_ Matrix_ Function ($\Gamma$, $P_V$, $\{P_i\}_{i=1}^{n}$)
1  CORE($\Gamma$) $\longleftarrow \emptyset$
2  SiM(CORE($\Gamma$)) $\longleftarrow \mathbf{0}$
3  for $i = 1$ to $|\Gamma|$
4    do $\mathbf{P_I}[i] \longleftarrow \mathbf{0}$
5      for $j = 1$ to n
6        if $\gamma_i \in \Delta_j$ then $\mathbf{P_I}[i] = \mathbf{P_I}[i] + P^*_V[j] * P^*_j(\gamma_i)$
7  MI($\Gamma$) $\longleftarrow$ Minimal_ Inconsistent_ Subsets ($\Gamma$)
8  CORE($\Gamma$) $\longleftarrow \cup\Phi, \Phi \in$ MI($\Gamma$)
9  for $i = 1$ to $|\Gamma|$ do
10   if $\gamma_i \in$ CORE($\Gamma$) then SiM(CORE($\Gamma$)) $\longleftarrow$ SiM(CORE($\Gamma$)) + $\mathbf{P_I}[i]$
11 Q $\longleftarrow \{\langle \Gamma - \{\gamma_1\}, 1 \rangle\}$
12 ScM($\Gamma$) $\longleftarrow$ SiM(CORE($\Gamma$))
13 ScM $\longleftarrow \{\langle \Gamma, \mathbf{ScM}(\Gamma) \rangle\}$
14 while $Q \neq \emptyset$
15   do $\langle \Theta, k \rangle \longleftarrow head(Q)$
16     MI($\Gamma - \Theta$) $\longleftarrow \{\Phi \in$ MI($\Gamma$)$|\Phi \cap \Theta = \emptyset\}$
```

```
17      if $\mathsf{MI}(\Gamma - \Theta) \neq \emptyset$ then
18        $\mathsf{CORE}(\Gamma - \Theta) \longleftarrow \cup\Phi, \Phi \in \mathsf{MI}(\Gamma - \Theta)$
19        $\mathbf{SiM}(\mathsf{CORE}(\Gamma - \Theta)) \longleftarrow \mathbf{0}$
20        for $i = 1$ to $|\Gamma|$ do
21          if $\gamma_i \in \mathsf{CORE}(\Gamma - \Theta)$
22          then $\mathbf{SiM}(\mathsf{CORE}(\Gamma - \Theta)) \longleftarrow \mathbf{SiM}(\mathsf{CORE}(\Gamma - \Theta)) + \mathbf{P_I}[i]$
23        $\mathbf{ScM}(\Theta) \longleftarrow \mathbf{SiM}(\mathsf{CORE}(\Gamma)) - \mathbf{SiM}(\mathsf{CORE}(\Gamma - \Theta))$
24      else $\mathbf{ScM}(\Theta) \longleftarrow \mathbf{ScM}(\Gamma)$
25      $\mathbf{ScM} \longleftarrow \mathbf{ScM} \cup \{\langle\Theta, \mathbf{ScM}(\Theta)\rangle\}$
26      for $i = k + 1$ to $|\Gamma|$
27        do ENQUEUE(Q, $\langle\Theta - \{\gamma_i\}, k\rangle$)
28    DEQUEUE(Q)
29  return ScM
```

This algorithm consists of three parts, i.e., computation of integrated prioritization, computation of minimal inconsistent subsets, and computation of scoring matrix function. Note that the computational complexity of this algorithm is also $O(2^{|\Gamma|} \cdot \mathsf{T})$ in the worst case. This implies that the algorithm **Minimal_ Inconsistent_ Subsets**(Γ) for computing all the minimal inconsistent subsets of Γ plays an overwhelming role in computational complexity of an algorithm for computing the significance scoring matrix function for Γ.

Next we discuss the scalability of our approach to measuring the significance of inconsistency in requirements. The computational complexity of the algorithm for computing all the minimal inconsistent subsets seems to hinder the application of our approach to scalable requirements. However, such a higher computation complexity is exactly a description of the degree of difficulty of the problem of articulating inconsistency in nature. That is, it results from the problem itself we confront, i.e, the difficulty of checking whether a set of formulas is inconsistent and the difficulty of searching all the minimal inconsistent subsets. Nevertheless, combining the efficient SAT algorithms, the root and leaf pruning techniques in binomial tree, and scenario-based approach together provides a promising support for the application of our approach to a large set of requirements. Recalling that we formulate the problem of finding all minimal inconsistent subsets as a search problem in a binomial tree, in which each node is a subset of the set of requirements. First of all, the efficient SAT algorithms can provide an effective support checking inconsistency of a large set of requirements. For example, the CDCL (Conflict-Driven Clause Learning) SAT algorithms can solve instances with *hundreds of thousand* (propositional) variables and *tens of millions* of clauses, e.g., Siege can solve a problem with *0.25 million* (*propositional*) *variables* in less than 30 seconds [42]. Second, the techniques for optimization based on root and leaf pruning in a binomial tree have been found to be effective in such search problems in the binomial tree [40]. Third, we consider use of the scenario-based approach to handling inconsistency in requirements. Although the local consistency of a set of requirements with regard to a given scenario does not necessarily guarantee the global consistency of a requirements specification, the whole requirements specification must be inconsistent if a set of requirements is inconsistent with regard to a given scenario. Moreover, the scenario-based approach always involves only a part of requirements tightly related to the given scenario. In this sense, the scenario-based approach is considered as a practical way to checking inconsistency in the whole requirements specification as much as possible in the case of inconsistency checking with rather expensive computation cost [43].

4.4. Implementation of the tool IncMeasurer

To carry out the validation of algorithms presented above and to support the application of the approach for measuring the significance of inconsistency, we have implemented a tool termed IncMeasurer. This tool has been developed in Java 1.6.0, based on the algorithms of **Minimal_ Inconsistent_ Subsets** and **Scoring_Matrix_Function**. In addition, we make use of SAT4J library,[2] which provides an efficient library of SAT solvers in Java. Compared to the OpenSAT project, the SAT4J library facilitates the first-time users of SAT "black boxes", who want to embed SAT technologies into their application without worrying about the details.[3] This tool takes the following information as input:

- the scale of priority levels;
- each viewpoint and its priority level;
- scenarios for inconsistency checking;
- instantiated requirements with priority levels based on given scenarios.

As for the instantiated requirements in a given scenario, if an individual requirement r can be instantiated as $r_1, \ldots, r_n$ with regard to a given scenario, then we consider $r_1 \wedge \cdots \wedge r_n$ as an instantiation of r with regard to that scenario. For example, consider the requirement of "*if an authorized user requests to borrow a book and the book is available, then the user can borrow the book*", i.e.,

$$\forall User \forall Book\ (auth(User) \wedge requ(User, Book) \wedge avai(Book) \rightarrow borr(User, Book))$$

[2] https://wiki.objectweb.org/sat4j/.

[3] https://www.sat4j.org.

ARTICLE IN PRESS

and suppose that a scenario consists of the following facts

- *Alice* and *Bob* are two authorized users ($auth(Alice), auth(Bob)$);
- *Alice* applies to borrow "*software engineering*" ($requ(Alice, Soft\_eng)$);
- *Bob* applies to borrow "*requirements engineering*" ($requ(Bob, Requ\_eng)$);
- "*Software engineering*" and "*requirements engineering*" are available ($avai(Soft\_eng), avai(Requ\_eng)$);

Then the requirement above can be instantiated as follows:

$$(1) \quad auth(Alice) \wedge requ(Alice, Soft\_eng) \wedge avai(Soft\_eng) \rightarrow borr(Alice, Soft\_eng);$$

$$(2) \quad auth(Bob) \wedge requ(Bob, Requ\_eng) \wedge avai(Requ\_eng) \rightarrow borr(Bob, Requ\_eng).$$

And we consider

$$\begin{aligned} &(auth(Alice) \wedge requ(Alice, Soft\_eng) \wedge avai(Soft\_eng) \rightarrow borr(Alice, Soft\_eng)) \\ \wedge\ &(auth(Bob) \wedge requ(Bob, Requ\_eng) \wedge avai(Requ\_eng) \rightarrow borr(Bob, Requ\_eng)) . \end{aligned}$$

as corresponding instantiated requirement in the scenario.

