Fundamenta Informaticae 91 (2009) 1–40
DOI 10.3233/FI-2009-0008
IOS Press

Handling Inconsistency In Distributed Software Requirements Specifications Based On Prioritized Merging

Kedian Mu[*]
School of Mathematical Sciences
Peking University
Beijing 100871, P.R.China
mukedian@math.pku.edu.cn

Weiru Liu
School of Electronics, Electrical Engineering and Computer Science
Queen's University Belfast, BT7 1NN, UK
w.liu@qub.ac.uk

Zhi Jin[†]
School of Electronics Engineering and Comp. Sci., Peking University
Key Laboratory of High Confidence Software Technologies, Ministry of Education
Beijing 100871, P.R.China
zhijin@sei.pku.edu.cn

Ruqian Lu[‡]
Academy of Mathematics and System Sciences
Chinese Academy of Sciences
Beijing 100080, P.R.China
rqlu@math.ac.cn

Anbu Yue
School of Electronics, Electrical Engineering and Computer Science
Queen's University Belfast, BT7 1NN, UK
a.yue@qub.ac.uk

David Bell
School of Electronics, Electrical Engineering and Computer Science
Queen's University Belfast, BT7 1NN, UK
da.bell@qub.ac.uk

Abstract. Developing a desirable framework for handling inconsistencies in software requirements specifications is a challenging problem. It has been widely recognized that the relative priority of requirements can help developers to make some necessary trade-off decisions for resolving conflicts.

Address for correspondence: Kedian Mu, School of Mathematical Sciences, Peking University, Beijing 100871, P.R. China
[*]Thanks to four anonymous referees for valuable comments and suggestions on improving the paper. Funding was provided in part by the National Natural Science Foundation of China under Grant No. 60703061, the National 863 High-tech Project of China under Grant No. 2006AA01Z155, the Key Project of National Natural Science Foundation of China under Grant No. 90818026, and the NSFC and the British Royal Society China-UK Joint Project.
[†]Funding was provided in part by the National Natural Science Fund for Distinguished Young Scholars of China under Grant No. 60625204, the National Key Research and Development Program of China under Grant No. 2002CB312004, and the Key Project of National Natural Science Foundation of China under Grant No. 90818026.
[‡]Funding was provided in part by the Key Project of National Natural Science Foundation of China under Grant No.60496324.

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 during the inconsistency handling process. The main contribution of this paper is to present a prioritized merging-based framework for handling inconsistency in distributed software requirements specifications. Given a set of distributed inconsistent requirements collections with the local prioritization, we first construct a requirements specification with a prioritization from an overall perspective. We provide two approaches to constructing a requirements specification with the global prioritization, including a merging-based construction and a priority vector-based construction. Following this, we derive proposals for handling inconsistencies from the globally prioritized requirements specification in terms of prioritized merging. Moreover, from the overall perspective, these proposals may be viewed as the most appropriate to modifying the given inconsistent requirements specification in the sense of the ordering relation over all the consistent subsets of the requirements specification. Finally, we consider applying negotiation-based techniques to viewpoints so as to identify an acceptable common proposal from these proposals.

Keywords: Inconsistency; Requirements Engineering; Prioritized Merging; Local Prioritization

1. Introduction

For any proposed software project, its software requirements specification [1] plays a prominent role in the development process. It provides a baseline for subsequent development stages including design, coding, testing and maintenance. Consequently, a software requirements specification of good quality is crucial for the project success.

However, poor requirements, incorrect specifications, and ineffective requirements management are still identified as major sources of problems in the development process [1, 2]. Errors being made during the requirements stage account for 40 to 60 percent of all the defeats found in a software project [3, 4]. To make matters worse, it is extremely expensive to correct these errors if they leaked into the subsequent phases in the software development life cycle [5]. Thus developing the software requirements specifications of good quality is still a very important but challenging issue.

Intuitively, to elicit authentic requirements for a system-to-be, it is advisable to advocate that each stakeholder expresses his demands only from his own perspective rather than from a global perspective. For any complex software system, the development of requirements typically involves many different stakeholders with different concerns. Then the software requirements specifications are increasingly developed in a distributed fashion.

Viewpoints-based approaches [6, 7, 8] may be considered as notable examples of distributed specifications development. We focus on the inconsistency handling in the Viewpoints framework in this paper. The Viewpoints framework has been developed to represent and analyze the different perspectives and their relationships during the requirements stage. A viewpoint is a description of system-to-be from the perspective of a particular stakeholder, or a group of stakeholders. It reflects the concerns of a particular stakeholder. The requirements specification of the system-to-be comprises a structured collection of loosely coupled, locally managed, distributable viewpoints, with explicit relationships between them to

[1]The term software requirements specification is referred to as requirements descriptions mostly in requirements community.

represent their overlaps [9]. These viewpoints may overlap, complement, or contradict with each other, then it makes inconsistency management more necessary during the requirements stage.

Generally speaking, inconsistency management may be divided into two parts, i.e. consistency checking and inconsistency handling. Consistency checking is a pervasive issue in requirements validation and verification. It focuses on techniques for detecting inconsistencies in a collection of requirements, including logic-based approaches [10, 11, 12] and consistency-rule-based approaches [13, 14]. For logic-based approaches, the term of *inconsistency* is defined as any situation in which some fact and its negation can be simultaneously derived from the same requirements collection. That is, inconsistency is referred to as the *logical contradiction*. Then the logic-based approaches have a theoretical basis. In contrast, the term of *inconsistency* in consistency-rule-based approaches is viewed as any situation in which two parts of a specification do not obey some relationship that should have been held between them. Some researchers argued that this definition of inconsistency is too general to be informative [12].

"*Inconsistency Implies Actions*" is recognized as a meta-rule for inconsistency handling [15, 11]. That is, when inconsistencies are detected, some actions should be performed to modify the inconsistent requirements. However, identifying appropriate actions is still a difficult, but important issue [11]. The choice of an inconsistency-handling action always depends on the nature and context of inconsistencies [15, 16]. But the context of inconsistency is rather complex. Many factors such as inappropriate description of requirements, misunderstanding between customers and requirements analysts, conflicting intentions of different stakeholders can all result in inconsistencies during the requirements stage.

Merging techniques have been considered in managing inconsistent viewpoints. For example, Easterbrook et al [34] presented a framework termed χbel for merging and reasoning about inconsistent state machine models using multi-valued logics. Their framework was intended to highlight the sources of inconsistency and to tolerate inconsistencies between viewpoints during model checking. It did not consider how to resolve these inconsistencies. Barragáns Martínez et al [35] defined a merging operator aiming to get a model which best reflects the combined knowledge of all the stakeholders (viewpoints) without first resolving inconsistencies and incompleteness. Although their methodology has envisioned two kinds of possible revision procedures to modify the original viewpoints, useful guidance on how to resolve these inconsistencies by using these revision procedures is not yet provided in [35]. These existing merging frameworks used in managing inconsistent viewpoints focused on tolerating inconsistency rather than resolving inconsistency in merged results.

It has been recognized that the relative priority of requirements can help requirements analysts resolve conflicts and make some necessary trade-off decisions during requirements elicitation and analysis stage [17, 18]. Given an inconsistent set of prioritized requirements, desirable actions should disengage most requirements with higher priority from inconsistency. Consequently, it may be desirable to manage inconsistent viewpoints to combine the relative priority of requirements and merging techniques.

However, prioritized merging [19] takes the relative priority of information into account when merging inconsistent information. Informally speaking, suppose that there are a stream or a sequence of possibly conflicting observations [2] with different reliability degrees about the same static world, prioritized merging aims to extract an *optimal* consistent view about the world by incorporating these observations based on a prioritized merging operator. Given an inconsistent requirements specification with prioritization, each requirements statement may be considered as an observation about the system-to-be.

[2] An observation is referred to as an item of information about the world.

Moreover, the priority level of an individual requirements statement may be considered as the relative importance of that requirements statement with regard to the system-to-be. Then the prioritized merging may be applied to identifying the consistent requirements appropriate to the system-to-be from the given requirements specification.

Note that we assume that the priority of each requirements statement implies its relative importance with regard to the whole system-to-be. That is, the requirements specification mentioned above should be prioritized from a global perspective. However, in the Viewpoints framework, the requirements always are only prioritized locally. For each viewpoint, the requirements of the viewpoint are prioritized from the perspective of the viewpoint rather than from the global perspective. Moreover, different viewpoints may adopt different scales of priority levels in local prioritization. Therefore, different viewpoints may assign different levels of priority to the same shared requirements statement. For a shared requirements statement, each priority given by a viewpoint is a measure of its relative importance with regard to this viewpoint. The disagreement in these local priorities assigned to the same shared requirements statement often puts developers into a dilemma.

To address these problems, we present a prioritized merging-based framework to handle inconsistency in the Viewpoints framework in this paper. First, we consider two methods for constructing a globally prioritized requirements specification from a set of requirements collections with the local prioritization, including the priority vector-based construction and the merging-based construction. Informally, the priority vector-based construction focuses on how to get the global priority of each requirements statement by integrating its existing local priorities. It is appropriate to the special cases that the viewpoints at the same level adopt the the same scale of local prioritization. In contrast, the merging-based construction considers each requirements collection with the local prioritization as a stratified knowledge base. The requirements specification with the global prioritization is constructed by merging these stratified knowledge bases based on the merging operator presented in [20]. It is appropriate for merging any viewpoints, especially for merging viewpoints with different scales of local prioritization. Following this, we map this requirements specification to a set of its consistent subsets by using a prioritized merging operator. The prioritized merging operator used in the mapping provides a relation over all the consistent subset of the requirements specification. Moreover, each consistent subset of the requirements specification in the prioritized merging result may be considered as optimal with regard to this relation. Then we derive some appropriate proposals for handling inconsistencies from the global perspective. Finally, we consider negotiation as a group making decision mechanism for identifying an acceptable common proposal from these proposals.

The rest of this paper is organized as follows. Section 2 gives an introduction to the Viewpoints framework, prioritized merging and the merging operator presented in [20], respectively. Section 3 provides a general framework for prioritized merging-based approach to handling inconsistency in the Viewpoints framework. Section 4 generalizes our previous work [32] on merging-based approaches to constructing a globally prioritized requirements collection. Section 5 presents a priority vector-based approach to constructing a globally prioritized requirements collection. The corresponding prioritized merging-based frameworks are also specified respectively in this two sections. Section 6 uses a case study to illustrate how to apply the prioritized merging-based approaches to handling inconsistency in requirements development. Section 7 discusses some issues such as comparison of the two approaches to constructing a globally prioritized requirements collection in the prioritized merging-based framework. Section 8 compares our work with related work. Finally, we conclude this paper in Section 9.

2. Preliminaries

2.1. Logical representation of Viewpoints

We consider the use of classical logic-based language in representation of viewpoints in this paper. Although heterogeneity of representation allows different viewpoints to use different notations and tools to represent their requirements during the requirements stage [9], first order logic is appealing for formal representation of requirements statements since most tools and notations for representing requirements could be translated into formulas of first order logic [11]. That is, first order logic may be considered as a promising tool to represent different viewpoints and their relationships uniformly. Moreover, in a logic based framework for representing requirements, reasoning about requirements is always based on some facts that describe a certain scenario [11]. It implies that checking the consistency of requirements collections only considers ground formulas[3] rather than unground formulas. Furthermore, if we restrict the first order language to propositional case, it may render consistency checking decidable. This gives some computational advantages. For these reasons, we assume a classical first order language without function symbols and existential quantifiers. This classical first order logic is the most convenient to illustrate our approach, as will be shown in the rest of the paper.

Let $\mathcal{L}_{\Phi_0}$ be the language composed from a set of classical atoms Φ_0 and logical connectives $\{\vee, \wedge, \neg, \rightarrow\}$ and let $\vdash$ be the classical consequence relation. Let $S_1, \cdots, S_n$ be a disjoint sequence of sets of formulas in $\mathcal{L}_{\Phi_0}$, we use $\langle S_1, \cdots, S_n\rangle$ to denote a stratified set of formulas, in which there is the following pre-order relation $\preceq$, over these formulas: $\forall \alpha \in S_i, \beta \in S_j$,

- $\alpha \preceq \beta$ [4] if and only if $i \leq j$;
- $\alpha \simeq \beta$ if and only if $\alpha \preceq \beta$ and $\beta \preceq \alpha$;
- $\alpha \prec \beta$ if and only if $\alpha \preceq \beta$ and $\beta \npreceq \alpha$.

If $S_i \neq \emptyset$ for each i, we also use $(S_1, \cdots, S_n)$ instead of $\langle S_1, \cdots, S_n\rangle$.

Let $\alpha \in \mathcal{L}_{\Phi_0}$ be a classical formula and $\Delta \subseteq \mathcal{L}_{\Phi_0}$ a finite set of formulas in $\mathcal{L}_{\Phi_0}$. In this paper, we call Δ *a set of requirements statements (or a requirements collection)* while each formula $\alpha \in \Delta$ represents a requirements statement. For example, given a requirement of " *if Alice requests to borrow the book of Algorithm and the book is available, then Alice can borrow the book*" in a certain scenario, we can represent the requirement by

$$require(Alice, Algorithm) \wedge available(Algorithm) \rightarrow borrow(Alice, Algorithm).$$

Generally, prioritization over a requirements collection Δ is just a strategy for differentiating requirements of Δ at a coarse granularity by its importance and urgency from some perspective. A common approach to prioritizing a requirements collection is to group requirements statements into several priority categories, such as the most frequent three-level scale of "*High*", "*Medium*", "*Low*" [21] and the five-level scale of priorities used in [18].

[3] 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.

[4] α is *more preferable to* β.

Another technique for prioritizing requirements specifications is based on numerical estimations of value, cost and risk of each requirements statement, such as cost-value approach [22] and Quality Function Deployment [23](QFD for short). However, K. Wiegers has pointed that few software organizations are willing to undertake the rigor of QFD in his experience [17].

In this paper, we adopt the common kind of prioritization to group requirements into several priority categories. Let m, a natural number, be the scale of the priority level and L^m be $\{l_1, \cdots, l_m\}$, a totally ordered finite set of m symbolic values of the priorities, i.e. $l_i < l_j$ iff $i < j$. Generally, $l_i < l_j$ means that *requirements with l_i are more preferable to requirements with l_j*. We also say that *requirements with l_i have a higher priority than that of requirements with l_j*. That is, a high value in L^m signifies a lower priority. Furthermore, each symbolic value in L^m could associate with a linguistic value. For example, for a three-level priority set, we have a totally ordered set L^3 as $L^3 = \{l_1, l_2, l_3\}$ where

$$l_1 : \textit{High},\ l_2 : \textit{Medium},\ l_3 : \textit{Low}$$

For example, if we assign l_1 to a requirements statement α, it means that α is one of the most important requirements statements. In the rest of paper, we adopt this three-level priority set in most examples, though it is not obligatory. From a given particular perspective, prioritization over Δ is in essence to establish a prioritization function $P : \Delta \longmapsto L^m$ by balancing the business value of requirements against its cost and risk. Actually, prioritizing a set of requirements statements Δ is to group Δ into m priority categories. That is, for every Δ, prioritization provides a partition of Δ, $\langle \Delta^1, \Delta^2, \cdots, \Delta^m \rangle$, where $\Delta^k = \{\alpha | \alpha \in \Delta, P(\alpha) = l_k\}$, for $k = 1, \cdots, m$. We then use $\langle \Delta^1, \Delta^2, \cdots, \Delta^m \rangle$ to denote a prioritized requirements collection in this paper. Just for convenience, we abbreviate $\langle \Delta^1, \Delta^2, \cdots, \Delta^m \rangle$ as $P \diamond \Delta$ in some discussion below.

In the Viewpoints framework, a viewpoint is a description of concerns of a particular group of stakeholders. Given a software project, let $V = \{v_1, \cdots, v_n\}$ $(n \geq 2)$ be the set of viewpoints. Suppose that L^{m_i} is the scale of priority levels adopted by v_i for all i. Let Δ_i be the set of requirements statements of viewpoint v_i and P_i the prioritization mapping from Δ_i to L^{m_i} for each $1 \leq i \leq n$. Then the requirements specification could be represented by a $n+1$ array $[P_1 \diamond \Delta_1, \cdots, P_n \diamond \Delta_n, R]$, where R is the set of relationships for consistency checking between these viewpoints, such as the relationships to represent their overlaps.

Because we use the classical logic as the uniform representation of viewpoints, an individual relation between v_i and v_j could also be explicitly represented by formulas involving some notations in $\Delta_i \cup \Delta_j$. For example, we may use $a \leftrightarrow b$ to denote that notation (or formula) a of v_i and b of v_j overlap totally [24]. Such notations should be added to the requirements set $\Delta_i \cup \Delta_j$ to check consistency of $\Delta_i \cup \Delta_j$ if necessary. Furthermore, for the sake of simplicity, we use $R(i_1, \cdots, i_k)$ to represent the relationship among viewpoints $v_{i_1}, \cdots, v_{i_k}$. We should check the consistency of $R(i_1, \cdots, i_k) \cup (\bigcup_{j=1}^{k} \Delta_{i_j})$ if $R(i_1, \cdots, i_k) \in R$.