Note that facts of scenarios and instantiated requirements are inputs in terms of Conjunctive Normal Form (CNF for short). Moreover, in measuring the significance of inconsistency for a requirements collection, we focus on the requirements involved in the inconsistency rather than the facts about the application domain. That is, we assume that all the scenarios provided by the user of this tool are developed correctly, and then facts in a given scenario should make no material contribution to the inconsistency in requirements in the scenario. Hence we would assume that the integrated priority of each fact is zero throughout our computation.[4]

This tool allows us to carry out the following tasks:

- checking consistency for a combined requirements collection;
- measuring the significance of inconsistency for a combined requirements collection, i.e., computing the significance scoring matrix function **ScM**;
- Comparing two combined requirements collections with the same cardinality in terms of significance of inconsistency.

Fig. 3 shows a screenshot of the editing window of IncMeasurer when comparing two requirements collections in terms of the significance of inconsistency.

By using this tool, we obtained the result of Example 4.3 in 15 ms on a Notebook PC equipped with a Mobile DualCore Intel Core Duo T2400 1.83 Ghz CPU and 2 G memory.

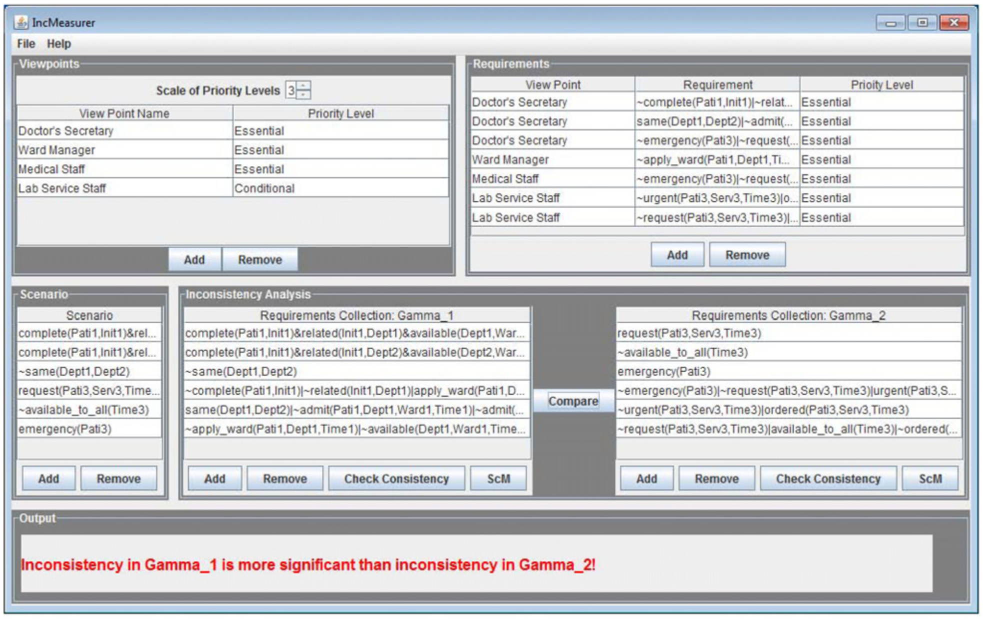
This tool facilitates the application of our inconsistency measurement to real requirements engineering to a great extent. First of all, this tool supports all the activities in a process of measuring inconsistency in terms of logical formulas, including consistency checking, measuring the significance of inconsistency in a set of requirements, and comparing two inconsistent sets of requirements with the same size. It covers the main tasks addressed by this paper. Second, the input of the tool is in the form of instantiated first order logic formulas. Actually, the first order logic can be considered as a preliminary of formal methods used in requirements engineering, such as in [9,6]. Requirements analysts with only preliminaries of the first order logic are able to use the tool easily. Third, there are special tools for translating requirements into logic formulas such as [6]. Then requirements analysts can combine or integrate our tool with some other available tools flexibly according to their needs in the analysis of inconsistency.

5. Case study

In order to validate our approach, we have so far provided some representative examples that fully illustrate the strength and utility of the integrated prioritization over requirements specification and the computation of the significance scoring matrix functions. Our approach (as well as the tool *IncMeasurer*) is intended for use in conjunction with other inconsistency management techniques in RE. It can be used to differentiate the inconsistencies by their relative importance in RE. We now present a more substantial case study from a hospital health record system to illustrate the strength as well as the utility of our formal techniques describe in previous sections.

Example 5.1. Consider an Electronic Health Record System (EHR System for short) for a hospital. We consider two important user tasks the system must support, i.e., Admit Patient Before Arrival and Clinical Session. We divide this case study into four parts. At first, we provide the description of requirements related to the two tasks in natural language. Then we formalize these requirements in terms of logical formulas. In the third part, we illustrate the utility of our approach and

[4] We assume the correctness of scenarios such that the illustration of the approach can be concentrated on the requirements. However, some scenarios may also be problematic in practical requirements development. In such cases, some facts are also involved in inconsistency and make material contribution to the inconsistency. And then we also need to prioritize facts in the scenario as well to measure the significance of inconsistency in requirements with regard to the scenario. In future work, we consider adaptation of our approach to the case of prioritized scenarios.



View Point Name	Priority Level
Doctor's Secretary	Essential
Ward Manager	Essential
Medical Staff	Essential
Lab Service Staff	Conditional

View Point	Requirement	Prioity Level
Doctor's Secretary	~complete(Pati1,Init1)|~relat...	Essential
Doctor's Secretary	same(Dept1,Dept2)|~admit(...	Essential
Doctor's Secretary	~emergency(Pati3)|~request(...	Essential
Ward Manager	~apply_ward(Pati1,Dept1,Ti...	Essential
Medical Staff	~emergency(Pati3)|~request(...	Essential
Lab Service Staff	~urgent(Pati3,Serv3,Time3)|o...	Essential
Lab Service Staff	~request(Pati3,Serv3,Time3)|...	Essential

Scenario
complete(Pati1,Init1)&rel...
complete(Pati1,Init1)&rel...
~same(Dept1,Dept2)
request(Pati3,Serv3,Time...
~available_to_all(Time3)
emergency(Pati3)

Requirements Collection: Gamma_1
complete(Pati1,Init1)&related(Init1,Dept1)&available(Dept1,War...
complete(Pati1,Init1)&related(Init1,Dept2)&available(Dept2,War...
~same(Dept1,Dept2)
~complete(Pati1,Init1)|~related(Init1,Dept1)|apply_ward(Pati1,D...
same(Dept1,Dept2)|~admit(Pati1,Dept1,Ward1,Time1)|~admit(...
~apply_ward(Pati1,Dept1,Time1)|~available(Dept1,Ward1,Time...

Requirements Collection: Gamma_2
request(Pati3,Serv3,Time3)
~available_to_all(Time3)
emergency(Pati3)
~emergency(Pati3)|~request(Pati3,Serv3,Time3)|urgent(Pati3,S...
~urgent(Pati3,Serv3,Time3)|ordered(Pati3,Serv3,Time3)
~request(Pati3,Serv3,Time3)|available_to_all(Time3)|~ordered(...

Inconsistency in Gamma_1 is more significant than inconsistency in Gamma_2!

Fig. 3. Snapshot of editing window of IncMeasurer.

its supporting tool IncMeasurer during the inconsistency analysis. Finally, we discuss some issues about application of our approach learned from this case study.

Part A. Descriptions of the two tasks

- Task 1: Admit Patient Before Arrival. This task creates an admission record or continues the admission process if it has been previously suspended (perhaps because some information is missing). The requirements for this task contain information such as the following:
  - Stakeholder : Doctor's Secretary

    (r1) On receipt of a call requiring an admission for a new patient with initial diagnosis and reservation department and time, a record of the patient shall be created.
    (r2) On receipt of a call requiring an admission for a patient in the system with initial diagnosis, the record of the patient shall be updated.
    (r3) If a patient's initial diagnosis information is judged to be incomplete, then the patient shall be put on the waiting list.
    (r4) On receipt of call providing missing information for a specific patient, updating the record of the patient and removing the patient from the waiting list.
    (r5) If a patient's initial diagnosis is complete, and there is no ward available for application time in the related department, or the department is irrelevant to the initial diagnosis, then the application for admission shall be transferred.
    (r6) If a patient's initial diagnosis is complete and the reservation department is related, then a ward with reservation time shall be applied for the patient.
    (r7) The same patient cannot be admitted to two different wards at the same time.

  - Stakeholder : Ward Manager

    (r8) On receipt of a request for a ward with reservation time and department, if the department has a ward available for application time, then the patient shall be admitted to the ward.
    (r9) If a patient is admitted to a specific ward, the patient shall be notified of the admission.

- Task 2: Clinical Session. A clinical session may comprise diagnosis, planning of treatment, etc. The requirements for this task contain the following information:
  - Stakeholder : Medical Staff

    (r10) Unconscious patients shall be monitored.
    (r11) Only surgeons in a department have rights to prescribe drugs in the department.

  - Stakeholder: Medical Staff, Doctor's Secretary

    (r12) If the deadline for a service is passed, a reminder shall be sent to the service provider.

Table 1
Variables and constants.

Variable	Explanation	Constant	Explanation
Pati	Patient	Add	The action of adding
Inti	Initial diagnosis	Delete	The action of deleting
Dept	Department	Prescribe	The action of prescribing
Time	Reservation time	Delay	The state of delay
Ward	Ward		
Doct	Doctor		
Serv	Service		
Resu	Result of service		
Misi	Missing information		

(r13) Services that can be given on the spot should be recommended to the patient under diagnosis.
(r14) Services for emergency patients shall be considered as urgent services.
(r15) Obtained results of a patient's ordered services as scheduled shall be sent to the patient.

– Stakeholder : Lab Service Staff

(r16) A service request shall be made if the reservation time is convenient for all parties involved.
(r17) Urgent services shall be ordered immediately.
(r18) If the reservation time is not available for at least one party, then the service cannot be ordered.

Note that Lab Service is often managed by an independent system in hospitals, including reservation of service, lab test, data analysis and so on. Moreover, allowing for the complexity of clinical affairs, the first version of EHR system tries to ensure that the requirements provided by Doctor's Secretary, Medical Staff, and Ward Manager will be satisfied. Then Doctor's Secretary, Medical Staff, and Ward Manager are essential stakeholders for EHR System. In contrast, Lab Service Staff is considered as a secondary stakeholder. Moreover,

- Doctor's Secretary considers requirements (r1)–(r7), (r12)–(r15) are *essential*.
- Ward Manager considers requirements (r8)–(r9) are *essential*.
- Medical Staff considers requirements (r10), (r11), (r13), and (r14) are *essential*, and requirements (r12) and (r15) are *conditional*.
- Lab Service staff states (r16)–(r18) are *essential*.

To check consistency of requirements of each task, we need to develop a scenario that describes an application case of performing the corresponding task. To address this, we develop the following scenarios S_1 and S_2 for consistency checking for requirements of Task 1 and Task 2, respectively. The scenario S_1 models a case about an individual patient with complete initial diagnosis requesting a ward, whilst the scenario S_2 describes a case about an individual emergent patient requesting a service. In detail,

- Scenario S_1:
  - An individual patient Pati_1 with complete initial diagnosis Init_1 requests a ward with reservation time Time_1;
  - The initial diagnosis Init_1 is related to the department Dept_1 as well as Dept_2;
  - An individual ward Ward_1 in Dept_1 is available for Time_1;
  - Another individual ward Ward_2 in Dept_2 is also available for Time_1;
  - Dept_1 and Dept_2 are two distinct departments.
- Scenario S_2:
  - An individual patient Pati_2 is an emergency patient;
  - An individual service Serv_2 with reservation Time_2 is requested for Pati_2;
  - Time_2 is not available to all parties.

Part B. Formalizing the requirements and related scenarios

From the requirements document, we can generate dataflow diagrams, use cases, and so on. In this paper, we will use logical representation to describe these requirements information. We use the variables and constants listed in Table 1, and the atoms listed in Table 2 to formalize these requirements. Table 3 lists the formalized requirements (r1)–(r18). Scenarios S_1 and S_2 are given by Table 4 and Table 5, respectively.

For the sake of simplicity, we use the labels such as (r1), (r2) and (r3) to denote the corresponding formulas. Let v_1, v_2, v_3, and v_4 be the viewpoints corresponding to Doctor's Secretary, Medical Staff, Ward Manager, and Lab Service Staff, respectively. Then the priority levels of viewpoints are given as follows:

$$P_V(v_1) = P_V(v_2) = P_V(v_3) = \text{essential}, \qquad P_V(v_4) = \text{conditional}.$$

And the corresponding requirements specification is given by

$$(\mathcal{R}, P) = \langle(\Delta_1, P_1), (\Delta_2, P_2), (\Delta_3, P_3), (\Delta_4, P_4)\rangle,$$

Table 2
Atoms.

Atom	Explanation
$call(Pati, Init, Dept, Time)$	A call requires an admission for a patient with initial diagnosis, reservation department and time.
$new(Pati)$	*Pati* is a new patient.
$unconscious(Pati)$	*Pati* is an unconscious patient.
$monitor(Pati)$	*Pati* should be monitored.
$under\_diagnosis(Pati)$	*Pati* is under diagnosis.
$emergency(Pati)$	*Pati* is a emergency patient.
$surgeon(Doct, Dept)$	*Doct* is a surgeon in *Dept*.
$has\_right(Doct, Dept, \text{Prescribe})$	*Doct* has right to prescribe drugs in *Dept*.
$create\_record(Pati)$	A record of *Pati* is created.
$update\_record(Pati)$	Updating the record of *Pati.*
$complete(Pati, init)$	*Pati*'s *Inti* is complete.
$complement(Pati, Misi)$	A call provides missing information *Misi* of *Pati.*
$waiting\_list(Pati, \text{Add})$	*Pati* is put on waiting list.
$waiting\_list(Pati, \text{Delete})$	Removing *Pati* from waiting list.
$related(Init, Dept)$	*Inti* relates to *Dept*.
$available(Dept, Ward, Time)$	There is available *Ward* for *Time* in *Depat*.
$apply\_ward(Pati, Dept, Time)$	Applying a ward with reservation *Time* in *Dept* for *Pati.*
$admit(Pati, Dept, Ward, Time)$	*Pati* is admitted to *Ward* with *Time* in *Dept*.
$same(Dept1, Dept2)$	*Dept*1 is the same as *Dept*2.
$transfer(Pati, Inti, Time)$	*Pati* with *Inti* and reservation *Time* is transferred to some other department.
$notify(Pati, Dept, Ward, Time)$	*Pati* is notified with *Dept*, *Ward,* and *Time.*
$ordered(Pati, Serv, Time)$	Ordered service *Serv* for *Pati* with reservation *Time.*
$overdue(Time)$	*Time* is overdue.
$remind\_provider(Serv, \text{Delay})$	A reminder of *Serv* being delayed is sent to service provider.
$on\_the\_spot(Serv)$	*Serv* can be given on the spot.
$recommend(Pati, Serv)$	*Serv* is recommended to *Pati.*
$request(Pati, Serv, Time)$	Requesting to order *Serv* for *Pati* with reservation *Time.*
$urgent(Pati, Serv, Time)$	*Serv* requested for *Pati* with reservation *Time* is a urgent service.
$obtained(Serv, Resu)$	The result *Resu* of *Serv* is obtained.
$send(Pati, Resu)$	*Resu* is sent to *Pati.*
$available\_to\_all(Time)$	*Time* is available for all parties.

Table 3
Requirements.