We call v_i a *supporting viewpoint* of α if $\alpha \in \Delta_i$. Let $V(\alpha)$ denote a set of supporting viewpoint of α, then $V(\alpha) = \{v_i | \alpha \in \Delta_i, i \in [1, n]\}$. $|V(\alpha)| > 1$ means that α is a shared requirements statement of at least two viewpoints.

Different stakeholders play different roles during the software development. It is not surprising that some stakeholders are more important than others. Similar to prioritization of requirements, we prioritize viewpoints by group them into several priority categories. Let L_V^r be a r-level priority set used in prioritizing viewpoints. Then prioritizing viewpoints is to establish a prioritization mapping $P_V : V \longmapsto L_V^r$ in essence. Just for convenience, we also adopt the three-level priority set used in prioritizing

requirements to prioritize the viewpoints in the examples. Similar to requirements prioritization, P_V provides a partition of V, $\langle V^1, V^2, \cdots, V^r\rangle$, where $V^k = \{v | v \in V, P_V(v) = l_k\}$ for each $1 \leq k \leq r$. In the rest of this paper, we abbreviate the prioritized viewpoints $\langle V^1, V^2, \cdots, V^r\rangle$ as $P_V \diamond V$.

As mentioned earlier, the term of *inconsistency* has different definitions such as consistency rule-based definition and logical contradiction in requirements engineering [12]. Most logic-based works such as [11, 12, 10] concentrated on a particular kind of inconsistency, i.e. *the logical contradiction*: any situation in which some fact α and its negation $\neg\alpha$ can be simultaneously derived from the same requirements collection Δ. In this paper, we shall also be concerned with the logical contradiction. Let $\mathsf{Consequence}(\Delta) = \{\alpha | \Delta \vdash \alpha\}$. It is the set of all the consequences derived from Δ. If there is a formula α such that $\alpha \in \mathsf{Consequence}(\Delta)$ and $\neg\alpha \in \mathsf{Consequence}(\Delta)$, then we consider Δ to be *inconsistent* and abbreviate $\alpha \wedge \neg\alpha$ by $\bot$ (read inconsistency).

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

2.2. Knowledge bases merging

Merging is a common approach to fusing a set of heterogeneous information. Given a set of knowledge bases, the gist of knowledge base merging is to derive an overall knowledge base which best reflects the combined knowledge of all the original knowledge bases. A flat knowledge base K is a set of formulas in $\mathcal{L}_{\Phi_0}$. An interpretation is a total function from Φ_0 to $\{0, 1\}$, denoted by a bit vector whenever a strict total order on Φ_0 is specified. Ω is the set of all possible interpretations. An interpretation ω is a model of a formula φ, denoted $\omega \models \varphi$, iff $\omega(\varphi) = 1$. Then K is consistent iff there is at least a model of K.

A stratified knowledge base is a finite set K of formulas in $\mathcal{L}_{\Phi_0}$ with a total pre-order relation $\preceq$ on K. Intuitively, if $\varphi \preceq \psi$ then φ is regarded as more preferred or more important than ψ. From the pre-order relation $\preceq$ on K, K can be stratified as $K = (S_1, \cdots, S_n)$, where S_i contains all the minimal propositions of set $\bigcup_{j=i}^{n} S_j$ with regard to $\preceq$. Each S_i is called a stratum of K and is non-empty.We denote $\bigcup K = \bigcup_{j=i}^{n} S_j$. A knowledge profile E is a multiset [5] of knowledge bases, i.e. $E = \{K_1, \cdots, K_n\}$.

Many model-based as well as syntax-based merging operators have been presented to merge either flat or stratified knowledge bases. Informally, syntax-based operators aim to pick some formula in the union of the original bases. It may result in loss of some implicit beliefs during merging. In contrast, model-based merging operators aim to select some interpretations that are the closest to the original bases. They may also introduce external formulas. Most merging operators just generate a flat knowledge base as the result. At present, only the merging operators presented in [20] can be used to construct a stratified merged knowledge base. In this paper, we adopt the syntax-based operators presented in [20] to merge inconsistent requirements collections.

Given a stratified knowledge base $K = (S_1, \cdots, S_n)$, its models are defined as minimal interpretations with regard to a total pre-order relation $\preceq_X$ on interpretations that is induced from K by an ordering strategy X. The three widely used ordering strategies are defined as follows:

[5] A multiset is a set in which different occurrences of the same knowledge base are distinguished. For instance, $E = \{K_1, K_1, K_2\}$ comprises of three knowledge bases, K_1, K_1, and K_2. It allows us to represent a scenario that two different stakeholders (viewpoints) have the same set of requirements.

- *Best out ordering* [25]: let $r_{BO}(\omega) = \min\{i : \omega \not\models S_i\}$ for $\omega \in \Omega$, where $\min$ function returns the minimum value of i such that $\omega \not\models S_i$. By convention, $\min\{\emptyset\} = +\infty$. Then the best out ordering $\preceq_{bo}$ on Ω is defined as:

$$\omega \preceq_{bo} \omega' \text{ iff } r_{BO}(\omega) \geq r_{BO}(\omega').$$

  Note that for each interpretation ω, $r_{BO}(\omega)$ gives the highest priority of formulas that cannot be satisfied by ω. Then the best out ordering focuses on the most preferred stratum of K that cannot be satisfied by ω.

- *Maxsat ordering* [26]: let $r_{MO}(\omega) = \min\{i : \omega \models S_i\}$, for $\omega \in \Omega$. Then the maxsat ordering $\preceq_{mo}$ on Ω is defined as:

$$\omega \preceq_{mo} \omega' \text{ iff } r_{MO}(\omega) \leq r_{MO}(\omega').$$

  Essentially, for each ω, $r_{MO}(\omega)$ gives the highest priority that all the formulas with this priority are satisfied by ω. So, contrary to the best out ordering, the maxsat ordering concerns with the most preferred stratum of K satisfied by ω.

- *Leximin ordering* [25]: let $K^i(\omega) = \{\varphi \in S_i : \omega \models \varphi\}$, for $\omega \in \Omega$. Then the leximin ordering $\preceq_{lo}$ on Ω is defined as: $\omega \preceq_{lo} \omega'$ iff

  (1) $|K^i(\omega)| = |K^i(\omega')|$ for all i, or
  (2) there is an i such that $|K^i(\omega)| > |K^i(\omega')|$, and for all $j < i$, $|K^j(\omega)| = |K^j(\omega')|$, where $|K^i(\cdot)|$ denotes the cardinality of set $K^i(\cdot)$.

  Generally, we use $K(\omega)$ to denote $(K^1(\omega), \cdots, K^n(\omega))$. Actually, the leximin ordering over Ω is based on the lexicographical relation $\{K(\omega), \omega \in \Omega\}$. Compared to the two ordering strategies above, it considers the number of all formulas satisfied by a given interpretation ω as well as the priority of each formula satisfied by ω.

Given a stratified knowledge base K, from the pre-order relation $\preceq_X$ induced from K on Ω, the interpretations in Ω can also be stratified as $\Omega_{K,X} = (\Omega_1, \cdots, \Omega_m)$.

Yue et al [20] have presented an approach to deriving a stratified knowledge base K as a merged result of given knowledge profile $E = \{K_1, \cdots, K_n\}$, which is more appropriate to scenarios that there is no common stratification scale among different original knowledge bases. They argued that if the knowledge bases are designed independently, then only the relative preference between interpretations induced from a knowledge base by some ordering strategy is meaningful in a merging process.

Definition 2.1. (Relative Preference Relation [20])
Given a knowledge Profile $E = \{K_1, \cdots, K_n\}$, let $\{\Omega_{K_1,X_1}, \cdots, \Omega_{K_n,X_n}\}$ be a multi-set of stratifications of Ω. A binary relative preference relation $R \subseteq \Omega \times \Omega$ is defined as: $R(\omega, \omega')$ iff $|\{\Omega_{K_i,X_i}\ s.t.\ \omega \prec_i \omega'\}| > |\{\Omega_{K_i,X_i}\ s.t.\ \omega' \prec_i \omega\}|$, where $\prec_i$ is the strict partial order relation induced from Ω_{K_i,X_i}.

Note that $R(\omega, \omega')$ means that more knowledge bases prefer ω than ω'.

Definition 2.2. (Undominated Set [20])
Let R be a relative preference relation over Ω and let Q be a subset of Ω. Q is called an undominated set of Ω, if $\forall\omega \in Q,\ \forall\omega' \in \Omega - Q$, $R(\omega', \omega)$ does not hold. Undominated set Q is called a minimal undominated set if none of its proper subsets is an undominated set.

In other words, Q is an undominated one if no interpretation outside Q dominates some interpretation in Q via relative preference relation R. We denote the set of minimal undominated sets of Ω w.r.t R as U^R_Ω. Then we can stratify the interpretations Ω as $\Omega = (\Omega_1, \cdots, \Omega_n)$, where $\Omega_i = \cup Q$, and $Q \in U^R_{\Omega - \cup_{j=1}^{i-1}\Omega_j}$. Following this, we may define X dominated construction as the stratified merged result of given knowledge profile E.

Definition 2.3. (X dominated construction [20])
Let $\Omega = (\Omega_1, \cdots, \Omega_n)$ be a stratification of interpretation and S be a set of formulas. Let X be an ordering strategy. A stratified knowledge base $K_S^{X,\Omega} = (S_1, \cdots, S_m)$ is an X dominated construction from S w.r.t Ω if $\bigcup_{i=1}^{m} S_i \subseteq S$ and $\Omega_{K_S^{X,\Omega},X} = \Omega$.

Essentially, an X dominated construction from S w.r.t Ω is a subset of S that stratifies Ω as $(\Omega_1, \cdots, \Omega_n)$. In this sense, it can be considered as a stratified knowledge base which reflects the combined preference as well as knowledge of original knowledge bases in E.

The following proposition [20] shows how to construct an X dominated construction as a stratified merged result from the original bases based on the stratification of Ω obtained from R.

Proposition 2.1. Let $\Omega = (\Omega_1, \cdots, \Omega_n)$ be a stratification of interpretation and S be a set of propositions.

- If there exists a stratified knowledge base K s.t. $\Omega_{K,bo} = \Omega$ and $\bigcup K \subseteq S$, then $K_S^{bo,\Omega} = (S_1, \cdots, S_{n-1})$ is a best out dominated construction from S w.r.t Ω, where

$$S_i = \{\varphi \in S | \forall \omega \in \Omega_j, \omega \models \varphi, \forall j \in [1, n-i]\} - \bigcup_{j=1}^{i-1} S_j \text{ and } S_i \neq \emptyset.$$

- If there exists a stratified knowledge base K s.t. $\Omega_{K,mo} = \Omega$ and $\bigcup K \subseteq S$, then $K_S^{mo,\Omega} = (S_1, \cdots, S_n)$ is a maxsat-dominated construction from S w.r.t Ω, where

$$S_i = \{\varphi \in S | \forall \omega \in \Omega_i, \omega \models \varphi\} - \bigcup_{j=1}^{i-1} S_j \text{ and } S_i \neq \emptyset.$$

- If there exists a stratified knowledge base $K = (S_1, \cdots, S_n)$ s.t. each S_i is a singleton set, $\Omega_{K,lo} = \Omega$ and $\bigcup K \subseteq S$, then $K_S^{lo,\Omega} = (S_1, \cdots, S_n)$ is a leximin dominated construction from S w.r.t Ω, where

$$S_i = \{\varphi \in S | \forall \omega \in \Omega_i, \omega \models \varphi \text{ where } \forall j > i, \forall \omega \in \Omega_j, \omega \not\models \varphi\} \text{ and } S_i \neq \emptyset.$$

Note that Proposition 2.1 only addresses the leximin dominated construction in which each stratum is a singleton set. In general, we may also construct a stratified merged result K according to the approach presented in Proposition 2.1 from $(\Omega_1, \cdots, \Omega_n)$, i.e., $K = (S_{n_1}, \cdots, S_{n_m})$, where $S_{n_1}, \cdots, S_{n_m}$ are given by deleting all $\emptyset$ from sequence $S_1, \cdots, S_n$. But we cannot guarantee that K satisfies $\Omega_{K,X} = (\Omega_1, \cdots, \Omega_n)$ if some stratum of K is not a singleton set. It needs further verification.

Now we give an example to illustrate the merging process.

Example 2.1. Consider $E = \{K_1, K_2, K_3\}$, where $K_1 = (\{p \wedge q\}, \{p\})$, $K_2 = (\{p \wedge q\}, \{p\}, \{\neg q\})$, $K_3 = (\{p\}, \{q\}, \{\ \neg p\})$. If we adopt the leximin ordering strategy, we can stratify the interpretations $\Omega = \{\omega_1 = 11, \omega_2 = 10, \omega_3 = 01, \omega_4 = 00\}$ as follows:

$$\Omega_{K_1,\, lo} = (\{\omega_1\}, \{\omega_2\}, \{\omega_3, \omega_4\});$$
$$\Omega_{K_2,\, lo} = (\{\omega_1\}, \{\omega_2\}, \{\omega_4\}, \{\omega_3\});$$
$$\Omega_{K_3,\, lo} = (\{\omega_1\}, \{\omega_2\}, \{\omega_3\}, \{\omega_4\}).$$

Evidently, all the three stratifications prefer ω_1 than ω_2, then $R(\omega_1, \omega_2)$ holds. We may get other similar relations such as $R(\omega_2, \omega_3)$ and $R(\omega_2, \omega_4)$.

Further, we have $\{\omega_1\}$ as a minimal undominated set of Ω. $\{\omega_2\}$ is a minimal undominated set of $\Omega \backslash \{\omega_1\}$. Then we stratify Ω based on relative preference relation R as follows:

$$\Omega = (\{\omega_1\}, \{\omega_2\}, \{\omega_3, \omega_4\}).$$

Let $S = \{p \wedge q, p, q, \neg p, \neg q\}$, then we can construct the following different X dominated constructions:

$$K_S^{bo,\Omega} = (\{p\}, \{p \wedge q, q\}),$$
$$K_S^{mo,\Omega} = (\{p \wedge q, p, q\}, \{\neg q\}, \{\neg p\}),$$
$$K_S^{lo,\Omega} = (\{p \wedge q\}, \{p\}, \{\neg p\}).$$

Each can be viewed as a stratified merged result.

Actually, if there is an integrity constraint μ during the merging process, all the models not satisfying μ are meaningless to the merged result. We need to use Ω^μ instead of Ω in the definitions above, where Ω^μ is the set of all the models of μ.

2.3. Prioritized merging

Given a knowledge profile $E = \{K_1, \cdots, K_n\}$, we can get a stratified knowledge base $K = (S_1, \cdots, S_m)$ as the merged result. However, the merged operator presented in [20] cannot guarantee that K is a consistent knowledge base. We need another merging operator such as prioritized merging to get a consistent knowledge base which best reflects the knowledge in K.

Prioritized merging is considered as an appropriate merging operator for incorporating a stream or a sequence of (possibly conflicting) observations with an attached reliability degree about the same static world [19]. An observation is referred to as an item of information about the world. For example, the requirements collection with a global prioritization could be considered as a stream of observations about the same system-to-be.

Some concrete prioritized merging operators have been proposed in the literature. Delgrande et al surveyed existing prioritized merging operators and provided postulates for prioritized merging in [19].

We start with a brief introduction to the prioritized merging with a definition of a prioritized observation base.

Definition 2.4. (A prioritized observation base (POB) [19])
An observation α is defined as a consistent formula of $\mathcal{L}_{\Phi_0}$. A prioritized observation base (POB) is defined as a set of observations with an attached reliability degree:

$$\sigma = \langle \sigma(1), \cdots, \sigma(n) \rangle, \text{ for some } n \geq 1,$$

where each $\sigma(i)$ is a (possibly empty) set of propositional formulas, namely, the observations of reliability level i.

Evidentally, every stratified knowledge base is also a prioritized observation base. Note that we use a total pre-order relation $\prec_\sigma$ on σ in this paper. That is, for any two observations $\beta \in \sigma(i)$ and $\beta' \in \sigma(j)$, $\beta \prec_\sigma \beta'$ if and only if $i < j$. It means that β is viewed as *more reliable* than β'. Actually, for each viewpoint v_i, $P_i \diamond \Delta_i = \langle \Delta_i^1, \cdots, \Delta_i^m \rangle$ may be viewed as a prioritized observation base, moreover, each of requirements statements of Δ_i^k can be viewed as an observation of priority level l_k about the system-to-be from the perspective of v_i for each k.