(r1)	$\forall Pati \forall Init \forall Dept \forall Time(call(Pati, Init, Dept, Time) \wedge new(Pati) \rightarrow create\_record(Pati));$
(r2)	$\forall Pati \forall Init \forall Dept \forall Time(call(Pati, Init, Dept, Time) \wedge \neg new(Pati) \rightarrow update\_record(Pati));$
(r3)	$\forall Pati \forall Init \forall Dept \forall Time(call(Pati, Init, Dept, Time) \wedge \neg complete(Pati, init) \rightarrow waiting\_list(Pati, \text{Add}));$
(r4)	$\forall Pati \forall Misi(complement(Pati, Misi) \rightarrow (update\_record(Pati) \wedge waiting\_list(Pati, \text{Delete})));$
(r5)	$\forall Pati \forall Init \forall Dept \forall Ward \forall Time(complete(Pati, Init) \wedge (\neg related(Init, Dept) \vee \neg available(Dept, Ward, Time)) \rightarrow transfer(Pati, Inti, Time));$
(r6)	$\forall Pati \forall Init \forall Dept \forall Ward \forall Time(complete(Pati, Init) \wedge related(Init, Dept) \rightarrow apply\_ward(Pati, Dept, Time));$
(r7)	$\forall Pati \forall Dept1 \forall Dept2 \forall Ward1 \forall Ward2 \forall Time(\neg same(Dept1, Dept2) \rightarrow (admit(Pati, Dept1, Ward1, Time) \rightarrow \neg admit(Pati, Dept2, Ward2, Time)));$
(r8)	$\forall Pati \forall Init \forall Dept \forall Ward \forall Time(apply\_ward(Pati, Dept, Time) \wedge available(Dept, Ward, Time) \rightarrow admit(Pati, Dept, Ward, Time));$
(r9)	$\forall Pati \forall Dept \forall Ward \forall Time(admit(Pati, Dept, Ward, Time) \rightarrow notify(Pati, Dept, Ward, Time));$
(r10)	$\forall Pati(unconscious(Pati) \rightarrow monitor(Pati));$
(r11)	$\forall Doct \forall Dept(surgeon(Doct, Dept) \rightarrow has\_right(Doct, Dept, \text{Prescribe}));$
(r12)	$\forall Pati \forall Serv \forall Time(ordered(Pati, Serv, Time) \wedge overdue(Time) \rightarrow remind\_provider(Serv, \text{Delay}));$
(r13)	$\forall Pati \forall Serv(under\_diagnosis(Pati) \wedge on\_the\_spot(Serv) \rightarrow recommend(Pati, Serv));$
(r14)	$\forall Pati \forall Serv \forall Time(emergency(Pati) \wedge request(Pati, Serv, Time) \rightarrow urgent(Pati, Serv, Time));$
(r15)	$\forall Pati \forall Serv \forall Time \forall Resu(ordered(Patient, Service, Time) \wedge \neg overdue(Time) \wedge obtained(Serv, Resu) \rightarrow send(Pati, Resu));$
(r16)	$\forall Pati \forall Serv \forall Time(request(Pati, Serv, Time) \wedge available\_to\_all(Time) \rightarrow ordered(Pati, Serv, Time));$
(r17)	$\forall Pati \forall Serv \forall Time(urgent(Pati, Serv, Time) \rightarrow ordered(Pati, Serv, Time));$
(r18)	$\forall Pati \forall Serv \forall Time(request(Pati, Serv, Time) \wedge \neg available\_to\_all(Time) \rightarrow \neg ordered(Pati, Serv, Time)).$

Table 4
Scenario S_1.

(s1)	$complete(\text{Pati}_1, \text{Init}_1) \wedge related(\text{Init}_1, \text{Dept}_1) \wedge available(\text{Dept}_1, \text{Ward}_1, \text{Time}_1);$
(s2)	$complete(\text{Pati}_1, \text{Init}_1) \wedge related(\text{Init}_1, \text{Dept}_2) \wedge available(\text{Dept}_2, \text{Ward}_2, \text{Time}_1);$
(s3)	$\neg same(\text{Dept}_1, \text{Dept}_2).$

Table 5
Scenario S_2.

(s4)	$\mathit{request}(\mathrm{Pati}_2, \mathrm{Serv}_2, \mathrm{Time}_2)$;
(s5)	$\neg \mathit{available\_to\_all}(\mathrm{Time}_2)$;
(s6)	$\mathit{emergency}(\mathrm{Pati}_2)$.

where Δ_i is the requirements of viewpoint v_i, and P_i is the prioritization of viewpoint i for each i. In detail,

$$\Delta_1 = \{(r1), (r2), (r3), (r4), (r5), (r6), (r7), r(12), r(13), (r14), (r15)\},$$
$$P_1((r1)) = P_1((r2)) = P_1((r3)) = P_1((r4)) = P_1((r5)) = P_1((r6)) = P_1((r7))$$
$$= P_1((r12)) = P_1((r13)) = P_1((r14)) = P_1((r15)) = \text{essential};$$
$$\Delta_2 = \{(r8), (r9)\},$$
$$P_2((r8)) = P_2((r9)) = \text{essential};$$
$$\Delta_3 = \{(r10), (r11), r(12), r(13), (r14), (r15)\},$$
$$P_3((r10)) = P_3((r11)) = P_3((r13)) = P_3((r14)) = \text{essential},$$
$$P_3((r12)) = P_3((r15)) = \text{conditional};$$
$$\Delta_4 = \{(r16), (r17), (r18)\},$$
$$P_4((r16)) = P_4((r17)) = P_4((r18)) = \text{essential}.$$

Let

$$R_1 = \{(r1), (r2), (r3), (r4), (r5), (r6), (r7), r(8), r(9)\} \quad \text{and}$$
$$R_2 = \{(r10), (r11), r(12), r(13), (r14), (r15), (r16), (r17), (r18)\},$$

then R_1 and R_2 are requirements corresponding to Task 1 and Task 2, respectively.

Part C. Inconsistency analysis

(C1) **Checking consistency of tasks**.
For Task 1, given scenario S_1, the requirements (r6), (r7), and (r8) can be instantiated (with regard to S_1) as follows:

(r6) $(\mathit{complete}(\mathrm{Pati}_1, \mathrm{Init}_1) \land \mathit{related}(\mathrm{Init}_1, \mathrm{Dept}_1) \to \mathit{apply\_ward}(\mathrm{Pati}_1, \mathrm{Dept}_1, \mathrm{Time}_1)) \land$
$(\mathit{complete}(\mathrm{Pati}_1, \mathrm{Init}_1) \land \mathit{related}(\mathrm{Init}_1, \mathrm{Dept}_2) \to \mathit{apply\_ward}(\mathrm{Pati}_1, \mathrm{Dept}_2, \mathrm{Time}_1))$;

(r7) $\neg \mathit{same}(\mathrm{Dept}_1, \mathrm{Dept}_2) \to (\mathit{admit}(\mathrm{Pati}_1, \mathrm{Dept}_1, \mathrm{Ward}_1, \mathrm{Time}_1)$
$\to \neg \mathit{admit}(\mathrm{Pati}_1, \mathrm{Dept}_2, \mathrm{Ward}_2, \mathrm{Time}_1))$;

(r8) $(\mathit{apply\_ward}(\mathrm{Pati}_1, \mathrm{Dept}_1, \mathrm{Time}_1) \land \mathit{available}(\mathrm{Dept}_1, \mathrm{Ward}_1, \mathrm{Time}_1) \to$
$\mathit{admit}(\mathrm{Pati}_1, \mathrm{Dept}_1, \mathrm{Ward}_1, \mathrm{Time}_1)) \land (\mathit{apply\_ward}(\mathrm{Pati}_1, \mathrm{Dept}_2, \mathrm{Time}_1)$
$\land \mathit{available}(\mathrm{Dept}_2, \mathrm{Ward}_2, \mathrm{Time}_1) \to \mathit{admit}(\mathrm{Pati}_1, \mathrm{Dept}_2, \mathrm{Ward}_2, \mathrm{Time}_1))$.