We also adopt the following notations used in [19]. If $S \subseteq \mathcal{L}_{\Phi_0}$ then $\bigwedge(S)$ is the conjunction of all formulas in S, with the usual convention $\bigwedge(\emptyset) = \top$. In particular, $\bigwedge \sigma(i)$ is the conjunction of all formulas in $\sigma(i)$ and $\bigwedge(\sigma) = \bigwedge_{i=1}^{n} \bigwedge \sigma(i)$. We abbreviate $\langle \sigma(i), \cdots, \sigma(j) \rangle$ as $\sigma_{i \to j}$, for $1 \leq i \leq j \leq n$. $\widehat{\sigma}$ is the multiset $\bigcup_{i=1}^{n} \sigma(i)$.

Definition 2.5. (Concatenation Operator [19])
If $\sigma = \langle \sigma(1), \cdots, \sigma(n) \rangle$ and $\sigma' = \langle \sigma'(1), \cdots, \sigma'(p) \rangle$ then $\sigma \circ \sigma'$ is the *concatenation* of σ and σ', defined by $\sigma \circ \sigma' = \langle \sigma(1), \cdots, \sigma(n), \sigma'(1), \cdots, \sigma'(p) \rangle$.

If $\sigma = \langle \sigma(1), \cdots, \sigma(n) \rangle$ and $\sigma' = \langle \sigma'(1), \cdots, \sigma'(n) \rangle$ are two prioritized observation bases, we write $\sigma' \subseteq \sigma$ iff for every i, $\sigma'(i) \subseteq \sigma(i)$, where $\subseteq$ is here multiset inclusion, and we simply say that σ' is a subset of σ.

$Cons(\sigma)$ is the set of consistent subsets of σ, that is, the set of all POBs $S = \langle S_1, \cdots, S_n \rangle$ such that $S \subseteq \sigma$ and $\bigwedge(S)$ is consistent.

Let $\prec$ be a strict pre-order on a set X. For any $Y \subseteq X$, we use $Min(\prec, Y)$ to denote the undominated subset of Y with respect to $\prec$, i.e.,

$$Min(\prec, Y) = \{y \in Y | \text{there is no } z \in Y \text{ such that } z \prec y\}.$$

A prioritized merging operator $\star$ maps any POB σ to a propositional formula $\star(\sigma)$. The three representative prioritized merging operators are *Best-out* ($\star_b$), *Discrimin* ($\star_d$), and *Leximin* ($\star_l$). They are given as follows:

- *Best-out* [25]. Let $\kappa(\sigma) = \max\{i, \bigwedge \sigma_{1 \to i} \text{ consistent}\}$. Then $\star_b(\sigma) = \bigwedge \sigma_{1 \to \kappa(\sigma)}$.

  Essentially, the Best-out operator does not consider any observations less preferred than $\sigma_{\kappa(\sigma)}$. In this sense, it conforms with the best out ordering strategy mentioned above.

- *Discrimin* [27, 28, 25]. For $S, S' \in Cons(\sigma)$, define $S' \prec_d S$ iff $\exists k$ such that

  (a) $\sigma_{1 \to k} \cap S' \supset \sigma_{1 \to k} \cap S$, and
  (b) for all $i < k$, $\sigma_{1 \to i} \cap S' = \sigma_{1 \to i} \cap S$.

  Then $\star_d(\sigma) = \bigvee \{\bigwedge S, S \in Min(\prec_d, Cons(\sigma))\}$.

- *Leximin* [25, 29]. For $S, S' \in Cons(\sigma)$, define $S' \prec_l S$ iff $\exists k$ such that

(a) $|\sigma_{1\to k} \cap S'| > |\sigma_{1\to k} \cap S|$, and

(b) for all $i < k$, $|\sigma_{1\to i} \cap S'| = |\sigma_{1\to i} \cap S|$.

Then $\star_l(\sigma) = \bigvee\{\bigwedge S, S \in Min(\prec_l, Cons(\sigma))\}$.

Note that both the Discrimin operator and the Leximin operator depend on lexicographical ordering relations over $Cons(\sigma)$. Informally, the lexicographical ordering relation used in the Discrimin operator is based on set inclusion. In contrast, the lexicographical ordering relation used in the Leximin operator conforms with the leximin ordering strategy mentioned above.

Now we give an example to illustrate these prioritized merging operators.

Example 2.2. Consider $\sigma = \langle\sigma(1), \sigma(2)\rangle$, where $\sigma(1) = \{\alpha, \beta\}$ and $\sigma(2) = \{\gamma, \gamma \to \neg\psi, \psi\}$. Then

(a) $\kappa(\sigma) = 1$, and $\star_b(\sigma) = \bigwedge \sigma(1) = \alpha \wedge \beta$.

(b) $Min(\prec_d, Cons(\sigma)) = \{S, S', S''\}$, where

$$\begin{aligned} S &= \langle\{\alpha, \beta\}, \{\gamma, \gamma \to \neg\psi\}\rangle, \\ S' &= \langle\{\alpha, \beta\}, \{\gamma, \psi\}\rangle, \\ S'' &= \langle\{\alpha, \beta\}, \{\gamma \to \neg\psi, \psi\}\rangle. \end{aligned}$$

Then

$$\begin{aligned} \star_d(\sigma) = \quad & (\alpha \wedge \beta \wedge \gamma \wedge (\gamma \to \neg\psi)) \\ \vee \quad & (\alpha \wedge \beta \wedge \gamma \wedge \psi) \\ \vee \quad & (\alpha \wedge \beta \wedge (\gamma \to \neg\psi) \wedge \psi). \end{aligned}$$

(c) $Min(\prec_l, Cons(\sigma)) = Min(\prec_d, Cons(\sigma))$, then

$$\begin{aligned} \star_l(\sigma) = \quad & (\alpha \wedge \beta \wedge \gamma \wedge (\gamma \to \neg\psi)) \\ \vee \quad & (\alpha \wedge \beta \wedge \gamma \wedge \psi) \\ \vee \quad & (\alpha \wedge \beta \wedge (\gamma \to \neg\psi \wedge \psi)). \end{aligned}$$

Note that the Discrimin operator and the Leximin operator may give different results. For example, Consider $\sigma = \langle\{p\}, \{\neg p \vee q, \neg q\}, \{q\}\rangle$, then $\langle\{p\}, \{\neg q\}, \emptyset\rangle$ is an undominated element of $Cons(\sigma)$ with regard to the discrimin ordering $\prec_d$. But it is not an undominated element with regard to $\prec_l$, since $\langle\{p\}, \{\neg p \vee q\}, \{q\}\rangle \prec_l \langle\{p\}, \{\neg q\}, \emptyset\rangle$.

Just as mentioned in [19], the outcome of a merging operator can be defined as a closed theory equivalently. However, we can also define the equivalent outcome of these merging operators as a set of consistent subsets of σ:

- *Best-out*: $\star_b(\sigma) = \{S = \langle\sigma(1), \cdots, \sigma(\kappa(\sigma)), \emptyset, \cdots, \emptyset\rangle\}$, where $\emptyset$ means that there are no formulas at corresponding levels.
- *Discrimin*: $\star_d(\sigma) = Min(\prec_d, Cons(\sigma))$.

- *leximin*: $\star_l(\sigma) = Min(\prec_l, Cons(\sigma))$.

In the rest of this paper, we use this equivalent definitions of prioritized merging operators. Evidently, given a prioritized observation base σ, $\prec_d$ (resp. $\prec_l$) provides an intuitive ordering relation on a set of all the consistent subsets of σ. In particular, each $S \in \star_d(\sigma)$ (resp. $\star_l(\sigma)$) can be viewed as an *optimal* consistent subset in the sense of $\prec_d$ (resp. $\prec_l$).

However, there are some constraints between these observations about the same world in many cases. For example, the relationship between viewpoints can be viewed as constraints between requirements about the same system-to-be. In such case, we need to integrate the constraints into prioritized merging.

Let $\sigma = \langle \sigma(1), \cdots, \sigma(n) \rangle$ be a prioritized observation base and μ be a consistent set of constraints between these observations. A constraint is referred to as a consistent formula of $\mathcal{L}_{\Phi_0}$. Then we define

$$Cons^{\mu}(\sigma) = \{S | S \subseteq \sigma, \bigwedge S \wedge \bigwedge \mu \text{ is consistent.}\}$$

It is the set of subsets of σ that are consistent with μ. Furthermore, we define the prioritized merging under constraints as follows:

Definition 2.6. (prioritized merging under constraints)
Let σ be a prioritized observation base and μ be a consistent set of constraints on σ, then

- *Best-out.* $\star_b^{\mu}(\sigma) = \{S = \langle \sigma(1), \cdots, \sigma(\kappa^{\mu}(\sigma)), \emptyset, \cdots, \emptyset \rangle\}$, where $\kappa^{\mu}(\sigma) = \max\{i, \bigwedge \sigma_{1 \to i} \wedge \bigwedge \mu \text{ consistent}\}$.
- *Discrimin.* $\star_d^{\mu}(\sigma) = Min(\prec_d, Cons^{\mu}(\sigma))$.
- *leximin.* $\star_l^{\mu}(\sigma) = Min(\prec_l, Cons^{\mu}(\sigma))$.

3. A General Prioritized Merging-based Framework for Handling Inconsistent Requirements

The gist of this paper is to provide prioritized merging-based approaches to handling inconsistency in the Viewpoints framework. We assume that there is no shortcoming during the requirements elicitation process. That is, the requirements are correctly elicited, stated, and represented from the perspective of corresponding stakeholders. All the stakeholders affirm to their demands. We start these approaches with a general framework for handling inconsistency based on prioritized merging, as shown in Figure 1.

Given an inconsistent requirements specification $[P_1 \diamond \Delta_1, \cdots, P_n \diamond \Delta_n, R]$, let Δ_G be a multiset of all the requirements statements. We use $P_G \diamond \Delta_G$ to denote the globally prioritized requirements collection constructed from $[P_1 \diamond \Delta_1, \cdots, P_n \diamond \Delta_n, R]$. Let $\star^R(P_G \diamond \Delta_G)$ be the prioritized merging result of $P_G \diamond \Delta_G$ using the operator $\star$ under constraints R. Let A be a set of formulas. We use $A \backslash B$ to denote the set of formulas outside B if $B \subseteq A$. In contrast, we use $A - B$ to denote a proposal to abandon B in A. Then according to this framework, developers may adopt the following steps to handle the inconsistency:

(1) **Globally Prioritizing**:

- prioritize the whole requirements collection from the global perspective, i.e., provide a priority-based partition of the requirements collection Δ_G, $P_G \diamond \Delta_G$;

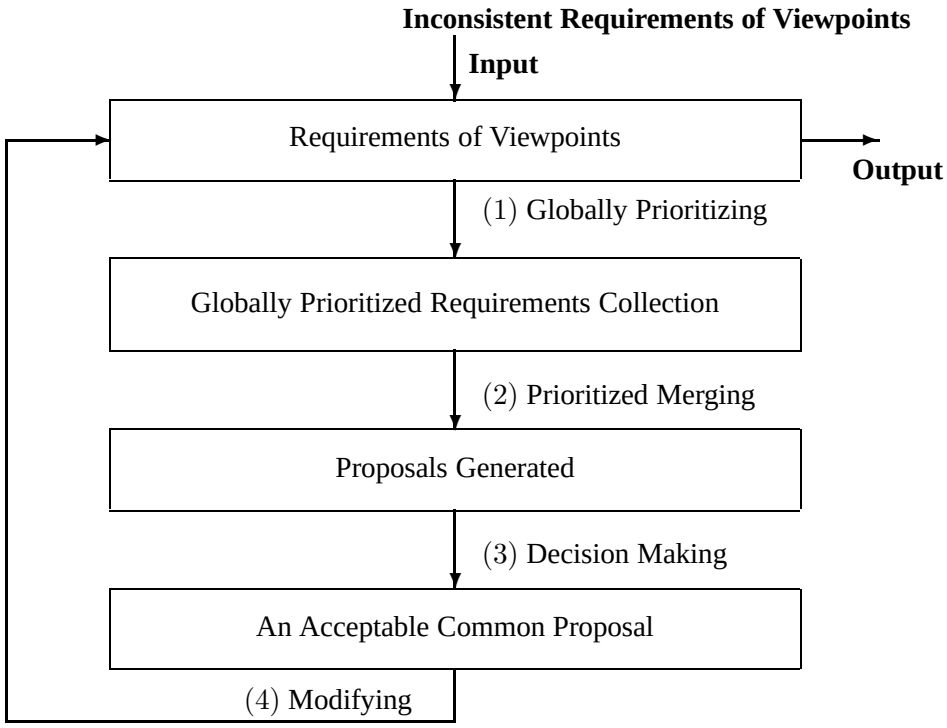

Figure 1. A General Framework of Prioritized Merging-based Approaches

(2) **Prioritized Merging**:

- map the requirements collection with the global prioritization $P_G \diamond \Delta_G$ into a set of its consistent subsets by a prioritized merging operator $\star$ under constraints R;
- derive some proposals for handling inconsistency in the requirements collection from the result of prioritized merging. Generally, if $S \in \star^R(P_G \diamond \Delta_G)$, then we may derive a proposal $\pi(S)$ to resolve the inconsistency as follows:

$$\pi(S)\text{: } \textit{abandon the requirements outside } S.$$

(3) **Decision Making**:

- identify a common proposal acceptable to all the viewpoints involved in the inconsistency;

(4) **Modifying**:

- modify $[P_1 \diamond \Delta_1, \cdots, P_n \diamond \Delta_n, R]$ according to the acceptable common proposal;

The union of modified requirements of each viewpoint involved in inconsistency becomes consistent. Then we output the modified requirements of viewpoints finally.

Two issues are important to this general framework, i.e. globally prioritizing Δ_G and prioritized merging $P_G \diamond \Delta_G$. Global Prioritization will be discussed in subsequent sections. Now we discuss the prioritized merging operators appropriate to this framework.

Generating proposals for resolving inconsistency based on prioritized merging. As mentioned earlier, there are three representative prioritized merging operators, including $\star_b$, $\star_d$, and $\star_l$. Suppose that we adopt the *best-out* operator $\star_b$ and $S \in \star_b^R(P_G \diamond \Delta_G)$, then according to the proposal $\pi(S)$, the requirements with the global priority level greater than $\kappa^R(P_G \diamond \Delta_G)$ should be abandoned, no matter whether these requirements are involved in inconsistencies or not. It may be considered as a disadvantage to using the *best-out* operator. In contrast, if we use $\star_d$ and $\star_l$, the proposals may conform to the intention of disengaging major requirements with higher priorities from the inconsistencies. Consequently, we adopt $\star_d$ and $\star_l$ in this paper.

Now we discuss how to compute $\star_d(\sigma)$ and $\star_l(\sigma)$ given σ. Generally, to compute $\star_d(\sigma)$ (resp. $\star_l(\sigma)$), each subset of σ, denoted σ', should be checked whether it is consistent and whether it is an undominated element of $Cons(\sigma)$ with regard to $\prec_d$ (resp. $\prec_l$). Since we restrict the first order logic representation of requirements to the propositional case, then consistency checking of σ' is a SAT problem. Many SAT solvers have been developed to solve the SAT problem efficiently for practical problem instances. Especially, the CDCL (Conflict-Driven Clause Learning) SAT algorithms can solve instances with *hundreds of thousand* (propositional) variables and *tens of millions* of clauses. For example, Siege [36] can solve a problem with 0.25 million (propositional) variables in less than 30 seconds. Thus we can adopt an available SAT solver to check consistency of a large-scale requirements collection σ'.

If σ' is consistent, we need also to check whether it is an undominated element of $Cons(\sigma)$. Some algorithms with optimization about this kind of problem have been developed. Informally, these algorithms first constructed a binomial tree of the boolean lattice of subsets of σ [37]. For example, if $\sigma = \langle\{\alpha, \beta, \neg\alpha\}\rangle$, then the boolean lattice and binomial tree are shown in Fig. 2.