Then for scenario S_1, we consider

$$\Gamma_1 = \{(\mathrm{s1}), (\mathrm{s2}), (\mathrm{s3}), (\mathrm{r6}), (\mathrm{r7}), (\mathrm{r8})\}.$$

Furthermore, for Task 2, given scenario S_2, the requirements (r14), (r17), and (r18) can be instantiated (with regard to S_2) as follows:

(r14) $\mathit{emergency}(\mathrm{Pati}_2) \land \mathit{request}(\mathrm{Pati}_2, \mathrm{Serv}_2, \mathrm{Time}_2) \to \mathit{urgent}(\mathrm{Pati}_2, \mathrm{Serv}_2, \mathrm{Time}_2)$;

(r17) $\mathit{urgent}(\mathrm{Pati}_2, \mathrm{Serv}_2, \mathrm{Time}_2) \to \mathit{ordered}(\mathrm{Pati}_2, \mathrm{Serv}_2, \mathrm{Time}_2))$;

(r18) $\mathit{request}(\mathrm{Pati}_2, \mathrm{Serv}_2, \mathrm{Time}_2) \land \neg \mathit{available\_to\_all}(\mathrm{Time}_2) \to$
$\neg \mathit{ordered}(\mathrm{Pati}_2, \mathrm{Serv}_2, \mathrm{Time}_2)$.

Then for scenario S_2, we consider

$$\Gamma_2 = \{(\mathrm{s4}), (\mathrm{s5}), (\mathrm{s6}), (\mathrm{r14}), (\mathrm{r17}), (\mathrm{r18})\}.$$

By using our tool *IncMeasurer*, we conclude that both Γ_1 and Γ_2 are *inconsistent*.

(C2) **Analyzing Inconsistency**. For the simplicity of discussion, we use $\perp_1$ and $\perp_2$ to denote the inconsistencies of Γ_1 and Γ_2, respectively. Then

- Doctor's Secretary (v_1) and Ward Manager (v_2) are involved in $\perp_1$;
- Doctor's Secretary (v_1), Medical Staff (v_3), and Lab Service Staff (v_4) are involved in $\perp_2$.

To illustrate how useful our approach is during the requirements stage, we provide and compare two cases of inconsistency analysis, i.e., inconsistency analysis without use of significance of inconsistency, and inconsistency analysis making use of significance of inconsistency.

- **Case 1: Inconsistency analysis without use of significance of inconsistency:** Each viewpoint wants to achieve his/her main goals in resolving *more important* inconsistencies by making concession in resolving *less important* inconsistencies. The developers of the EHR system want to differentiate these two inconsistencies from an integrated perspective, which takes each viewpoint's opinion into account. At first, they consider using *the degree of inconsistency* to distinguish $\perp_1$ from $\perp_2$. Let S_1 and S_2 be the scoring functions for Γ_1 and Γ_2, respectively. Then

$$S_1(\Gamma_1) = 1, \quad S_2(\Gamma_2) = 1; \quad S_1(\emptyset) = 0, \quad S_2(\emptyset) = 0;$$
$$\forall \Theta_1 \subset \Gamma_1 \text{ and } \Theta_1 \neq \emptyset, \quad S_1(\Theta_1) = 1;$$
$$\forall \Theta_2 \subset \Gamma_2, \text{ and } \Theta_2 \neq \emptyset, \quad S_2(\Theta_2) = 1;$$

  And they can get $S_1 \simeq S_2$. That is, the degree of inconsistency in Γ_1 equals to that in Γ_2. The developers cannot differentiate $\perp_1$ and $\perp_2$ by the degree of inconsistency. Then developers need to consider the relative importance of inconsistencies. However, these viewpoints do not give each inconsistency equal chance to be considered. In detail,

  (a) v_1 (Doctor's Secretary) is involved in $\perp_2$ as well as $\perp_1$. v_1 considers that $\perp_1$ is more important than $\perp_2$, since two essential requirements of v_1, i.e., (r6) and (r7), are involved in $\perp_1$, in contrast, only one essential requirement of v_1 is involved in $\perp_2$.

  (b) v_2 (Ward Manager) is indifferent to $\perp_2$ since v_2 is not involved in $\perp_2$. Actually, v_2 is only anxious to resolve $\perp_1$. Then v_2 tells the developers that $\perp_1$ is more significant than $\perp_2$.

  (c) v_3 (Medical Staff) and v_4 (Lab Service Staff) consider that $\perp_2$ is more important than $\perp_1$, since v_3 and v_4 are only involved in $\perp_2$.

  Obviously, v_1 and v_2 *disagree* with v_3 and v_4 in the problem of which inconsistency should be preferred. Particularly, v_1 and v_3 *disagree* with each other in the importance of $\perp_2$, which involves the shared requirement between v_1 and v_3. *This disagreement puts the developers into a dilemma.*

- **Case 2: Inconsistency analysis making use of significance of inconsistency:** If we consider the measurement of significance of inconsistency presented in this paper, we can help developers

  (2.1) differentiate the two inconsistencies in terms of significance of inconsistency from an integrated perspective.

  – To reach an agreement on the significance of the inconsistency, we need compare Γ_1 and Γ_2 from the integrated perspective. We input the instantiated facts and requirements in the tool *IncMeasurer*, as shown in Fig. 3, we can get the following result in 33 ms [5]:

  $\perp_1$ *is more significant than* $\perp_2$.

  This result signifies that the developers can consider that $\perp_1$ is more important than $\perp_2$ by aggregating four viewpoints' opinions. That is, a reasonable compromise among the four stakeholders is that $\perp_1$ should be preferred. The actions for resolving $\perp_1$ should take priority over the actions for resolving $\perp_2$.

  – To support and explain the result above, we can also compute the significance scoring matrix functions for Γ_1 and Γ_2, respectively. By using the *IncMeasurer*, we get each of the following corresponding significance scoring matrix functions $\mathbf{ScM}^1$ and $\mathbf{ScM}^2$ in 17 ms on the same Notebook PC.

$$\mathbf{ScM}^1(\Gamma_1) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 3 \end{bmatrix}; \qquad \mathbf{ScM}^2(\Gamma_2) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 2 \\ 0 & 0 & 2 \end{bmatrix};$$

$$\forall \Theta_1 \subset \Gamma_1, \Theta_1 \neq \emptyset, \quad \mathbf{ScM}^1(\Theta_1) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 3 \end{bmatrix}; \qquad \mathbf{ScM}^1(\emptyset) = \mathbf{0};$$

$$\forall \Theta_2 \subset \Gamma_2, \Theta_2 \neq \emptyset, \quad \mathbf{ScM}^2(\Theta_2) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 2 \\ 0 & 0 & 2 \end{bmatrix}; \qquad \mathbf{ScM}^2(\emptyset) = \mathbf{0}.$$

  The significance scoring matrix functions give the measurements of the significance of inconsistencies in Γ_1 and Γ_2, respectively. $(\mathbf{ScM}^1(\Gamma_1))_{33} = 3$ implies that each requirement involved in $\perp_1$ is considered as one of the most important requirements to its supporting viewpoint. Moreover, these supporting viewpoints are the most important to the project. In contrast, $(\mathbf{ScM}^2(\Gamma_2))_{33} = 2$ implies that (r14) is a shared *essential* requirement between v_1 and v_3, and $(\mathbf{ScM}^2(\Gamma_2))_{23} = 2$ coincides with that (r17) and (r18) are essential requirements of v_4, a *conditional* viewpoint.

[5] We performed it on a Notebook PC equipped with a Mobile DualCore Intel Core Duo T2400 1.83 Ghz CPU and 2 G memory.

(2.2) judge whether the intuitive opinion of each viewpoint about the relative importance of inconsistencies is reasonable.

Note that case (2.2) can be considered as a special case of case (2.1), in which there is an unique viewpoint and its requirements. For example, for viewpoint v_1, we can make use of our tool *IncMeasurer* to judge the opinion of v_1 mentioned in Case 1. We consider that (r8), (r17), and (r18) are also requirements of v_1, moreover, we would use $L = \{l_0^4, l_0^3, l_1^3, l_2^3\}$ and assign the priority level l_0^4 to requirements of (r8), (r17), and (r18) in the tool *IncMeasurer*.[6] Then we can get the following conclusion by comparing Γ_1 and Γ_2:

From the perspective of v_1, $\perp_1$ *is more significant than* $\perp_2$.

It signifies that the opinion of v_1 mentioned in **Case 1(a)** is reasonable from the perspective of v_1.

Part D. Further discussions

We learned the following three aspects from the case study:

- At first, having compared the second case to the first case, we can conclude that the measurement of the significance of inconsistency and the related tool *IncMeasurer* do play an important role in facilitating rational inconsistency analysis. In detail, by using the tool *IncMeasurer*, we have
  - checked inconsistency for Γ_1 and Γ_2;
  - compared inconsistencies in Γ_1 and Γ_2 from an integrated perspective;
  - provided an explanation for the compared results by computing the significance scoring matrix functions.
  - judged whether an individual viewpoint's intuitive opinion is rational from its own perspective.
- Secondly, we shall emphasize that this example demonstrates typically the application of our approach. It has been widely recognized that the relative priority of requirements can help developers to make some necessary trade-off decisions for resolving conflicts. However, for most distributed development such as viewpoints-based approaches, different stakeholders may assign different levels of priority to the same shared requirements statement from their own perspectives. The disagreement in the local levels of priority assigned to the same shared requirements statement often puts developers into a dilemma in many cases such as Case 1 mentioned above. It makes analyzing inconsistency from a integrated or global perspective more necessary. Actually, as illustrated by this case study, our approach is appropriate for such cases.
- Thirdly, as illustrated by this case, the scenario-based inconsistency checking and analysis involves only several relevant requirements rather than all the requirements. However, as argued in [5], consistency checking for the whole requirements specification often becomes impractical due to rather expensive computation cost. Although scenario-based inconsistency checking does not always guarantee consistency of the whole requirements specification, it is a practical way to improve the quality of requirements specification under computational complex constraints of inconsistency checking. This is also why we do not take an example of large scale to validate our approach.

6. Related works

Inconsistency is one of the pervasive issues in the software development. It has been increasingly recognized that flexible and effective approaches to managing inconsistency can facilitate the requirements process. In particular, measuring inconsistency has been considered a crucial part of the effective inconsistency management framework in requirements and software engineering [4]. In this paper, we presented an approach to measuring the significance of inconsistency arising from multiple viewpoints in requirements engineering. In the following, we compare our approach with some of closely related proposals for measuring inconsistency.

As mentioned in [12], most of the current proposals for measuring the degree of inconsistency may be classified into two categories, including formula-based category and variable-based (or interpretation-based) category. Roughly speaking, the variable-based proposal for measuring inconsistency of a set of formulas focuses on counting the (normalized) minimum number of variables assigned to inconsistent truth values in some paraconsistent models such as quasi-classical models [16] and LP_m models [13]. As argued earlier, lack of syntax sensitivity makes the variable-based proposal inappropriate for measuring the inconsistency arising from requirements. In contrast, formula-based approaches such as [14,13] are associated with the set of minimal inconsistent subsets of a set of formulas. The measures presented in [13] focused on identifying the degree of blame of each formula on the inconsistency arising from a set of formulas by distributing some measurements of inconsistency to all the formulas, whilst the scoring function proposed in [14] aimed to articulate the nature of inconsistency arising from a set of formulas by evaluating the contribution made by each subset to the inconsistency.

Obviously, most existing methods ignore an important issue concerning inconsistency, that is, the significance of inconsistency. Several research efforts have tried to look into this issue. An approach to evaluating the significance of inconsistency in the framework of QC logic was proposed in [21]. It is based on specifying the relative significance of

[6] Similar to example 4.1.

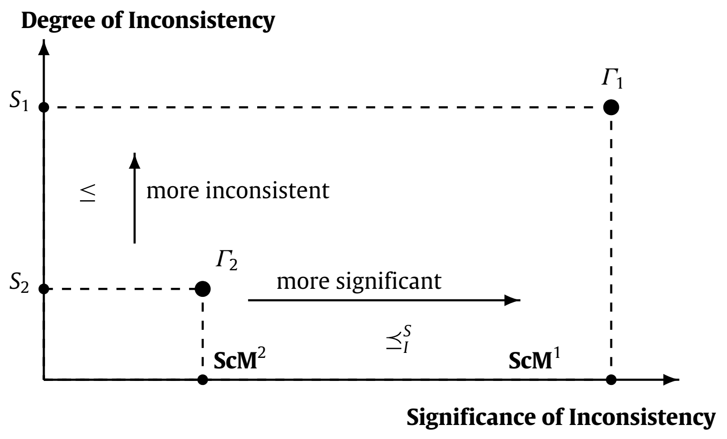


Fig. 4. Comparing two inconsistent sets of requirements statements.

incoherent QC models using some additional information, which is encoded as a mass assignment in Dempster-Shafer theory. It might not be appropriate for measuring inconsistency in requirements collections, since obtaining the relevant and important information is difficult in requirements engineering. Another particular research is the metric proposed to evaluate the level of significance of each detected inconsistency in multi-perspective requirements specifications [44,35]. However, the term of inconsistency in [44,35] is referred to as disagreement rather than classical logical contradictions. This approach and related works such as [45] focus on the stakeholder's belief in a given requirement (represented by a weighted truth value such as 1,0,1/2) and the belief merging rather than the relative importance of the requirement with regard to each stakeholder. Then the degree of disagreement among viewpoints rather than the relative importance of inconsistency with regard to the integrated perspective is described in these works.

In contrast, our approach to measuring the inconsistency presented in this paper focuses on the significance of the inconsistency arising from different viewpoints. We argue that the significance of the inconsistency arising from different viewpoints is linked to the relative importance of these different viewpoints as well as the relative importance of requirements with regard to these viewpoints. However, as pointed out in [28], there are few software organizations are willing to undertake rigorous numerical priority levels in practice. Therefore, in this paper, we adopt the qualitative priority levels for requirements within viewpoints as well as for viewpoints. The strength of our approach is to turn the qualitative priorities of viewpoints and the qualitative priorities of requirements with regard to the supporting viewpoints into a 0-1 vector. This vectorial transformation articulates the nature of qualitative priority levels. Moreover, it makes the computation of an integrated prioritization over the requirements specification from priorities in viewpoints feasible. There are few approaches to integrate the local priority levels into global priority levels such as [34,46,35]. However, the approaches to integrating the priorities levels from multiple perspectives presented in [34,35] focused on numerical or weighted priority levels. These approaches are inappropriate to handling the qualitative priority levels directly. In addition, combinatorial vote presented by Lang [46] also involved aggregating local preferences so as to obtain globally preferences in group decision. However, the computational complexity of votes [46] would block off its application in requirements engineering. Moreover, the relative importance of voters (corresponding to the viewpoints) is not considered in many voting rules. In contrast, our approach adopted the priority vector-based proposal presented in [11] to compute the integrated prioritization, which is easy to be implemented and more intuitive.

Based on the global prioritization over the requirements specification, the significance scoring matrix function is presented to measure the significance of the inconsistency in a combined requirements collection from the integrated perspective. We also present an algorithm for computing the significance scoring matrix functions. Moreover, we provide an intuitive ordering relationship between two inconsistent combined requirements collections that we can describe as *more significant than*. In this sense, our approach constructs a bridge from qualitative preference of requirements such as *essential* and *conditional* to numerical preference of inconsistency in requirements collections. Furthermore, as shown in Fig. 4, given two inconsistent requirements collections Γ_1 and Γ_2, we may compare Γ_1 and Γ_2 from two dimensions of the inconsistency, i.e., the degree of the inconsistency and the significance of the inconsistency using their scoring functions and significance matrix functions, respectively.