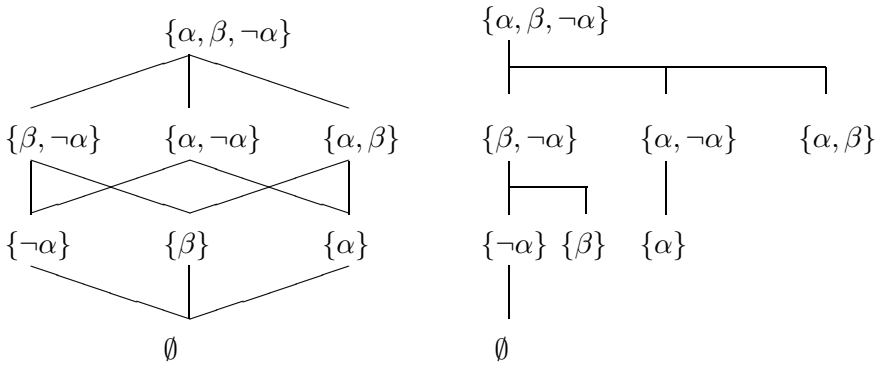

Figure 2. Boolean lattice and its binomial tree

Then the algorithms performed a breadth-first search of the subset lattice of σ since a breadth-first search of the binomial tree will consider all larger sets before considering any smaller ones. The algorithms with an important optimization (root pruning) presented by Grover et al. [38] proved that the branches rooted by a consistent subset can be pruned from the search space, because no descendants of a consistent set can be undominated. For example, $\{\beta, \neg\alpha\}$ is consistent, then its subtree is pruned. We get the locally undominated elements as $\{\beta, \neg\alpha\}$, $\{\alpha, \beta\}$, and $\{\alpha\}$. Furthermore, a final post-check for set inclusion (resp. lexicographical relation for $\prec_l$) can remove pseudo-undominated results like $\{\alpha\}$ from the set of locally undominated consistent subsets in their branch of the binomial tree. However,

Malouf [39] argued that the organization of the search space into a binomial tree allows another valuable optimization, i.e, leaf pruning. Roughly speaking, if the foot of a subtree is inconsistent then no node in the tree can be consistent, and the entire tree can be skipped. Then the only subtrees which contain a undominated consistent subset are those whose roots are inconsistent but whose deepest leafs are consistent. Malouf also pointed out that keeping track of leftmost children allows us to avoid a substantial number of redundant consistency checks [39]. Moreover, as $|\sigma|$ increases, leaf pruning can offer substantial improvements. These techniques for optimizations makes identification of undominated subsets efficient [39]. This means we can adopt the techniques described by Malouf [39] to find all the undominated elements of $Cons(\sigma)$.

On the other hand, requirements free from any inconsistency are always included in all S in $\star_d^R(P_G \diamond \Delta_G)$ (resp. $\star_l^R(P_G \diamond \Delta_G)$). Then these requirements do not appear in any proposal derived from $\star_d^R(P_G \diamond \Delta_G)$ (resp. $\star_l^R(P_G \diamond \Delta_G)$). Thus, we focus on the viewpoints involved in the inconsistencies rather than all the viewpoints. Suppose that $v_1, \cdots, v_m$ are the viewpoints involved in the inconsistencies and $R(v_1, \cdots, v_m) \in R$, let Δ be a multiset of all the requirements of viewpoints $v_1, \cdots, v_m$. Then we use $\Delta_G = \Delta$ and $\mu = R(v_1, \cdots, v_m)$ instead of Δ_G and R in the framework above, respectively.

Based on the prioritized merging operators, the proposals for handling inconsistent requirements can be derived as follows:

- If we use $\star_d$ as prioritized merging operator, then we may derive a set of proposals for inconsistency handling, denoted $\Pi_d = \{\pi(S) | S \in \star_d^\mu(P_G \diamond \Delta_G)\}$, as follows:
  - *for each S, $\pi(S)$ is a proposal that the requirements outside S should be abandoned, i.e. $\pi(S) : P_G \diamond \Delta_G \ - \ P_G \diamond \Delta_G \backslash S$.*
- If we use $\star_l$ as prioritized merging operator, then we may derive a set of proposals for inconsistency handling, denoted $\Pi_l = \{\pi(S) | S \in \star_l^\mu(P_G \diamond \Delta_G)\}$, as follows:
  - *for each S, $\pi(S)$ is a proposal that the requirements outside S should be abandoned.*

Note that a requirement statement α being abandoned by viewpoint v_i is only referred to as disappearance of α in the modification of Δ_i. That is, v_i may delete α or replace α by other new requirements.

The proposals derived from the result of prioritized merging $\star^\mu(P_G \diamond \Delta_G)$ provide possible ways of modifying the distributed requirements specification. We need to argue that each of $\Pi_\star$ is the most appropriate to modifying $\star^\mu(P_G \diamond \Delta_G)$ in some sense from the globally perspective.

Given an inconsistent collection of globally prioritized requirements, $P_G \diamond \Delta_G$, let Π be a set of all the possible proposals to modifying $P_G \diamond \Delta_G$. An intuitive criteria of appropriateness of an individual proposal $\pi \in \Pi$ to modification is precedence of the consistent subset of $P_G \diamond \Delta_G$ resulted from π. Let $\pi(S)$ be a proposal derived from $S \in Cons^R(P_G \diamond \Delta_G)$. Then S is the result of modification of $P_G \diamond \Delta_G$ by using the proposal $\pi(S)$. We define a relation on Π termed as *more appropriate than* as follows.

Definition 3.1. (The relation of more appropriate than)
Let Π be a set of all the possible proposals to modifying $P_G \diamond \Delta_G$. A binary relation on Π, denoted $\ll_G$, is defined as follows:

$$\forall \pi(S_1), \pi(S_2) \in \Pi, \pi(S_1) \ll_G \pi(S_2) \text{ iff } S_1 \prec_\star S_2,$$

where $\prec_\star$ is an ordering relation over $Cons^\mu(P_G \diamond \Delta_G)$ adopted in prioritized merging. Furthermore, we say that π_1 is *more appropriate than* π_2 to modifying $P_G \diamond \Delta_G$ if $\pi_1 \ll_G \pi_2$.

Note that the relation *more appropriate than* on Π is defined with regard to the ordering relation on $Cons(P_G \diamond \Delta_G)$.

Example 3.1. Suppose that $P_G \diamond \Delta_G = \langle\{\alpha\}, \{\beta\}, \{\neg\alpha\}\rangle$. Consider $S_1 = \langle\{\alpha\}, \{\beta\}, \emptyset\rangle$ and $S_2 = \langle\emptyset, \{\beta\}, \{\neg\alpha\}\rangle$. Obviously, $S_1, S_2 \in Cons(P_G \diamond \Delta_G)$. Let $\pi(S_1)$ and $\pi(S_2)$ be proposals for resolving inconsistency in $P_G \diamond \Delta_G$ derived from S_1 and S_2, respectively. Then the proposal $\pi(S_1)$ argues that $\neg\alpha$ should be abandoned so as to resolve inconsistency, i.e, the result of modification is S_1. In contrast, $\pi(S_2)$ argues that α should be abandoned. Evidently, $\pi(S_1)$ is more appropriate than $\pi(S_2)$ with regard to $\prec_l$ since $S_1 \prec_l S_2$.

Evidently, we can conclude the following result.

Proposition 3.1. Let Π be a set of all the possible proposals to modifying $P_G \diamond \Delta_G$. Let $\Pi_\star$ be a set of proposals derived from the prioritized merging result $\star^\mu(P_G \diamond \Delta_G)$. Then

$$\forall \pi_\star \in \Pi_\star, \text{ there is no } \pi \in \Pi \text{ such that } \pi \ll_G \pi_\star.$$

That is, each $\pi_\star$ may be viewed as the most appropriate to modifying $P_G \diamond \Delta_G$ with regard to $\prec_\star$.

Identifying acceptable common proposals. In the sense of $\prec_\star$, the proposals derived from the prioritized merging result may be considered as the most appropriate ones for handling inconsistency *from the global perspective*. Moreover, it is possible that at least two different proposals may be derived from the same prioritized merged result.These proposals are equivalent to each other in the sense of some ordering relation (such as $\prec_d$) from the global perspective. That is, we can not differentiate these proposals from the global perspective.

However, it does not mean that all these proposals are the most appropriate to each viewpoint involved in inconsistencies. For an individual viewpoint, different proposals may have different impact on the requirements change with regard to the viewpoint. Thus, different viewpoints may have different preferences over these proposals. The identification of acceptable common proposals for inconsistency handling also depends on the context of the inconsistency. Many factors such as the stakeholder's intention, expectation of benefit from the system-to-be and communication skills of developers have influences on making a trade-off decision. Therefore, it is needed to involve some social behaviors such as argumentation, negotiation and vote between viewpoints during the inconsistency handling process.

The proposals derived from the prioritized merging result can be considered as recommendations to viewpoints or stakeholders. Some projects developers would like to negotiate over these proposals, whilst other projects may prefer votes rather than argumentation. Whatever methods that the viewpoints adopt, identifying acceptable common proposals should consider the preference over the derived proposals of each viewpoint.

An acceptable proposal to a given viewpoint should disengage its major requirements from inconsistencies by abandoning minor requirements. Thus, for each viewpoint, the local priority of requirements plays a prominent role in making a trade-off decision about these proposals.

Let $\Pi_\star = \{\pi_1, \cdots, \pi_d\}$ be a set of derived proposals. Let $\pi_k \circ [P_i \diamond \Gamma_i]$ be a set of requirements that should be abandoned by v_i according to the proposal π_k for all $k = 1, \cdots, d$. For each viewpoint v_i, we provide a preference relation $\ll_i$ over $\Pi_\star$ from the perspective of v_i as follows:

$$\forall \pi_l, \pi_k \in \Pi_\star,\ \pi_l \ll_i \pi_k \text{ if and only if } \pi_k \circ [P_i \diamond \Gamma_i] \preceq_L \pi_l \circ [P_i \diamond \Gamma_i],$$

where $\pi_k \circ [P_i \diamond \Gamma_i] \preceq_L \pi_l \circ [P_i \diamond \Gamma_i]$ defined as

(1) $\pi_k \circ [P_i \diamond \Gamma_i] = \pi_l \circ [P_i \diamond \Gamma_i]$; or

(2) $\exists j$ such that $|(P_i \diamond \Delta_i)_{1\to j} \cap \pi_k \circ [P_i \diamond \Gamma_i]| > |(P_i \diamond \Delta_i)_{1\to j} \cap \pi_l \circ [P_i \diamond \Gamma_i]|$, and for all $p < j$, $|(P_i \diamond \Delta_i)_{1\to p} \cap \pi_k \circ [P_i \diamond \Gamma_i]| = |(P_i \diamond \Delta_i)_{1\to p} \cap \pi_l \circ [P_i \diamond \Gamma_i]|$.

Essentially, $\pi_l \ll_i \pi_k$ means that viewpoint v_i prefers π_l to π_k if the number of requirements with higher local priority to be abandoned by π_k is greater than that to be abandoned by π_l.

Based on the set of preference relations on $\Pi_\star$, $\{\ll_1, \ll_2, \cdots, \ll_m\}$, viewpoints may adopt a group decision making mechanism to identify an acceptable common proposal. In our previous paper [30], we discussed an approach to reaching an agreement over viewpoints based on combinatorial voting [31] and stakeholders goals.

Negotiation is also considered as a useful way to resolving inconsistency during the requirements stage [35, 40, 41]. In this paper, we may also consider negotiation as a group decision making mechanism in our general framework. When a proposal for handling inconsistency is presented by a viewpoint, the negotiation for a common acceptable proposal starts. For proceeding the negotiation process, we may use a negotiation meta-language for Multi-agent automated system defined in [42]. This language is richer for talking about proposals than negotiation languages designed for special scenarios [42], since it includes the following illocutions for describing the speech acts of conveying intentions:

- $request(i, j, \pi)$: a request from viewpoint v_i to viewpoint v_j for proposal π;
- $offer(i, j, \pi)$: a proposal of π from v_i to v_j;
- $accept(i, j, \pi)$: v_i accepts proposal π made by v_j;
- $reject(i, j, \pi)$: v_i rejects proposal π made by v_j;
- $withdraw(i, j)$: v_i withdraws from negotiation with v_j.

Here π is a formula of the negotiation language. If we use a predicate formula $Abandon(A, B)$ instead of $A - B$, then $\pi(S)$ can be represented as $Abandon(P_G \diamond \Delta_G, P_G \diamond \Delta_G \backslash S)$.

Generally, a negotiation begins when one agent makes an $offer$ to another, or when one makes a $request$ to another. Negotiation ceases when one agent $accept$ an $offer$ or $withdraw$ from negotiation [42]. We adopt the protocol used in [43], which has been shown that the protocol guarantees success in [42]. For example, at the k-th step of the negotiation, if v_i says $offer(i, j, \pi)$, then v_j replies $offer(j, i, \pi')$, or $accept(j, i, \pi)$, or $reject(j, i, \pi)$, or $withdraw(j, i)$.

Generally, v_i always puts forward the most preferred proposal with regard to $\ll_i$ at first. To respond to proposal π presented by v_i, viewpoint v_j needs to consider the following aspects:

- If some the most important requirements of Δ_j are involved in π, then v_j may reject the proposal π, or offer a more preferred proposal π' w.r.t $\ll_j$ to v_i, or withdraw from negotiation with v_i.

- For v_j, if the proposal π will disengage more preferred requirements of Δ_j from inconsistency by making some minor concession, then v_j may accept the proposal π, or offer a more preferred π' to v_i in general case.

However, some subjective factors may also have influence on the preference relation over $\Pi_\star$ and decision making on resolving inconsistency. It is a really context-based issue beyond this paper.

In the next two sections, we will provide two approaches to specializing the general framework.

4. A Merging-based Framework for Handling Inconsistency in the Manner of Prioritized Merging

Both the relative importance of requirements statements with regard to their supporting viewpoints and the priority of viewpoints have been paid attention in managing inconsistency in requirements engineering [35]. If we merge these different viewpoints as an overall viewpoint of the system-to-be, it is necessary to assign a relative priority to each requirements statement from an overall or global perspective. Moreover, the global priority of an individual requirements statement should be an integration of the priorities of the supporting viewpoints of the requirements statement and the local priorities of the requirements statement.

As mentioned earlier, both the priority level of each viewpoint and the local priority level of each requirements statement are qualitative values rather than numerical weights. It seems to be difficult to integrate these qualitative values directly.

However, we have presented a merging-based approach to globally prioritizing the requirements in our previous paper [32]. Informally, we transform the locally prioritized requirements specification into a knowledge profile consisting of stratified knowledge bases first. Then we get a *maxsat* dominated construction based on the merging operator presented in [20], which may be considered as a globally prioritized requirements specification in some sense. Along this line, here we further provide a mechanism for generating proposals appropriate to resolving inconsistency based on the merged result.

In this section, we give a brief overview to the merging-based approach in [32] first. Then we will discuss the problem of how to generate proposals by combining the merging-based approach and the prioritized merging. The general prioritized merging-based framework is also specialized correspondingly.

4.1. Merging an ordered knowledge profile

Each of viewpoints involved in inconsistencies may be viewed as a stratified knowledge base. The knowledge profile consisting of these knowledge bases should be ordered since viewpoints are also prioritized.

An ordered knowledge profile is a finite set E of knowledge bases with a total pre-order relation $\leq_E$ on E. Intuitively, if $K_i \leq_E K_j$ then K_i is regarded as more important than K_j. From the pre-order relation $\leq_E$ on E, E can be stratified as $E = (T_1, \cdots, T_m)$, where T_i contains all the minimal knowledge bases of set $\bigcup_{j=i}^{m} T_j$ with regard to $\leq_E$. In particular, a multiset of weighted knowledge bases [33] could be viewed as a special kind of ordered knowledge profile, in which each knowledge base is associated with a non-negative number.

The pre-order relation on E implies a difference in the relative importance of knowledge bases of E. Then we should take this difference into account in definition of the relative preference relation

$R(\omega, \omega')$. Generally, with regard to prioritization in requirements engineering, each viewpoint with a higher priority is more important than all the viewpoints with lower priorities. Thus we adopt a vector rather than a weight to capture the relative importance of each knowledge base in an ordered knowledge profile.

Given a lexicographical ordering relation $\leq$ on N^m as follows, where m ($m \geq 2$) is a natural number:

- $\forall(a_1, \cdots, a_m), (b_1, \cdots, b_m) \in \mathrm{N}^m$, $(a_1, \cdots, a_m) \leq (b_1, \cdots, b_m)$ if and only if $a_i = b_i$ for all i, or $\exists i$ s.t $a_i > b_i$ and $a_j = b_j$ for all $j < i$.
- Further, $(a_1, \cdots, a_m) < (b_1, \cdots, b_m)$ if and only if $(a_1, \cdots, a_m) \leq (b_1, \cdots, b_m)$ and $(b_1, \cdots, b_m) \not\leq (a_1, \cdots, a_m)$.

Based on the lexicographical ordering $\leq$ on N^m, we defined the level vector function for an ordered knowledge profile as follows:.

Definition 4.1. (Level Vector Function [32])
Let $E = (T_1, \cdots, T_m)$ be an ordered knowledge profile. Level vector function s is a mapping from E to $\{0, 1\}^m$ such that $\forall K \in E$, if $K \in E_i$ ($1 \leq i \leq m$), then $s(K) = (a_1, \cdots, a_m)$, where $a_i = 1$ and $a_j = 0$ for all $j \in [1, m],\ j \neq i$.

Obviously,$K_i \leq_E K_j$ if and only if $s(K_i) \leq s(K_j)$. Moreover, the location of 1 in level vector function $s(K)$ captures the relative preference of K in E. It means $s(K)$ gives a numerical measure of the relative importance of K w.r.t $\leq_E$.

Then we presented an alternative definition of relative preference relation over interpretations as follows:

Definition 4.2. (Relative Preference Relation [32])
Let $E = \{K_1, \cdots, K_n\}$ be an ordered knowledge profile and $\{\Omega_{K_1,X_1}, \cdots, \Omega_{K_n,X_n}\}$ be a multi-set. A binary relative preference relation $R_s \subseteq \Omega \times \Omega$ is defined as

$$R_s(\omega, \omega') \text{ iff } \sum_{\Omega_{K_i,X_i}\ s.t.\ \omega \prec_i \omega'} s(K_i) < \sum_{\Omega_{K_j,X_j}\ s.t.\ \omega' \prec_j \omega} s(K_j),$$

where $\prec_i$ is the strict partial order relation induced from Ω_{K_i,X_i}.

Essentially, by introducing a level vector function s, R_s considers $\leq_E$ as well as $\prec_i$ for each i. In this section, we adopt R_s instead of R to construct a stratified merged knowledge base from an ordered knowledge profile.

Example 4.1. Consider an ordered knowledge profile $E = (K_1, K_2)$, where $K_1 = (\{p\}, \{\neg p\})$ and $K_2 = (\{\neg p\}, \{p\})$. Then the set of interpretations is $\Omega = \{\omega_1 = 1, \omega_2 = 0\}$. Then $r_{BO}(\omega)$, $r_{MO}(\omega)$ and $K^i(\omega)$ are given in Table 1.

(1) Suppose that we do not consider the pre-order relation on E. We have shown that neither $R(\omega_1, \omega_2)$ nor $R(\omega_2, \omega_1)$ holds if we adopt the *maxsat* ordering strategy [32].

However, for any $X \in \{bo, mo, lo\}$, we can get $\omega_1 \prec_{K_1,X} \omega_2$ and $\omega_2 \prec_{K_2,X} \omega_1$. Neither $R(\omega_1, \omega_2)$ nor $R(\omega_2, \omega_1)$ holds. Then $\Omega = (\{\omega_1, \omega_2\})$ signifies that there is no meaningful merged result.

Table 1. Ranks of interpretations

ω	$r_{BO}(\omega)$		$r_{MO}(\omega)$		$K(\omega)$	
	K_1	K_2	K_1	K_2	K_1	K_2
$\omega_1 = 1$	2	1	1	2	(1,0)	(0,1)
$\omega_2 = 0$	1	2	2	1	(0,1)	(1,0)

(2) If we consider the pre-order relation on E, then $s(K_1) = (1, 0)$ and $s(K_2) = (0, 1)$. Furthermore, $\omega_1 \prec_{K_1,X} \omega_2$ and $\omega_2 \prec_{K_2,X} \omega_1$, where $X \in \{bo, mo, lo\}$. So, $R_s(\omega_1, \omega_2)$ holds. The stratification of interpretations is $\Omega = (\{\omega_1\}, \{\omega_2\})$. From Proposition 2.1, we can get

(a) a best out dominated construction $K = (\{p\})$.

(b) a maxsat-dominated construction $K = (\{p\}, \{\neg p\})$.

(c) a leximin-dominated construction $K = (\{p\}, \{\neg p\})$.

These merged results are intuitive.

4.2. Specifying the general prioritized merging-based framework

If we adopt the merging-based approach mentioned above to construct the globally prioritized requirements collection, the general prioritized merging-based framework can be specified as shown in Figure 3. Informally, we first transform each requirements collection with a local prioritization to a stratified knowledge base (SKB). The relationship between corresponding viewpoints is viewed as an integrity constraint during the merging process. Then we construct a stratified merged knowledge base (SMKB) from an ordered knowledge profile consisting of these SKBs. The merged result can be considered as an overall view of these viewpoints. Moreover, the ordering relation over the merged knowledge base could be viewed as a global prioritization on merged requirements collection. Finally, we derive proposals for handling inconsistency by incorporating the stratified merged knowledge base in the manner of prioritized merging and identify acceptable common proposals.

(1). From Locally Prioritized Requirements Collections To Stratified Knowledge Bases. Let $P_i \diamond \Delta_i$ be a requirements collection of viewpoint v_i ($1 \leq i \leq n$). Then a stratified knowledge base induced by $P_i \diamond \Delta_i$, denoted K_i, is defined as follows:

- $K_i = \Delta_i$;
- A total pre-order relation $\preceq_i$ on K_i is presented as :

$$\forall \alpha,\ \beta \in K_i,\ \alpha \preceq_i \beta \text{ iff } P_i(\alpha) \leq P_i(\beta).$$

- K_i is stratified as $K_i = (S_{i_1}, \cdots, S_{i_m})$, where $S_{i_1}, \cdots, S_{i_m}$ is given by deleting all $\emptyset$ from $\Delta_i^1, \cdots, \Delta_i^{m_i}$.

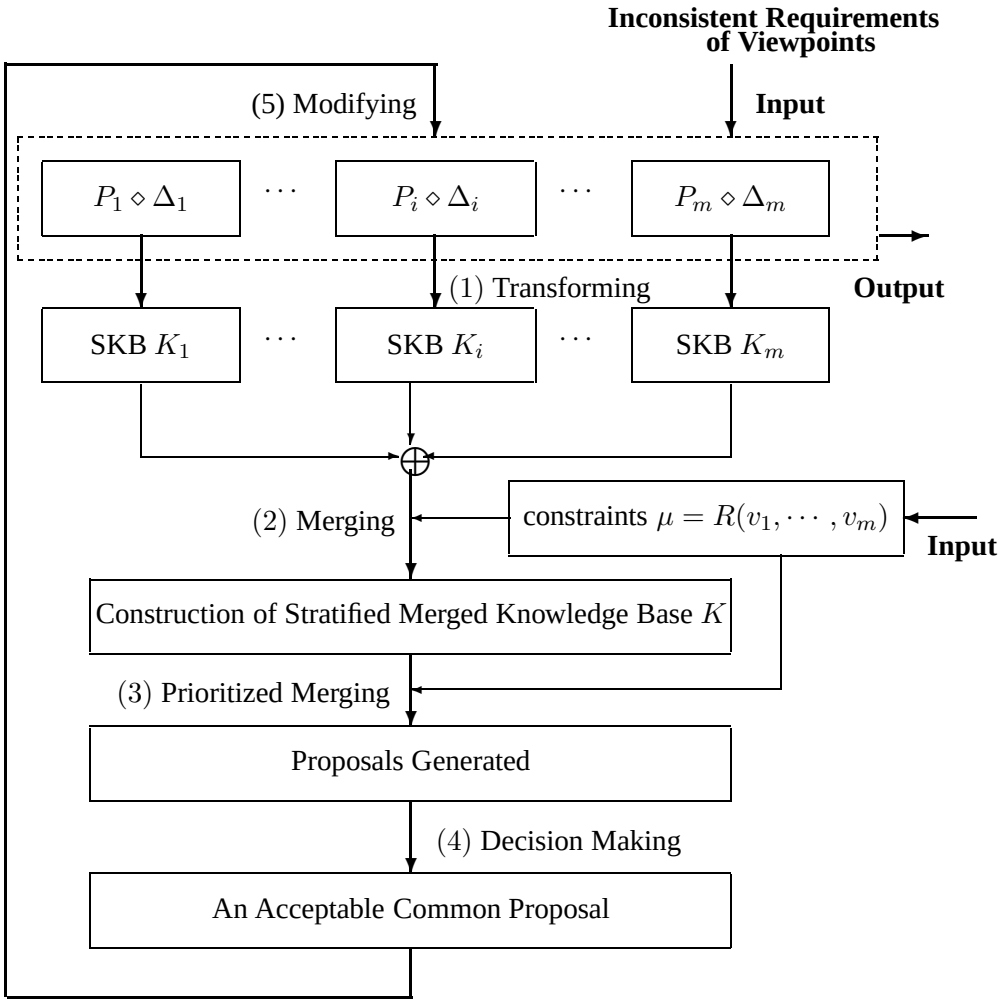

Figure 3. A Merging-based Framework to Handling Inconsistent Requirements with Local Priorities

(2). Constructing A Stratified Merged Knowledge Base. Suppose that v_1,$\cdots$, v_m are the viewpoints involved in inconsistency. Let $E = \{K_1, \cdots, K_m\}$ be a knowledge profile, where K_i is the stratified knowledge base induced by $P_i \diamond \Delta_i$ for all $1 \leq i \leq m$. Let Ω be the set of interpretations. Then

- we define a total pre-order relation $\leq_E$ on E as follows:

$$K_i \leq_E K_j \text{ iff } P_V(v_i) \leq P_V(v_j).$$

  Then we compute the level vector function s based on the stratification of E w.r.t $\leq_E$.

- Let $\mu = R(v_1, \cdots, v_m)$ and $\Omega^\mu = \{\omega \in \Omega, \omega \models \mu\}$, that is, we consider the relationship between viewpoints as an integrity constraint.

- Given an ordering strategy X (such as *best out*, *maxsat*, and *leximin*), find $\Omega^\mu_{K_i,X}$ for all i.

- Based on $\{\Omega^{\mu}_{K_1,X}, \cdots, \Omega^{\mu}_{K_m,X}\}$, construct a stratification of interpretations $\Omega^{\mu} = (\Omega^{\mu}_1, \cdots, \Omega^{\mu}_k)$ by using relative preference relation R_s over Ω^{μ}.

- Get X dominated construction K based on Proposition 2.1 .

(3). Deriving Proposals To Handling Inconsistency. Here, we explain how to combine the merging-based approach and the prioritized merging techniques to derive proposals.

Let $K = (S_1, \cdots, S_k)$ be an X-dominated construction extracted from an ordered knowledge profile E. Then the preference relation $\preceq_K$ on K describes the relative importance of requirements from a global perspective. Note that K does not always consist of all the requirements statements of $\cup^m_{i=1}(\cup K_i)$ since we adopt the syntax-based merging operator. That is, it is possible that there exists some requirements not appearing in K. For example, consider Example 2.1 again, $\neg q \in K_2$ but $\neg q \notin K_S^{lo,\Omega}$. We need to extend K to the set of all the requirements.

If there exists some requirements not appearing in K, i.e. $\cup^m_{i=1}(\cup K_i) \setminus \cup^k_{i=1} S_i \neq \emptyset$, then we define an extension of K, denoted K^*, as

$$K^* = (S_1, \cdots, S_k, S_{k+1}),$$

where $S_{k+1} = \cup^m_{i=1}(\cup K_i) \setminus \cup^k_{i=1} S_i$ and the ordering relation $\preceq_{K^*}$ is defined as

- $\forall \alpha, \beta \in K$, $\alpha \preceq_{K^*} \beta$ iff $\alpha \preceq_K \beta$;

- $\forall \alpha \in K$, $\alpha \preceq_{K^*} \psi$ and $\psi \not\preceq_{K^*} \alpha$ for all $\psi \in S_{k+1}$;

- $\forall \psi, \phi \in S_{k+1}$, $\phi \preceq_{K^*} \psi$ and $\psi \preceq_{K^*} \phi$.

The ordering relation $\preceq_{K^*}$ means that requirements in S_{k+1} are less preferred than requirements in K. By the convention, we define $K^* = K$ if $\cup^m_{i=1}(\cup K_i) = \cup^k_{i=1} S_i$. Essentially, K^* consists of all the requirements with different priorities involved in the inconsistencies. Moreover, from a global perspective, $\preceq_{K^*}$ provides a preference relation over all the requirements statements involved in the inconsistencies. In this sense, the globally prioritized requirements collection is

$$P_G \diamond \Delta_G = K^*.$$

Finally, we may derive some proposals from $\star^{\mu}_d(K^*)$(or $\star^{\mu}_l(K^*)$) as mentioned in Section 3. The rest of this framework is the same as that of the general framework.

Now we give an example to illustrate the merging-based approach.

Example 4.2. Consider $[P_1 \diamond \Delta_1, P_2 \diamond \Delta_2, P_3 \diamond \Delta_3, R]$, where

$$\begin{aligned}
&\Delta_1 = \{\alpha, \beta\}, P_1 : \Delta_1 \longmapsto L^3 \text{ such that } P_1(\alpha) = l_1,\ P_1(\beta) = l_3,\\
&\Delta_2 = \{\alpha, \gamma\}, P_2 : \Delta_2 \longmapsto L^4 \text{ such that } P_2(\alpha) = l_1,\ P_2(\gamma) = l_2,\\
&\Delta_3 = \{\phi, \neg\gamma\}, P_3 : \Delta_3 \longmapsto L^3 \text{ such that } P_3(\phi) = l_1,\ P_3(\neg\gamma) = l_3,\\
&R = \{R(v_1, v_2, v_3)\},\ \text{where } R(v_1, v_2, v_3) = \{\alpha \leftrightarrow \phi\}.
\end{aligned}$$

Table 2. Ranks of interpretations given by the best out ordering strategy

ω	K_1	K_2	K_3
1111	$+\infty$	$+\infty$	2
1101	$+\infty$	2	$+\infty$
1011	2	$+\infty$	2
1001	2	2	$+\infty$
0110	1	1	1
0100	1	1	1
0010	1	1	1
0000	1	1	1

Note that viewpoint v_2 adopts the scale of priority $L^4 = \{l_1, l_2, l_3, l_4\}$, whilst v_1 and v_3 adopt the scale of priority $L^3 = \{l_1, l_2, l_3\}$. If we adopt the merging-based approach, then the stratified knowledge bases induced by the three viewpoints are

$$K_1 = (\{\alpha\}, \{\beta\}),\ K_2 = (\{\alpha\}, \{\gamma\}),\ \text{and}\ K_3 = (\{\phi\}, \{\neg\gamma\}).$$

We denote each model by a bit vector consisting of truth values of (α, β, γ, ϕ). Then $\mu = \{\alpha \leftrightarrow \phi\}$ and

$$\Omega^\mu = \{\omega_1 = 1111, \omega_2 = 1101, \omega_3 = 1011, \omega_4 = 1001$$
$$\omega_5 = 0110, \omega_6 = 0100, \omega_7 = 0010, \omega_8 = 0000\}.$$

Suppose that we use the best out ordering strategy. r_{BO} is given by Table 2.

Then the stratification of Ω is given as follows:

$$\Omega^\mu = (\{\omega_1, \omega_2\}, \{\omega_3, \omega_4\}, \{\omega_5, \omega_6, \omega_7, \omega_8\}).$$

Let $S = \Delta_1 \cup \Delta_2 \cup \Delta_3$, then according to Proposition 2.1, we may get a best-out dominated construction

$$K = (\{\alpha, \phi\}, \{\beta\}).$$

Furthermore, we can get

$$P_G \diamond \Delta_G = K^* = (\{\alpha, \phi\}, \{\beta\}, \{\gamma, \neg\gamma\}).$$

Suppose that we adopt the leximin operator $\star_l$ in prioritized merging, then

$$\star_l^\mu(K^*) = \{S_1 = (\{\alpha, \phi\}, \{\beta\}, \{\neg\gamma\}),\ S_2 = (\{\alpha, \phi\}, \{\beta\}, \{\gamma\})\}.$$

From these prioritized merged results, we can derive the following proposals for handling inconsistency in $P_G \diamond \Delta_G$:

- $\pi_l(S_1)$ means that γ should be abandoned by v_2, i.e. $\Delta_2 - \{\gamma\}$;
- But $\pi_l(S_2)$ means that v_3 should abandon $\neg\gamma$, that is, $\Delta_3 - \{\neg\gamma\}$.

Generally, since v_3 assigned the lowest priority to $\neg\gamma$, individual viewpoints would expect that v_3 makes some concession in negotiation over $\{\pi(S_1), \pi(S_2)\}$. Then it seems that $\pi(S_2)$ rather than $\pi(S_1)$ will be accepted by viewpoints.

5. A Priority Vector-based Framework for Handling Inconsistency in the Manner of Prioritized Merging

The merging-based approach only assumes that each original knowledge base to be merged is stratified. It does not consider if all the knowledge bases use the same scale of prioritization. Thus it is appropriate for merging viewpoints, especially for merging viewpoints with different scales of local prioritization.

However, grouping requirements into three priority categories is viewed as a common approach to prioritization in requirements engineering [17]. That is, most stakeholders are accustomed to use typical 3-level scale in local prioritization. Then it is worth considering the setting of all the viewpoints using the same scale of local prioritization. In this section, we provide a more concise approach to globally prioritizing requirements for such cases.

We start our approach with an alternative representation of the local priority levels. Suppose that L^{m_i} be the scale of priority levels adopted by viewpoint v_i for all i. Let $\mathrm{I}_{m_i \times m_i}$ be the unit matrix and $\overrightarrow{a}_k$ the k-th row vector of $\mathrm{I}_{m_i \times m_i}$. For each viewpoint v_i, we provide an alternative prioritization mapping P_i' as follows:

$$\forall \alpha \in \Delta_i,\ P_i'(\alpha) = \overrightarrow{a}_k \text{ iff } P_i(\alpha) = l_k.$$

Obviously, $P_i'(\alpha) \leq P_i'(\alpha)$ if and only if $P_i(\alpha) \leq P_i(\alpha)$. In the rest of this paper, we also term P_i' *the priority vector function* of viewpoint v_i. For example, consider $\Delta_1 = \{\alpha, \beta\}$ and $P_1(\alpha) = l_1$, $P_1(\beta) = l_3$ under L^3, then $P_1'(\alpha) = (1, 0, 0)$ and $P_1'(\beta) = (0, 0, 1)$.