The inconsistency measure Inc_v for Type-II prioritized knowledge base presented in [47] is very close to the significance scoring matrix function presented in this paper. Both the two inconsistency measures for a set of formulas (resp. requirements) take into account the priority of each formula (resp. each requirement) in the set. However, the significance scoring matrix function aims to use all the contributions made by each subset to the significance of requirements involved in inconsistency together to capture the significance of inconsistency arising in a set of requirements. In contrast, the inconsistency measure Inc_v aims to integrate the degree of inconsistency and the significance of formulas involved in inconsistency by a single vector [47]. It may be considered as an integrated measure for inconsistency in some sense.

The approach to measuring the significance of inconsistencies arising from different viewpoints presented in this paper conforms with the scoring function presented by Hunter [14] with respect to the nature of inconsistency. As mentioned earlier, for any subset of a requirements collection Γ, denoted Θ, its significance scoring matrix function $\mathbf{ScM}(\Theta)$ may be viewed as a description of the reduction of the significance of those inconsistent information in Γ after Θ were removed from Γ. It is in essence the contribution made by Θ to the significance of the inconsistency in Γ. In this sense, it consists of a scoring function for articulating inconsistency in a set of formulas.

Our previous approach to measuring the significance of inconsistencies in a given viewpoints [23] is also a scoring function-based approach to measuring the inconsistency. It may be viewed as a simple case of the approach presented in this paper when there is only one viewpoint in the Viewpoints framework. However, the inconsistency arising from multiple viewpoints rather than the inconsistency in a given viewpoint is a more deserving and common issue in practical requirements engineering. Given an inconsistency involving a shared requirement among multiple viewpoints, the measurement of the inconsistency from an individual viewpoint is difficult to be accepted by other related viewpoints. The disagreement on the measurement of the inconsistency will result in confusion in making trade-off decision about inconsistency resolution. It makes the motivation of the approach presented here more necessary.

Note that the approach presented in this paper focuses on functional requirements only. It should be pointed out that conflicts between non-functional requirements (NFR for short) have also been the subject of substantial body of research literature (e.g. [32]). Although conflicts between non-functional requirements cannot be formulated as logical contradiction explicitly in many cases, there are several common characteristics among the conflicts between non-functional requirements and that between functional requirements:

- in many cases, some non-functional requirements are more preferred than others. This implies that some conflicts between non-functional requirements are more important than others. This makes measuring the significance of conflicts between non-functional requirements more necessary.
- any formulation of conflicts between non-functional or functional requirements should have two aspects: minimality and conflicts characterized in some special form. This seems to suggest that handling conflicts between NFR is similar to generating minimal inconsistent subsets that was used in functional requirements.

This makes our approach more promising to be adapted to the case of non-functional requirements.

To support the application of the approach to measuring inconsistency, we implemented a tool termed IncMeasurer, in which we formulated the problem of finding minimal inconsistent subsets as a search problem in a binomial tree. However, there are several other ways to finding minimal inconsistent subsets for a knowledge base in applications. For example, McAreavey et al. provided an algorithm for finding minimal inconsistent subsets by using existing algorithms for identifying Minimal unsatisfiable subformulas (MUSes) in a formula [48]. It may provide a potential way to promote our tool support in future.

7. Conclusions

In terms of the local priorities of requirements with respect to a given viewpoint and the relative importance of viewpoints, we have presented an approach to measuring the significance of inconsistency arising from different viewpoints.

This paper argues that the significance of the inconsistency arising from different viewpoints is associated with the relative importance of these different viewpoints as well as the relative importance of requirements with regard to these viewpoints.

We transformed the qualitative priority of an individual requirement in a given viewpoint to a 0-1 vector in which the location of unique 1 reflects the relative importance of the requirements with regard to the viewpoint. An integrated prioritization over the requirements specification was computed from the numerical representation of the priorities of viewpoints and the priorities of requirements with regard to their supporting viewpoints. Then we presented a scoring function-based approach to measuring the significance of inconsistency arising from different viewpoints. An ordering relationship between inconsistencies of two combined requirements collections, termed *more significant than*, has also been presented by comparing their significance scoring matrix functions.

In summary, this paper presented the following contributions to inconsistency measuring in requirements engineering:

(a) We argued that measurement of the significance of inconsistency in a requirements collection depends on the prioritization of requirements. Most researchers and practitioners have recognized that the prioritization of requirements plays an important role in requirements engineering [28,29]. Available prioritization of requirements makes measuring the significance of inconsistency feasible in software development.
(b) We defined the significance scoring matrix function to measure the significance of inconsistency in a combined requirements collection. It gives a concise articulation of the significance of inconsistency in requirements collections. Furthermore, we can compare two inconsistent combined requirements collections by means of their significance scoring matrix functions. This gives an intuitive ordering relationship between two combined requirements collections that we defined as *more significant than* with respect to inconsistency.
(c) We presented an algorithm for computing the significance scoring matrix function for a given combined requirements collection. To carry out the validation of algorithms presented in this paper and to support the application of the approach for measuring inconsistency, we implemented a tool termed IncMeasurer.

The approach presented in this paper also facilitates automated decision support for inconsistency handling. As mentioned above, the only input information that our approach needs is requirements priorities. Requirements priorities are often obtained during requirements elicitation activities either implicitly or explicitly through interaction between the developers and project sponsors and users. If our approach to measuring inconsistency is combined with techniques for automated inconsistency analysis and management, this integration may provide a fully automated approach for the identification of inconsistency, the measurement of inconsistency as well as the management of inconsistency in any given requirements collection. This will be the main direction for our future work. However, it should be also pointed out that this approach has applications in many other fields such as evaluating the progress in negotiations between a number of participants [14] and merging information from heterogeneous sources.

Note that inconsistency identifying in requirements in our approach is based on the choice of scenarios in some sense. Moreover, we assume that all the scenarios are developed correctly in the tool IncMeasurer in order to concentrate on requirements. However, some scenarios may also be problematic. How to choose and develop scenarios appropriately is still a challenge in scenario-based approaches for inconsistency handling [43]. In future work, we will also consider the case of prioritized scenarios and develop an approach to handling the inconsistency of a prioritized requirements specification with regard to a given prioritized scenario.

Acknowledgments

The authors are grateful to reviewers for their valuable comments. The authors would like to thank Dr. Guohui Xiao for his help on construction of the tool support. This work was partly supported by the National Natural Science Foundation of China under Grant Nos 61170300 and 60703061, the National Basic Research 973 program of China under Grant No. 2009CB320701, and the Key Project of National Natural Science Foundation of China under Grant No. 90818026. Didar Zowghi was supported by the Research Grant (CH070139) under the International Science Linkages Programme — Australia-China Special Fund for Science & Technology Cooperation Round 7.

References

[1] A. Finkelstein, J. Kramer, B. Nuseibeh, L. Finkelstein, M. Goedicke, Viewpoints: a framework for integrating multiple perspectives in system development, International Journal of Software Engineering and Knowledge Engineering 2 (1) (1992) 31–58.
[2] B. Nuseibeh, J. Kramer, A. Finkelstein, Viewpoints: meaningful relationships are difficult! in: Proceedings of International Conference on Software Engineering, IEEE Computer Society, Los Alamitos, CA, 2003, pp. 676–683.
[3] D. Zowghi, V. Gervasi, On the interplay between consistency, completeness, and correctness in requirements evolution, Information and Software Technology 45 (14) (2003) 993–1009.
[4] B. Nuseibeh, S. Easterbrook, A. Russo, Leveraging inconsistency in software development, IEEE Computer 33 (4) (2000) 24–29.
[5] B. Nuseibeh, S. Easterbrook, A. Russo, Making inconsistency respectable in software development, Journal of Systems and Software 58 (2) (2001) 171–180.
[6] V. Gervasi, D. Zowghi, Reasoning about inconsistencies in natural language requirements, ACM Transaction on Software Engineering and Methodologies 14 (3) (2005) 277–330.
[7] S. Castro, C. Roover, A. Kellens, A. Lozano, K. Mens, T. D'Hondt, Diagnosing and correcting design inconsistencies in source code with logical abduction, Science of Computer Programming 76 (12) (2011) 1113–1129.