The global priority of an individual requirements statement depends on the relative importance of its supporting viewpoints as well as its local priority with regard to each supporting viewpoint. However, for each requirements statement, the relative importance of its supporting viewpoints plays a dominating role in prioritizing this statement with regard to the whole system-to-be.

Let P_V be the prioritization mapping on V. We assume that all the viewpoints at the same priority level adopt the same scale of priority levels to prioritize the requirements. That is, $L^{m_i} = L^{m_j}$ if $P_V(v_i) = P_V(v_j)$. Let $V_{In} = \{v_1, \cdots, v_m\}$ be the set of viewpoints involved in a given inconsistency. Suppose that V_{In} can be stratified as $P_V \diamond V_{In} = \langle V(1), \cdots, V(r)\rangle$. For each i ($1 \leq i \leq r$), we construct $\sigma_i = P_{G_i} \diamond \Gamma_i$ as follows:

$$\Gamma_i = \bigcup_{v_j \in V(i)} \Delta_j; \quad \forall \alpha \in \Gamma_i,\ P_{G_i}(\alpha) = \sum_{\alpha \in \Delta_j, v_j \in V(i)} P_j'(\alpha).$$

Essentially, the k-th component of $P_{G_i}(\alpha)$ is the number of supporting viewpoints at level i of α, which assigns k-th priority level to α. In this sense, $P_{G_i}(\alpha)$ gives the relative importance with regard to all the viewpoints at level i.

Then we construct a requirements collection with a global prioritization as follows:

$$P_G \diamond \Delta_G = \sigma_1 \circ \sigma_2 \circ \cdots \circ \sigma_r,$$

where $\circ$ is the concatenation operator.

If we adopt the priority vector function of each viewpoint to construct the globally prioritized requirements collection, the general prioritized merging-based framework can be specified as shown in Figure 4, in which the approach to constructing a globally prioritized requirements collection from inconsistent viewpoints has been detailed as follows:

(1) *Vectorizing the local priority*. For each viewpoint involved in inconsistencies, we provide an equivalent priority vector function to the original prioritization mapping.

(2) *Stratifying the set of inconsistent viewpoints by P_V*. That is,

$$P_V \diamond V_{In} = \langle V(1), \cdots, V(r) \rangle.$$

(3) *Constructing a globally prioritized requirements collection*. Firstly, for each $V(i)$, we construct $P_{G_i} \diamond \Gamma_i$ as follows:

$$\Gamma_i = \bigcup_{v_j \in V(i)} \Delta_j;\ \forall \alpha \in \Gamma_i,\ P_{G_i}(\alpha) = \sum_{\alpha \in \Delta_j, v_j \in V(i)} P'_j(\alpha);$$

Then we concatenate all $P_{G_i} \diamond \Gamma_i$ to construct a globally prioritized requirements collection, i.e.,

$$P_G \diamond \Delta_G = (P_{G_1} \diamond \Gamma_1) \circ (P_{G_2} \diamond \Gamma_2) \circ \cdots \circ (P_{G_r} \diamond \Gamma_r).$$

The rest of this framework is the same as that of the general framework.

Given $P_G \diamond \Delta_G = \langle \Delta(1), \cdots, \Delta(l) \rangle$, if there exists α such that $\alpha \in \Delta(i) \cap \Delta(j)$, $i \neq j$, then it is evident to prove that for all $S \in \star_d^{\mu}(P_G \diamond \Delta_G)$ (or $\star_d(P_G \diamond \Delta_G)$), $\alpha \in S(i)$ if and only if $\alpha \in S(j)$. That is, we guarantee that the approach does not derive any unreasonable proposal of only abandoning α in $\Delta(i)$ (or $\Delta(j)$).

Now we give an example to illustrate the priority-vector based approaches to constructing a globally prioritized requirements collection.

Example 5.1. consider $[P_1 \diamond \Delta_1, P_2 \diamond \Delta_2, P_3 \diamond \Delta_3, P_4 \diamond \Delta_4, R]$,where

$$\begin{aligned}
&P_1 \diamond \Delta_1 = \langle \{\alpha\}, \{\beta\}, \{\gamma\} \rangle,\\
&P_2 \diamond \Delta_2 = \langle \{\beta\}, \{\alpha\}, \{\psi\} \rangle,\\
&P_3 \diamond \Delta_3 = \langle \{\gamma\}, \{\neg\beta\}, \{\alpha\} \rangle,\\
&P_4 \diamond \Delta_4 = \langle \{\beta\}, \{\alpha\}, \{\phi\} \rangle,\\
&R = \{R(v_1, v_2, v_3, v_4)\},\ \text{where } R(v_1, v_2, v_3, v_4) = \{\phi \leftrightarrow \gamma, \phi \leftrightarrow \neg\psi\}.
\end{aligned}$$

Suppose that $P_V(v_1) = P_V(v_3) = l_1$ and $P_V(v_2) = P_V(v_4) = l_2$. It means that viewpoints v_1 and v_3 are more important than v_2 and v_4.

Then $P_V \diamond V = \langle V(1), V(2), V(3) \rangle$, where $V(1) = \{v_1, v_3\}$, $V(2) = \{v_2, v_4\}$ and $V_3 = \emptyset$. Furthermore,

$$\begin{aligned}
&\Gamma_1 = \Delta_1 \cup \Delta_3 = \{\alpha, \beta, \gamma, \neg\beta\},\\
&P_{G_1}(\alpha) = P'_1(\alpha) + P'_3(\alpha) = (1,0,0) + (0,0,1) = (1,0,1),\\
&P_{G_1}(\beta) = P'_1(\beta) = (0,1,0),\\
&P_{G_1}(\gamma) = P'_1(\gamma) + P'_3(\gamma) = (0,0,1) + (1,0,0) = (1,0,1),\\
&P_{G_1}(\neg\beta) = P'_3(\neg\beta) = (0,1,0),\\
&P_{G_1}(\alpha) = P_{G_1}(\gamma) < P_{G_1}(\beta) = P_{G_1}(\neg\beta).
\end{aligned}$$

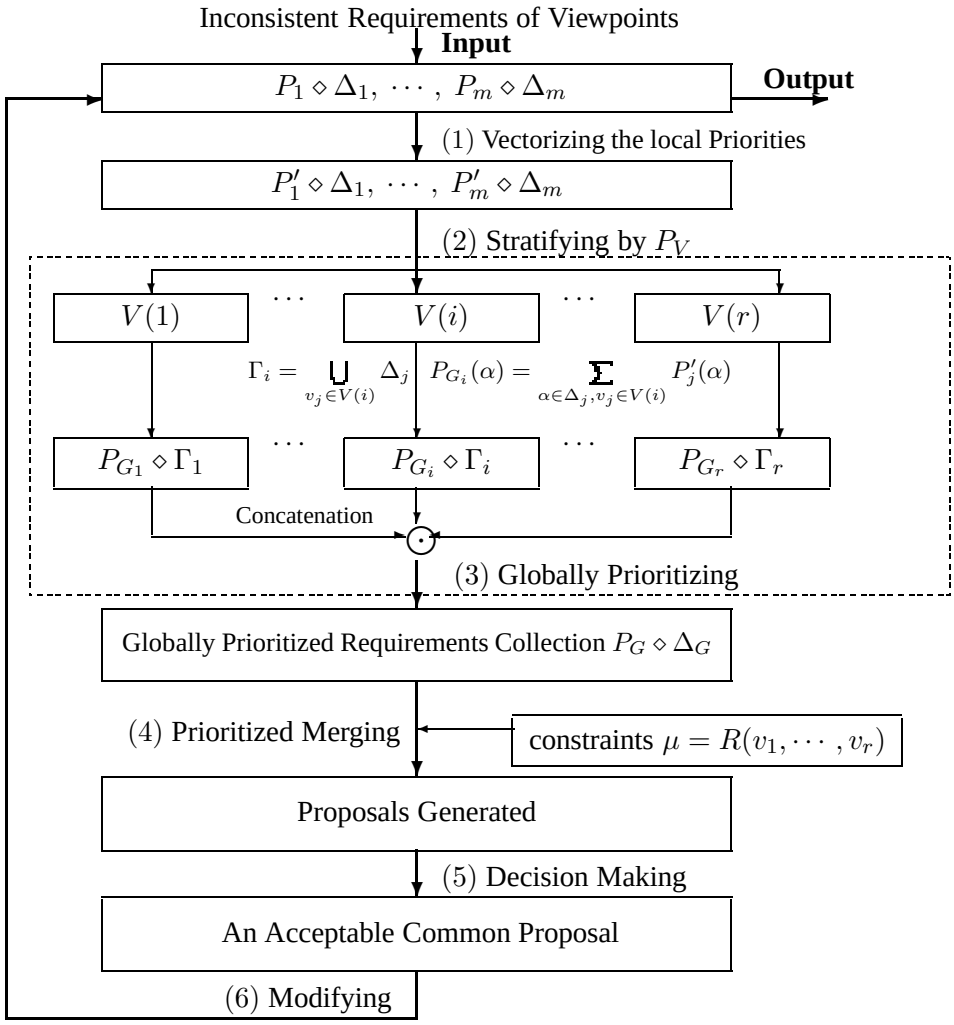

Figure 4. A Priority Vector-based Framework of Prioritized Merging-based Approaches

Then

$$\sigma_1 = P_{G_1} \diamond \Gamma_1 = \langle \{\alpha, \gamma\}, \{\beta, \neg\beta\} \rangle.$$

Similarly, we can get

$$\sigma_2 = P_{G_2} \diamond \Gamma_2 = \langle \{\beta\}, \{\alpha\}, \{\psi, \phi\} \rangle.$$

Evidently,

$$\sigma_3 = \emptyset.$$

Finally, we get a requirements specification with the global prioritization as follows:

$$P_G \diamond \Delta_G = \sigma = \sigma_1 \circ \sigma_2 \circ \sigma_3 = \langle \{\alpha, \gamma\}, \{\beta, \neg\beta\}, \{\beta\}, \{\alpha\}, \{\psi, \phi\} \rangle.$$

Note that β appears in both $\sigma(2) = \{\beta, \neg\beta\}$ and $\sigma(3) = \{\beta\}$. $\beta \in \sigma(2)$ signifies that there exists at least one supporting viewpoint of β having the highest priority, whilst $\beta \in \sigma(3)$ indicates that there also exists

at least one supporting viewpoint of β having the priority level of Medium. It embodies the dominant role of the relative importance of viewpoints in globalizing the requirements.

If we consider $\mu = R(v_1, v_2, v_3, v_4)$, then

$$\star_d^{\mu}(P_G \diamond \Delta_G) = \{S_1, S_2\}, \text{ where}$$
$$S_1 = \langle\{\alpha, \gamma\}, \{\beta\}, \{\beta\}, \{\alpha\}, \{\phi\}\rangle,\ S_2 = \langle\{\alpha, \gamma\}, \{\neg\beta\}, \emptyset, \{\alpha\}, \{\phi\}\rangle,$$

Correspondingly, the proposal derived from S_1 is

$$\pi_d(S_1) : P_G \diamond \Delta_G\ -\ \langle\emptyset, \{\neg\beta\}, \emptyset, \emptyset, \{\psi\}\rangle$$

This proposal means that viewpoints v_3 and v_2 should abandon $\neg\beta$ and ψ, respectively. In contrast, the second proposal is

$$\pi_d(S_2) : P_G \diamond \Delta_G\ -\ \langle\emptyset, \{\beta\}, \{\beta\}, \emptyset, \{\psi\}\rangle$$

This proposal means that viewpoints v_1, v_2 and v_3 should abandon β. Moreover, v_2 should abandon ψ.

The four viewpoints show different preferences over $\{\pi(S_1), \pi(S_2)\}$:

- for v_i $(i = 1, 2, 4)$, $\pi(S_1) \ll_i \pi(S_2)$;
- for v_3, $\pi(S_2) \ll_3 \pi(S_1)$.

If they vote for an acceptable common proposal, then $\pi(S_1)$ should be considered as a winner.

If they negotiate over $\{\pi(S_1), \pi(S_2)\}$, v_2 and v_4 seem to make no concession in abandoning β since β is the most preferred requirement for v_2 and v_4, respectively. That is, it is very possible that they cannot reach an agreement on $\pi(S_2)$. But if v_3 makes some concession in abandoning $\neg\beta$, it is reasonable that v_2 accepts $\pi(S_1)$ since $\pi(S_1)$ disengages the most preferred requirement β of v_2 from inconsistency by making minor concession in abandoning ψ.

If we adopt the leximin operator in prioritized merging, we may get $\star_d^{\mu}(P_G \diamond \Delta_G) = \{S_1\}$ and only one derived proposal $\pi_l(S_1) = \pi_d(S_1)$.

6. A Case Study

This section uses a close residential area management system as an example to illustrate the feasibility of the prioritized merging-based framework. This case study shows that the method can be conjuncted with a host of other techniques in RE for handling inconsistency in the Viewpoints framework.

Example 6.1. Using the Viewpoints framework to elicit the requirements of a computer-aided close residential area management system. The requirements document contains:

- Viewpoint 1: Vehicles Entrance Manager
  - (a) The vehicles without special authorization of the Management Board of the residential area can not be allowed to enter the area;
  - (b) The system should trigger warning alarm if a vehicle without authorization enters the area.

Viewpoint 1 assigned the priority level of *High* to both (a) and (b).

- Viewpoint 2: Emergency Manager

  (c) The fire engine should be viewed as the vehicle of emergency;

  (d) The vehicle of emergency such as fire engines can enter the area;

  (e) The vehicle of emergency need not to be authorized by the Management Board of the residential area in advance.

  (f) The system should trigger warning alarm when a vehicle without authorization enters the area.

  Viewpoint 2 assigned the priority level of *High* to both (c) and (d). (e) and (f) were assigned to the levels of *Medium* and *low*, respectively.

- Viewpoint 3: Authorization Manager

  (g) The fire engine should be viewed as a special kind of vehicle;

  (h) The special vehicle can enter the area;

  (i) The special vehicle does not need to be authorized;

  (j) The system should not trigger warning alarm if the special vehicle enters the area.

  Viewpoint 3 assigned the priority level of *High* to both (g) and (h). (i) and (j) were assigned to *Medium* and *Low*, respectively.

Furthermore, Viewpoints 2 and 3 were assigned the priority level of *High*. Viewpoint 1 was assigned the priority level of *Medium*.

We use logical representation to describe these requirements information. Suppose that we use

- $aut(X)$ to denote that X is authorized by the Management Board of the residential area;
- $ent(X)$ to denote that X can enter the residential area;
- $eme(X)$ to denote that X is a vehicle for emergency;
- $ala(X)$ to denote that the system triggers alarm if X enters the area;
- $spe(X)$ to denote that X is a special vehicle.
- constant FE to denote the fire engine.

If we consider a scenario about the entrance of the fire engine, then the requirements of the system-to-be can be described as:

$P_V(v_1) = l_2,\ P_V(v_2) = P_V(v_3) = l_1,$
$\Delta_1 = \{\neg aut(FE) \to \neg ent(FE),\ \neg aut(FE) \to ala(FE)\},$
$P_1(\neg aut(FE) \to \neg ent(FE)) = l_1,\ P_1(\neg aut(FE) \to ala(FE)) = l_1.$
$\Delta_2 = \{eme(FE),\ eme(FE) \to ent(FE),\ eme(FE) \to \neg aut(FE), \neg aut(FE) \to ala(FE)\},$
$P_2(eme(FE)) = P_2(eme(FE) \to ent(FE)) = l_1,$
$P_2(eme(FE) \to \neg aut(FE)) = l_2,\ P_2(\neg aut(FE) \to ala(FE)) = l_3.$
$\Delta_3 = \{spe(FE),\ spe(FE) \to ent(FE),\ spe(FE) \to \neg aut(FE),\ spe(FE) \to \neg ala(FE)\},$
$P_3(spe(FE)) = P_3(\ spe(FE) \to ent(FE)) = l_1,$
$P_3(spe(FE) \to \neg aut(FE)) = l_2,\ P_3(spe(FE) \to \neg ala(FE)) = l_3.$

Suppose that requirements analysts provide the following relation between the viewpoints:

$$R = \{R(v_1, v_2, v_3)\},\ \text{where } R(v_1, v_2, v_3) = \{spe(FE) \leftrightarrow eme(FE)\}.$$

Then we get the requirements specification $[P_1 \diamond \Delta_1, P_2 \diamond \Delta_2, P_3 \diamond \Delta_3, R]$, where

$$\begin{aligned}
P_1 \diamond \Delta_1 &= \langle\{\neg aut(FE) \to \neg ent(FE),\ \neg aut(FE) \to ala(FE)\}, \emptyset, \emptyset\rangle,\\
P_2 \diamond \Delta_2 &= \langle\{eme(FE),\ eme(FE) \to ent(FE)\},\ \{eme(FE) \to \neg aut(FE)\},\\
&\quad \{\neg aut(FE) \to ala(FE)\}\rangle,\\
P_3 \diamond \Delta_3 &= \langle\{spe(FE),\ spe(FE) \to ent(FE)\},\ \{spe(FE) \to \neg aut(FE)\},\\
&\quad \{spe(FE) \to \neg ala(FE)\}\rangle.
\end{aligned}$$

Moreover, we draw the following inconsistencies:

$$\Delta_1 \cup \Delta_2 \cup \Delta_3 \cup R(v_1, v_2, v_3) \vdash ent(FE) \wedge \neg ent(FE);$$
$$\Delta_1 \cup \Delta_2 \cup \Delta_3 \cup R(v_1, v_2, v_3) \vdash ala(FE) \wedge \neg ala(FE).$$

(A) First, we use the merging-based approach to constructing a globally prioritized requirements collection. We get the following stratified knowledge bases induced by v_1, v_2, and v_3, respectively.

$$\begin{aligned}
K_1 &= (\{\neg aut(FE) \to \neg ent(FE),\ \neg aut(FE) \to ala(FE)\}),\\
K_2 &= (\{eme(FE),\ eme(FE) \to ent(FE)\},\ \{eme(FE) \to \neg aut(FE)\},\\
&\quad \{\neg aut(FE) \to ala(FE)\}),\\
K_3 &= (\{spe(FE),\ spe(FE) \to ent(FE)\},\ \{spe(FE) \to \neg aut(FE)\},\\
&\quad \{spe(FE) \to \neg ala(FE)\}).
\end{aligned}$$

The integrity constraint is $\mu = \{eme(FE) \leftrightarrow spe(FE)\}$. We denote each model by a bit vector consisting of truth values of ($eme(FE)$, $ent(FE)$, $aut(FE)$, $ala(FE)$, $spe(FE)$). Then

$$\begin{aligned}
\Omega^\mu = \{&\omega_1 = 11111, \omega_2 = 11101, \omega_3 = 11011, \omega_4 = 11001, \omega_5 = 10111, \omega_6 = 10101,\\
&\omega_7 = 10011, \omega_8 = 10001, \omega_9 = 01110, \omega_{10} = 01100, \omega_{11} = 01010, \omega_{12} = 01000,\\
&\omega_{13} = 00110, \omega_{14} = 00100, \omega_{15} = 00010, \omega_{16} = 00000.\}.
\end{aligned}$$

The level vector function s of $E = (\{K_2, K_3\}, \{K_1\})$ is

$$s(K_2) = s(K_3) = (1, 0),\ s(K_1) = (0, 1).$$

Suppose that we use the *best out* ordering strategy, then the corresponding ranks of interpretations are given in Table 3.

Table 3. The ranks of interpretations

ω	$r_{BO}(\omega)$		
	K_1	K_2	K_3
$\omega_1 = 11111$	$+\infty$	2	2
$\omega_2 = 11101$	$+\infty$	2	2
$\omega_3 = 11011$	1	$+\infty$	3
$\omega_4 = 11001$	1	3	$+\infty$
$\omega_5 = 10111$	$+\infty$	1	1
$\omega_6 = 10101$	$+\infty$	1	1
$\omega_7 = 10011$	$+\infty$	1	1
$\omega_8 = 10001$	1	1	1
$\omega_9 = 01110$	$+\infty$	1	1
$\omega_{10} = 01100$	$+\infty$	1	1
$\omega_{11} = 01010$	1	1	1
$\omega_{12} = 01000$	1	1	1
$\omega_{13} = 00110$	$+\infty$	1	1
$\omega_{14} = 00100$	$+\infty$	1	1
$\omega_{15} = 00010$	$+\infty$	1	1
$\omega_{16} = 00000$	1	1	1

Then the stratification of interpretations is

$$\Omega^{\mu} = (\{\omega_3, \omega_4\}, \{\omega_1, \omega_2\}, \{\omega_5, \omega_6, \omega_7, \omega_8, \omega_9, \omega_{10}, \omega_{11}, \omega_{12}\omega_{13}, \omega_{14}, \omega_{15}, \omega_{16}\}).$$

We get the *best-out* construction

$$\begin{aligned} K \ = \ & (\{eme(FE),\ eme(FE) \rightarrow ent(FE),\ spe(FE),\ spe(FE) \rightarrow ent(FE)\}, \\ & \{\ eme(FE) \rightarrow \neg aut(FE), spe(FE) \rightarrow \neg aut(FE)\}). \end{aligned}$$

We extend K to K^* :

$$\begin{aligned} K^* \ = \ & (\{eme(FE),\ eme(FE) \rightarrow ent(FE),\ spe(FE),\ spe(FE) \rightarrow ent(FE)\}, \\ & \{\ eme(FE) \rightarrow \neg aut(FE), spe(FE) \rightarrow \neg aut(FE)\}, \\ & \{\ spe(FE) \rightarrow \neg ala(FE), \neg aut(FE) \rightarrow \neg ent(FE),\ \neg aut(FE) \rightarrow ala(FE)\}). \end{aligned}$$

Then K^* may be viewed as the globally prioritized requirements specification, i.e., $K^* = P_G \diamond \Delta_G$. If we use the discrimin operater $\star_d$, then $\star_d^{\mu}(P_G \diamond \Delta_G) = \{S_1, S_2\}$, where

$$\begin{aligned} S_1 &= \langle \{eme(FE),\ eme(FE) \rightarrow ent(FE),\ spe(FE),\ spe(FE) \rightarrow ent(FE)\}, \\ &\quad \{\ eme(FE) \rightarrow \neg aut(FE), spe(FE) \rightarrow \neg aut(FE)\}, \\ &\quad \{\ spe(FE) \rightarrow \neg ala(FE)\}\rangle. \\ S_2 &= \langle \{eme(FE),\ eme(FE) \rightarrow ent(FE),\ spe(FE),\ spe(FE) \rightarrow ent(FE)\}, \\ &\quad \{\ eme(FE) \rightarrow \neg aut(FE), spe(FE) \rightarrow \neg aut(FE)\}, \\ &\quad \{\ \neg aut(FE) \rightarrow ala(FE)\}\rangle. \end{aligned}$$

Further, we derive two proposals $\pi(S_1)$ and $\pi(S_2)$ for resolving inconsistencies as follows:

$$\begin{aligned} \pi(S_1): &\quad P_G \diamond \Delta_G - \langle \emptyset, \emptyset, \emptyset, \{\neg aut(FE) \rightarrow \neg ent(FE),\ \neg aut(FE) \rightarrow ala(FE)\}\rangle, \\ \pi(S_2): &\quad P_G \diamond \Delta_G - \langle \emptyset, \emptyset, \emptyset, \{\neg aut(FE) \rightarrow \neg ent(FE),\ spe(FE) \rightarrow \neg ala(FE)\}\rangle. \end{aligned}$$

Essentially, $\pi(S_1)$ means that v_1 should abandon all the requirements, and meanwhile v_2 should abandon $\neg aut(FE) \rightarrow ala(FE)$. $\pi(S_2)$ means that v_1 should abandon $\neg aut(FE) \rightarrow \neg ent(FE)$, and meanwhile v_3 should abandon $spe(FE) \rightarrow \neg ala(FE)$. Although both $\pi(S_1)$ and $\pi(S_2)$ are the most appropriate to resolving inconsistencies from the global perspective, the three viewpoints have different preferences over $\{\pi(S_1), \pi(S_2)\}$:

$$\begin{aligned} &\text{for } v_1:\ \pi(S_2) \ll_1 \pi(S_1); \\ &\text{for } v_2:\ \pi(S_2) \ll_2 \pi(S_1); \\ &\text{for } v_3:\ \pi(S_1) \ll_3 \pi(S_2). \end{aligned}$$

Finally, the viewpoints may negotiate over $\{\pi(S_1), \pi(S_2)\}$ as follows:

(1) At the beginning of negotiation, v_3 puts forward $\pi(S_1)$ to v_1 and v_2:

$$offer(v_3, v_1, \pi(S_1)), \quad offer(v_3, v_2, \pi(S_1)).$$

(2) Since both v_1 and v_2 prefer $\pi(S_2)$ to $\pi(S_1)$, then they reject the proposal $\pi(S_1)$:

$$reject(v_1, v_3, \pi(S_1)), \quad reject(v_2, v_3, \pi(S_1)).$$

(3) v_3 puts forward $\pi(S_2)$ to v_1 and v_2 again:

$$offer(v_3, v_1, \pi(S_2)), \quad offer(v_3, v_2, \pi(S_2)).$$

(4) v_1 and v_2 may accept the proposal $\pi(S_2)$:

$$accept(v_1, v_3, \pi(S_2)), \quad accept(v_2, v_3, \pi(S_2)).$$

The negotiation ends. $\pi(S_2)$ is considered as an acceptable common proposal for resolving inconsistencies.

(B) Second, we use the priority vector-based approach to constructing a globally prioritized requirements collection. Since $P_V(v_2) = P_V(v_3) = l_1$ and $P_V(v_1) = l_2$, $P_V \diamond V = \langle\{v_2, v_3\}, \{v_1\}, \emptyset\rangle$. The corresponding priority vector functions are:

$$P_1'(\neg aut(FE) \rightarrow \neg ent(FE)) = (1,0,0),\ P_1'(\neg aut(FE) \rightarrow ala(FE)) = (1,0,0).$$
$$P_2'(eme(FE)) = P_2(eme(FE) \rightarrow ent(FE)) = (1,0,0),$$
$$P_2'(eme(FE) \rightarrow \neg aut(FE)) = (0,1,0),\ P_2'(\neg aut(FE) \rightarrow ala(FE)) = (0,0,1).$$
$$P_3'(spe(FE)) = P_3(\ spe(FE) \rightarrow ent(FE)) = (1,0,0),$$
$$P_3'(spe(FE) \rightarrow \neg aut(FE)) = (0,1,0),\ P_3'(spe(FE) \rightarrow \neg ala(FE)) = (0,0,1).$$

We construct $P_{G1} \diamond \Gamma_1$ and $P_{G_2} \diamond \Gamma_2$ as follows:

$$\begin{aligned} P_{G1} \diamond \Gamma_1 &= \langle\{eme(FE),\ eme(FE) \rightarrow ent(FE),\ spe(FE),\ spe(FE) \rightarrow ent(FE)\}, \\ &\quad \{\ eme(FE) \rightarrow \neg aut(FE), spe(FE) \rightarrow \neg aut(FE)\}, \\ &\quad \{\ spe(FE) \rightarrow \neg ala(FE),\ \neg aut(FE) \rightarrow ala(FE)\}\rangle, \\ P_{G2} \diamond \Gamma_2 &= \langle\{\neg aut(FE) \rightarrow \neg ent(FE),\ \neg aut(FE) \rightarrow ala(FE)\}\rangle \end{aligned}$$

Then we construct the globally prioritized requirements collection $P_G \diamond \Delta_G$ as follows:

$$\begin{aligned} P_G \diamond \Delta_G &= P_{G1} \diamond \Gamma_1 \circ P_{G2} \diamond \Gamma_2 \\ &= \langle\{eme(FE),\ eme(FE) \rightarrow ent(FE),\ spe(FE),\ spe(FE) \rightarrow ent(FE)\}, \\ &\quad \{\ eme(FE) \rightarrow \neg aut(FE), spe(FE) \rightarrow \neg aut(FE)\}, \\ &\quad \{\ spe(FE) \rightarrow \neg ala(FE),\ \neg aut(FE) \rightarrow ala(FE)\}, \\ &\quad \langle\{\neg aut(FE) \rightarrow \neg ent(FE),\ \neg aut(FE) \rightarrow ala(FE)\}\rangle \end{aligned}$$

If we use the discrimin operater $\star_d$, then $\star_d^{\mu}(P_G \diamond \Delta_G) = \{S_1', S_2'\}$, where

$$\begin{aligned} S_1' &= \langle\{eme(FE),\ eme(FE) \rightarrow ent(FE),\ spe(FE),\ spe(FE) \rightarrow ent(FE)\}, \\ &\quad \{\ eme(FE) \rightarrow \neg aut(FE), spe(FE) \rightarrow \neg aut(FE)\}, \\ &\quad \{\ spe(FE) \rightarrow \neg ala(FE)\}, \\ &\quad \emptyset\rangle. \\ S_2' &= \langle\{eme(FE),\ eme(FE) \rightarrow ent(FE),\ spe(FE),\ spe(FE) \rightarrow ent(FE)\}, \\ &\quad \{\ eme(FE) \rightarrow \neg aut(FE), spe(FE) \rightarrow \neg aut(FE)\}, \\ &\quad \{\ \neg aut(FE) \rightarrow ala(FE)\}, \\ &\quad \{\ \neg aut(FE) \rightarrow ala(FE)\}\rangle. \end{aligned}$$

Further, we derive two proposals $\pi(S_1')$ and $\pi(S_2')$ for resolving inconsistencies as follows:

$$\pi(S_1') = \pi(S_1),\ \pi(S_2') = \pi(S_2).$$

That is, we derive the same proposals by using the second approach to constructing a globally prioritized requirements collection. The subsequent process of negotiation is the same as that stated in (A).

As mentioned earlier, v_1 abandoning $\neg aut(X) \rightarrow \neg ent(X)$ does not mean that v_1 deletes the corresponding requirements forever. For example, v_1 may change $\neg aut(X) \rightarrow \neg ent(X)$ into $\neg aut(X) \wedge \neg eme(X) \rightarrow \neg ent(X)$. Anyway, $\neg aut(X) \rightarrow \neg ent(X)$ disappears in a revised requirements set of v_1.

7. Discussion

In our prioritized merging-based framework for handling inconsistency, we

- construct a globally prioritized requirements collection by using the merging-based approach or the priority vector-based approach;
- derive proposals for handling inconsistency based on global prioritization by prioritized merging;
- define the relation *more appropriate than* $\ll$ and preference relation $\ll_i$ on the set of proposals to help viewpoints reach a reasonable agreement on the final proposal for handling inconsistency.

In this section, we argue that both the local prioritization and the global prioritization deserve consideration in handling inconsistency in the Viewpoints framework first. Then we compare the two merging-based approach with the priority vector-based approach. Finally, we point out that the relation *more appropriate than* is associated with the given relation on the set of consistent subsets of globally prioritized requirements collection.

The disagreement in the local prioritization over shared requirements often puts inconsistency handling in a dilemma. The prioritized merging-based approach to generating proposals aims to derive *the most appropriate* proposals to handling inconsistency *from the global perspective* rather than from a perspective of a particular viewpoint. But this does not mean that global prioritization is more crucial than local prioritization. We argue that both the global prioritization and the local prioritization play important roles in resolving inconsistency. In particular, the local prioritization of each viewpoint has a prominent impact on identifying an acceptable common proposal.

To globally prioritize the inconsistent requirements from different viewpoints, we provide two approaches to constructing a requirements collection with the global prioritization, including the merging-based approach and the priority vector-based approach. The only assumption of the merging-based approach is that each original requirements collection is stratified. The different viewpoints do not need to adopt the same scale of local priority levels. This conforms to the principle of heterogeneity in the Viewpoints. On the other hand, we adopt the syntax-based merging operators presented in [20] during merging process. The syntax-based merging operator aims to pick out some formulas from original knowledge bases. Then the merged result can be explained clearly. But it is possible that there is no merged result for some knowledge profiles with regard to some ordering strategy. This means that we can not get a globally prioritized requirements collection in some case. However, introducing model-based merging operators also leads to a problem of how to explain additional formulas in the merged result in terms of viewpoints demands. It seems to be a dilemma.

In contrast, the priority vector-based approach assumes that all the viewpoints with the same importance level should adopt the same scale of local priority levels. Since most stakeholders consider the three-level scale as a common scale of prioritization in requirements engineering [17], this assumption of the priority vector-based approach is meaningful in requirements engineering. Moreover, the simplicity of computation for getting the global priority of each requirements statement from its local priorities makes the priority vector-based approach intuitive and acceptable to requirements engineering.

The relation of *more appropriate than* on the set of proposals for resolving inconsistency is associated with the given relation ($\prec_d$ or $\prec_l$) on $Cons(P_G \diamond \Delta_G)$. It is a prioritized merging-based framework-sensitive term. In this sense, *the most appropriate* proposal $\pi(S)$ is referred to as the proposals disengaging the undominated element S of $Cons(P_G \diamond \Delta_G)$ with regard to $\prec_d$ or $\prec_l$ from inconsistency rather than absolutely perfect proposals. In our prioritized merging-base framework, each proposal is essentially a recommendation of abandoning some requirements. However, it is difficult to derive other kinds of proposal such as requirements change only based on the priority of requirements, since inconsistency handling is really context-sensitive as mentioned in [15, 16]. Consequently, how to generate absolutely perfect proposals to resolving inconsistency in requirements is still a challenging issue [11].