[8] A. Lamsweerde, R. Darimont, E. Letier, Managing conflicts in goal-driven requirements engineering, IEEE Transactions on Software Engineering 24 (11) (1998) 908–926.
[9] A. Hunter, B. Nuseibeh, Managing inconsistent specification, ACM Transactions on Software Engineering and Methodology 7 (4) (1998) 335–367.
[10] K. Mu, Z. Jin, D. Zowghi, A priority-based negotiations approach for handling inconsistency in multi-perspective software requirements, Journal of Systems Science and Complexity 21 (4) (2008) 574–596.
[11] K. Mu, W. Liu, Z. Jin, A. Yue, R. Lu, D. Bell, Handling inconsistency in distributed software requirements specifications based on prioritized merging, Fundamenta Informaticae 91 (3–4) (2009) 631–670.
[12] A. Hunter, S. Konieczny, Shapley inconsistency values, in: Principles of knowledge representation and reasoning, Proceedings of the 10th international conference (KR06), AAAI Press, 2006, pp. 249–259.
[13] A. Hunter, S. Konieczny, Measuring inconsistency through minimal inconsistent sets, in: Principles of knowledge representation and reasoning, Proceedings of the eleventh international conference (KR08), 2008, pp. 358–366.
[14] A. Hunter, Logical comparison of inconsistent perspectives using scoring functions, Knowledge and Information Systems 6 (5) (2004) 528–543.
[15] A. Hunter, S. Konieczny, Approaches to measuring inconsistent information, in: Inconsistency Tolerance: Lecture Notes in Computer Science, vol. 3300, Springer-Verlag, New York, Secaucus, NJ, USA, 2004, pp. 191–236.
[16] A. Hunter, Measuring inconsistency in knowledge via quasi-classical models, in: Proc. The 18th National Conference on Artificial Intelligence, AAAI'2002, AAAI Press, Menlo Park, USA, 2002, pp. 68–73.
[17] S. Benferhat, D. Dubois, S. Kaci, H. Prade, Encoding information fusion in possibilistic logic: a general framework for rational syntactic merging, in: Proc. ECAI'2000, IOS Press, Amsterdam, 2000, pp. 3–7.
[18] S. Konieczny, J. Lang, P. Marquis, Quantifying information and contradiction in propositional logic through test actions, in: Proc. IJCAI2003, Morgan Kaufmann, San Fransisco, CA, USA, 2003, pp. 106–111.
[19] J. Grant, A. Hunter, Measuring inconsistency in knowledgebases, Journal of Intelligent Information Systems 27 (2006) 159–184.
[20] J. Grant, A. Hunter, Analysing inconsistent first-order knowledge bases, Artificial Intelligence 172 (2008) 1064–1093.
[21] A. Hunter, Evaluating the significance of inconsistency, in: Proc. IJCAI'03, Morgan Kaufmann, San Fransisco, CA, USA, 2003, pp. 468–473.
[22] K. Knight, Measuring inconsistency, Journal of Philosophical Logic 31 (1) (2002) 77–98.
[23] K. Mu, Z. Jin, R. Lu, W. Liu, Measuring inconsistency in requirements specifications, in: Proc.ECSQARU2005: LNCS vol. 3571, Springer-Verlag, New York, Secaucus, NJ, USA, 2005, pp. 440–451.
[24] G. Shafer, A Mathematical Theory of Evidence, Princeton University Press, Princeton, 1976.
[25] A. Russo, B. Nuseibeh, On the use of logical abduction in software engineering, in: Handbook of Software Engineering and Knowledge Engineering, World Scientific Publishing, 2001, pp. 889–914.
[26] D. Jackson, Automating first-order relational logic, ACM SIGSOFT Software Engineering Notes 25 (6) (2000) 130–139.
[27] R. Reiter, A theory of diagnosis from first priniciples, Artificial Intelligence 32 (1) (1987) 57–95.

[28] K.E. Wiegers, First things first: prioritizing requirements, Software Development 7 (9) (1999) 48–53.
[29] A. Davis, Just Enough Requirements Management: Where Software Development Meets Marking, Dorset House, New York, NY, USA, 2005.
[30] J. Karlsson, K. Ryan, A cost-value approach for prioritizing requirements, IEEE Software 14 (5) (1997) 67–74.
[31] J.W. Pardee, To Satisfy and Delight Your Customer: How to Manage for Customer Value, Dorset House Publishing, New York, 1996.
[32] K.E. Wiegers, Software Requirements, 2nd ed, Microsoft Press, 2003.
[33] IEEEStd830-1998, IEEE Recommended Practice for Software Requirements Specifications, IEEE Computer Society Press, Los Alamitos, CA, 1998.
[34] X. Liu, Y. Sun, C. Veera, Y. Kyoya, K. Noguchi, Priority assessment of software process requirements from multiple perspectives, Journal of Systems and Software 79 (11) (2006) 1649–1660.
[35] A.B.B. Martinez, J.J.P. Arias, A.F. Vilas, J.G. Duque, M.L. Norse, R.P.D. Redondo, Y.B. Fernandez, Composing requirements specifications from multiple prioritized sources, Requirements Engineering 13 (3) (2008) 187–206.
[36] K. Mu, W. Liu, Z. Jin, R. Lu, A. Yue, D.A. Bell, A merging-based approach to handling inconsistency in locally prioritized software requirements, in: Z. Zhang, J.H. Siekmann (Eds.), KSEM, in: Lecture Notes in Computer Science, vol. 4798, Springer, 2007, pp. 103–114.
[37] S.A. Cook, The complexity of theorem-proving procedures, in: Proc. of the Third Annual ACM Symposium on the Theory of Computing, ACM, New York, NY, USA, 1971, pp. 151–158.
[38] J. Marques-Silva, I. Lynce, S. Malik, Conflict-driven clause learning sat solvers, in: A. Biere, M. Heule, H. van Maaren, T. Walsh (Eds.), Handbook of Satisfiability, IOS Press, 2009, pp. 131–154.
[39] C. Grover, C. Brew, M. Moens, S. Manandhar, Priority union and generalization in dsicourse grammar, in: Proc. of the 32nd Annual Meeting of the Association for Computational Linguistics, Morgan Kaufmann, San Fransisco, CA, USA, 1994, pp. 17–24.
[40] R. Malouf, Maximal consistent subsets, Computational Linguistics 33 (2) (2007) 153–160.
[41] R. Bird, R. Hinze, Functional pearl:trouble shared is trouble halved, in: Proceedings of the 2003 ACM SIGPLAN workshop on Haskell, ACM, New York, NY, USA, 2003, pp. 1–6.
[42] L. Ryan, Efficient algorithms for clause learning sat solvers, Master's Thesis, Simon Fraser University, 2004.
[43] K. Mu, J. Hong, Z. Jin, W. Liu, From inconsistency handling to non-canonical requirements management: a logical perspective, International Journal of Approximate Reasoning 54 (1) (2013) 109–131.
[44] A.B.B. Martinez, J.J.P. Arias, A.F. Vilas, J.G. Duque, M.L. Norse, R.P.D. Redondo, Y.B. Fernandez, On the interplay between inconsistency and incompleteness in multi-perspective requirements specifications, Information and Software Technology 50 (4) (2008) 296–321.
[45] A. Martinez, J. Arias, A. Vilas, Merging requirements views with incompleteness and inconsistency, in: Proc. of Australian Software Engineering Conference 2005, IEEE Computer Society, Los Alamitos, CA, 2005, pp. 58–67.
[46] J. Lang, From logical preference representation to combinatorial vote, in: Proc. of KR2002, Morgan Kaufmann, San Fransisco, CA, USA, 2002, pp. 277–288.
[47] K. Mu, W. Liu, Z. Jin, Measuring the blame of each formula for inconsistent prioritized knowledge bases, Journal of Logic and Computation 22 (3) (2012) 481–516.
[48] K. McAreavey, W. Liu, P. Miller, C. Meenan, Tools for finding inconsistencies in real-world logic-based systems, in: Proceedings of the 6th European Starting AI Researcher Symposium, STAIRS'12, IOP Press, Montpellier, France, 2012, pp. 192–203.