8. Related Work

Managing inconsistency in requirements has received considerable attention in requirements engineering recently. In this section, we compare the prioritized merging-based framework presented in this paper with some of closely related research.

As a common approach to fusing a set of heterogenous information, merging techniques have been adopted to manage inconsistency in the Viewpoints framework. Easterbrook et al [34] presented the multi-valued logics-based framework χbel for merging and reasoning about inconsistent viewpoints. They formalized the viewpoints as state machine models. The variables used in the state machine models are boolean variables. For example, suppose that a state s_1 in viewpoint v_1 is represented by $\{X = \mathrm{T}, Y = \mathrm{F}\}$. But in viewpoint v_2, s_1 is represented by $\{X = \mathrm{F}, Y = \mathrm{F}\}$. Then in a merged viewpoint, s_1 will be represented by $\{X = \mathrm{TF}, Y = \mathrm{FF}\}$. Moreover, $X = \mathrm{TF}$ implies that the two viewpoints disagree with each other on X. Their framework was intended to highlight the sources of inconsistency or disagreement and to tolerate inconsistencies (e.g. $X = \mathrm{TF}$) between viewpoints during model checking. It did not consider how to resolve these inconsistencies. Barragáns Martínez et al [35] defined a merging operator whose aim was to get the model which best reflects the combined knowledge of all the stakeholders (viewpoints) without first resolving inconsistencies and incompleteness. Although their methodology has envisioned two kinds of possible revision procedures to modify the original viewpoints, useful guidance on how to resolve these inconsistencies by using these revision procedures is not yet provided in [35]. These existing merging frameworks used in managing inconsistent viewpoints focused on tolerating inconsistency rather than resolving inconsistency in merged results.

In contrast, our prioritized merging-based framework is intended to derive appropriate proposals for resolving inconsistencies in requirements specification. Taking the priority of requirements into consideration distinguishes our prioritized merging-based framework from the related works [34] [35]. We argue that an appropriate proposal for resolving inconsistency in requirements should disengage more preferred requirements from inconsistency by making minor concession in abandoning less preferred requirements. Given a globally prioritized requirements collection, we identify the undominated consistent subsets of the requirement collection by a prioritized merging operator (the discrimin operator [27, 28, 25] or the leximin operator [25, 29]), and derive the proposals for resolving inconsistency from these undominated consistent subsets. We also show that these proposals may be viewed as the most appropriate to resolving inconsistency in the sense of the ordering relation over consistent subsets from the global perspective. But we cannot guarantee that these proposals are the most appropriate to each viewpoint involved in inconsistencies. Thus a group decision making mechanism such as vote and negotiation for identifying

an acceptable common proposal from these proposals is also discussed based on the local prioritization of viewpoints.

To construct a globally prioritized requirements specification from a set of locally prioritized requirements collection, we provide two approaches namely the merging-based approach and the priority vector-based approach. The idea of merging-based approach was first discussed and presented in our previous paper [32]. In this framework, each locally prioritized requirements collection is considered as a stratified knowledge base, whilst a *maxsat* dominated construction obtained from corresponding stratified knowledge bases was viewed as the globally prioritized requirements specification. Here in this paper, we extended the idea reported in [32] and considered an extension of the merged result of the corresponding stratified knowledge bases rather than the merged result itself as a globally prioritized requirements specification. Therefore, this method can be easily combined with the prioritized merging in our new framework.

Another particular point about the merging-base approach is that we define the level vector function to capture the relative importance of viewpoints. A closely related work is on weighted knowledge base [33], in which the relative importance of each knowledge base is represented by a non-negative number. Let $w(K)$ be a weight of knowledge base K. Generally, for any K_i and K_j, K_i is regarded as more important than K_j if $w(K_j) < w(K_i)$. Then we may define relative preference relation R_w with regard to weighted knowledge bases as follows:

Definition 8.1. (Relative Preference Relation R_w)
Let $E = \{K_1, \cdots, K_n\}$ be an ordered knowledge profile and $\{\Omega_{K_1,X_1}, \cdots, \Omega_{K_n,X_n}\}$ be a multi-set. A binary relative preference relation $R_s \subseteq \Omega \times \Omega$ is defined as

$$R_s(\omega, \omega') \text{ if and only if } \sum_{\Omega_{K_i,X_i} \ s.t. \ \omega \prec_i \omega'} w(K_i) > \sum_{\Omega_{K_j,X_j} \ s.t. \ \omega' \prec_j \omega} w(K_j),$$

where $\prec_i$ is the strict partial order relation induced from Ω_{K_i,X_i}.

If we want to use the weight-based relative preference relation R_w instead of the *level vector function*-based relative preference relation R_s in requirements engineering, some additional problem will result from this replacement. First of all, how to find appropriate weights assigned to viewpoints is a difficult problem to solve. Although some developers would like to use $\{3, 2, 1\}$ instead of $\{High,\ Medium,\ Low\}$ in requirements development, it can not be viewed as a set of weights assigned to viewpoints. In requirements engineering, each viewpoint with higher priorities prevail over all the viewpoints with lower priorities. In terms of weighted knowledge bases presented in [33], the knowledge base induced by the viewpoint with highest priority should be considered as master knowledge base if there is only one viewpoint with the highest priority. Thus, the weight assigned to the master knowledge base should be greater than the combined weight of the other knowledge bases in the profile [33]. That is, given $E = (T_1, T_2, T_3)$, suppose that $T_1 = \{K_1\}$ and $|T_2| > 1$, then

$$w(K_1) > \sum_{K \in E_i, i \geq 2} w(K)$$

should be satisfied, where $w(K)$ is the weight of K. Obviously, $\{3, 2, 1\}$ is not competent for such cases. Thus, it is necessary to design a set of weights for considering that each viewpoint with higher priorities prevail over all the viewpoints with lower priorities. Moreover, the designed weights may not

be explained intuitively in many cases. For example, if we assign weight 9 to a viewpoint v_A at the *high* level and weight 1 to a viewpoint v_B at the *low* level, we do not think that the degree of importance of v_A is 9 times of that of v_B. Actually, we can also assign any number greater than 9 to v_A.

In contrast, the level vector function can be obtained directly from the original priorities of viewpoints. There is no need for other additional information. Moreover, it embodies that each viewpoint with higher priorities prevails over all the viewpoints with lower priorities by different locations of element 1 in the vectors. Consequently, it may be advisable to use the level vector function rather than weights to capture the relative degree of importance of knowledge bases induced by viewpoints.

The priority vector-based approach is appropriate to the special cases that the viewpoints at the same level adopt the the same scale of local prioritization. Given a requirements statement, both the number of its supporting viewpoints and its local priorities with regard to its supporting viewpoints play an important role in the global priority of the requirements statement. Roughly speaking, the priority vector-based approach is similar to the social vote techniques [31] if we consider each viewpoint as a voter. However, most social voting rules assume that no voter is more important than others. But the priority vector-based approach takes the relative importance of viewpoints into account. It distinguishes the priority vector-based approach from social choice theory such as [31].

With regard to the problem of how to identify an acceptable common proposals for resolving inconsistency, the closely related work is an approach to identifying acceptable common proposals presented in [30], in which the combinatorial vote is adopted as a group decision making mechanism to choose the common proposal. The preference over proposals of each viewpoints was associated with the preference over the goals of the viewpoint rather than the priority of requirements in [30]. In contrast, we argue that both the local priority of requirements and the group decision making mechanism used by viewpoints have impact on the result of identifying an acceptable common proposal. For each viewpoint, its preference over proposals is associated with the local priorities of requirements involved in these proposals. Moreover, we present a sketchy framework for negotiation over derived proposals, which is considered as a part of the prioritized merging-based framework.

9. Conclusions

Developing desirable proposals for handling inconsistency is still a challenging issue in requirements engineering. The relative priority of requirements is considered as a useful indication as how to resolve conflicts and to make trade-off decisions. However, in distributed development of requirements specifications such as the Viewpoints framework, local prioritization with regard to a particular viewpoints is the only available prioritization in many cases. The disagreement in local priorities of shared requirements statements often leads inconsistency handling to a dilemma.

We presented a prioritized merging-based framework for handling inconsistency in the Viewpoints framework in this paper. According to this framework, given a set of inconsistent viewpoints, we constructed a globally prioritized requirements specification from the original requirements collections with the local prioritization firstly. We then mapped this requirements specification to a set of its consistent subsets by using a prioritized merging operator. The prioritized merging operator used in the mapping provides a relation over all the consistent subsets of the requirements specification. Moreover, according to this relation, each consistent subset of the requirements specification in the prioritized merging result is optimal. Following this, we derived some proposals for handling inconsistencies from the prioritized

merging result. These proposals may be considered as the most appropriate ones to handling inconsistency from the global perspective in the sense of ordering relation used in the prioritized merging. But we cannot guarantee that these proposals are the most appropriate to each viewpoint involved in inconsistencies. For an individual viewpoint, different proposals may have different impact on the requirements change with regard to the viewpoint. Therefore, a group decision making mechanism such as negotiation for identifying an acceptable common proposal from these proposals has also been discussed in this paper. Moreover, we argue that the local priorities of requirements play a prominent role in identifying an acceptable common proposal.

We considered two methods for constructing a globally prioritized requirements specification from a set of requirements collections with the local prioritization, including the priority vector-based construction and the merging-based construction. Informally, the priority vector-based construction focuses on how to get the global priority of each requirements statement by integrating its existing local priorities in the form of priority vector function. In contrast, merging-based construction considers each requirements collection with the local prioritization as a stratified knowledge base. The requirements specification with the global prioritization is constructed by merging these stratified knowledge bases. By using the two methods, we also provided two special prioritized merging-based frameworks correspondingly.

If our prioritized merging-based approach is combined with techniques for translating demands in nature language to logical formulas, the integration may provide a basis for automated inconsistency management in requirements engineering. This will be the main direction for our future work.

References

[1] CHAOS: Software Development Report by the Standish Group. Avaiable online at http://www.standishgroup.com/chaos.html (1995).

[2] Ibanez, M.: European user survey analysis. Tech. rep. ESI report TR95104. Euro- pean Software Institute, Zamudio, Spain. http://www.esi.es (1996).

[3] Davis, A.M.: Software Requirements:Objects,Functions, and States. Englewood Cliffs, NJ:PTR Prentice Hall (1993).

[4] Leffingwell, D.: Calculating the return on investment from more effective require- ments management. American Programmer 10 (1997).

[5] Leffingwell, D. Widrig, D.: Managing Software Requirements: A Use Case Approach. Boston, MA: Addison-Wesley (2003)

[6] Finkelsetin, A., J.Kramer, B.Nuseibeh, L.Finkelstein, M.Goedick, “Viewpoints: A Framework for Integrating Multiple Perspectives in System Development”, International Journal of Software Engineering and Knowledge Engineering, 2(1):31-58,1992.

[7] Kotonya, G., I.Sommerville: Viewpoints for requirements definition. IEE Software Eng.Journal 7 (1992) 375-387.

[8] Andrade, J., Ares, J., Garcia, R., Pazos, J., Rodriguez, S., Silva, A.: A methodolog ical framework for viewpoint-oriented conceptual modeling. IEEE Trans. Softw. Eng. 30 (2004) 282-294.

[9] Nuseibeh, B., Kramer, J., Finkelstein, A.: Viewpoints: meaningful relationships are difficult! In: Proceedings of the 25th International Conference on Software Engineering. IEEE CS Press (2003) 676-681.

[10] Gervasi, V., D.Zowghi: Reasoning about inconsistencies in natural language re quirements. ACM Transaction on Software Engineering and Methodologies 14 (2005) 277-330.

[11] Hunter, A., B.Nuseibeh: Managing inconsistent specification. ACM Transactions on Software Engineering and Methodology 7 (1998) 335-367.

[12] Zowghi, D., Gervasi, V.: On the interplay between consistency, completeness, and correctness in requirements evolution. Information and Software Technology 45 (2003) 993-1009.

[13] Nuseibeh, B., Easterbrook, S., Russo, A.: Leveraging inconsistency in software development. IEEE Computer 33 (2000) 24-29.

[14] Nuseibeh, B., S.Easterbrook, A.Russo: Making inconsistency respectable in software development. Journal of Systems and Software 58 (2001) 171-180.

[15] Gabbay, D., Hunter, A.: Making inconsistency respectable 2:meta-level handling of inconsistent data. In: ECSQARU93,LNCS. Volume 747. Springer (1993) 129-136.

[16] Finkelstein, A., Gabbay, D., Hunter, A., Kramer, J., Nuseibeh, B.: Inconsistency handling in multiperspective specifications. IEEE Trans. on Software Engineering 20 (1994) 569-578.

[17] Wiegers, K.: First things first:prioritizing requirements. Software Development 7 (1999) 48-53.

[18] Davis, A.: Just Enough Requirements Management:Where Software Development Meets Marking. Dorset House (2005)

[19] Delgrande, J., Dubois, D., Lang, J.: Iterated revision as priorited merging. In: KR 06. (2006) 210-220.

[20] Yue, A., Liu, W., Hunter, A.: Approaches to constructiing a stratified merged knowledge base. In: ECSQARU2007,LNCS. (2007).

[21] Wiegers, K.E.: Software Requirements,2nd ed. Microsoft Press (2003).

[22] Karlsson, J., Ryan, K.: A cost-value approach for prioritizing requirements. IEEE Software 14 (1997) 67-74.

[23] Pardee, J.: To Satisfy and Delight Your Customer:How to Manage for Customer Value. New York:Dorset House Publishing (1996).

[24] Spanoudakis, G., Finkelstein, A., Till, D.: Overlaps in requirements engineering. Automated Software Engineering 6 (1999) 171-198.

[25] Benferhat, S., Cayrol, C., Dobois, D., Lang, J., Prade, H.: Inconsistency manage- ment and prioritized syntax-based entailment. In: Proceedings of IJCAI93. (1993) 640-647.

[26] Brewka, G.: A rank-based description language for qualitative preferences. In: Proc. of ECAI'04. (2004) 303-307.

[27] Brewka, G.: Preferred subtheories: an extended logical framework for default rea- soning. In: Proc. of IJCAI 1989. (1989) 1043-1048.

[28] Nebel, B.: Belief revision and default reasoning: Syntax-based approaches. In: Proc. of KR91. (1991) 417-428.

[29] Lehmann, D.: Another perspective on default reasoning. Annals of Mathematics and Artificial Intelligence 15 (1995) 61-82.

[30] Mu, K., Jin, Z.: Identifying acceptable common proposals for handling inconsistent software requirements. In: FORTE2007, LNCS4754. (2007) 296-308.

[31] Lang, J.: From logical preference representation to combinatorial vote. In: Proceedings of 8th International Conference on Principles of Knowledge Representation and Reasoning. Morgan Kaufmann (2002) 277-288.

[32] Mu, K., Liu, W., Jin, Z., Lu, R., Yue, A., Bell, D.: A merging-based approach to handling inconsistency in locally prioritized software requirements. In: KSEM2007, LNCS4798. (2007)10-114.

[33] Lin, J.: Integration of weighted knowledge bases. Artificial Intelligence 83 (1996) 363-378.

[34] Easterbrook, S., M.Chechik: A framework for multi-valued reasoning over inconsistent viewpoints. In: Proceedings of International Conference on Software Engineering (ICSE'01), Toronto, Canada (2001) 411-420.

[35] Barragán Martínez, A. B, J. P. Arias, A.F. Vilas, J.G. Duque, M.L. Nores, R.P.D. Redondo, Y.B. Fernández: On the interplay between inconsistency and incompleteness in multi-perspective requirements specification. Information and Software Technology 50(2008) 296-321.

[36] Ryan, L. O.: "Efficient algorithms for clause learning sat solvers",*Masters Thesis, Simon Fraser University*, 2004.

[37] Bird. R., and Hinze, R.:"Functional pearl:Trouble shared is trouble halved", *Proc. of the 2003 ACM SIGPLAN Workshop on Haskell*, pp. 1-6, 2003.

[38] Grover, C., Brew, C., Moens, M., and Manandhar,S.: "Priority union and generalization in dsicourse grammar". *Proc. of the 32nd Annual Meeting of the Association for Computational Linguistics*, pp. 17-24, 1994.

[39] Malouf,R.: Maximal consistent subsets,*Computational Linguistics*, vol.33, no.2,pp.153-160,2007.

[40] Robinson, W.N.: Negotiation behavior during requirements specification. In: ICSE1990, 268-276.

[41] Robinson, W.N., Volkov, V.: Supporting the negotiation life cycle. Communications of the ACM 41 (1998) 95-102

[42] Wooldridge, M., and Parsons, S.: Languages for negotiation. In Proceedings of ECAI2000, 393-397. 2000.

[43] Sierra,C., Jennings, N., Noriega, P., and Parsons, S.: A framework for argumentation-based negotiation. In Intelligent Agents IV (LNAI Vol. 1365)(1998)177-192.