A Merging-Based Approach to Handling Inconsistency in Locally Prioritized Software Requirements

Kedian Mu[1], Weiru Liu[2], Zhi Jin[3], Ruqian Lu[3], Anbu Yue[2], and David Bell[2]

[1] School of Mathematical Sciences
Peking University, Beijing 100871, P.R. China
[2] School of Electronics, Electrical Engineering and Computer Science
Queen's University Belfast, BT7 1NN, UK
[3] Academy of Mathematics and System Sciences
Chinese Academy of Sciences, Beijing 100080, P.R. China

Abstract. It has been widely recognized that the relative priority of requirements can help developers to resolve inconsistencies and make some necessary trade-off decisions. 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 priorities assigned to the same shared requirements statement often puts developers into a dilemma during inconsistency handling process. As a solution to this problem, we present a merging-based approach to handling inconsistency in the Viewpoints framework in this paper. In the Viewpoints framework, each viewpoint is a requirements collection with local prioritization. Informally, we transform such a requirements collection with local prioritization into a stratified knowledge base. Moreover, the relationship between viewpoints is considered as integrity constraints. By merging these stratified knowledge bases, we then construct a merged knowledge base with a global prioritization, which may be viewed as an overall belief in these viewpoints. Finally, proposals for inconsistency handling are derived from the merged result. The global prioritization as well as the local prioritization may be used to argue these proposals and to help developers make a reasonable trade-off decision on handling inconsistency.

Keywords: Inconsistency, Knowledge bases merging, Requirements engineering, Viewpoints, Local prioritization.

1 Introduction

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. The Viewpoints framework [1] has been developed to represent and analyze the

Z. Zhang and J. Siekmann (Eds.): KSEM 2007, LNAI 4798, pp. 103–114, 2007.

different perspectives and their relationships during the requirements stage. A viewpoint is a description of the system-to-be from a particular stakeholder, or a group of stakeholders. It reflects 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 represent their overlaps [2]. These viewpoints may overlap, complement, or contradict each other. Then it makes inconsistency management more necessary during the requirements stage [2,3].

It has been recognized that the relative priority of requirements can help project managers resolve conflicts and make some necessary trade-off decisions [4,5]. However, different stakeholders may assign different levels of priority to the same shared requirements statement in the distributed development such as Viewpoints-based approaches. Actually, for a shared requirements statement, each priority given by a particular stakeholder is a measure of its relative importance only from the perspective of the stakeholder. Moreover, only these local priorities are available in many cases. The disagreement in these local priorities assigned to the same shared requirements statement often puts developers into a dilemma. To make a reasonable trade-off decision on resolving inconsistency, developers need to know global prioritization as well as local prioritization.

As a solution to this problem, we provide a merging-based approach to handling inconsistency in the Viewpoints framework in this paper. Merging is viewed as an usual way to globalization from a set of local information. Informally speaking, we construct a merged requirements specification with global prioritization by merging locally prioritized requirements collections of viewpoints based on the merging operators presented in [6]. The relationship between viewpoints is considered as integrity constraints during the merging process. Then we derive proposals for inconsistency handling from the merged result. Moreover, the global prioritization as well as the local priorities can be used to argue the proposals and help developers make trade-off decisions.

The rest of this paper is organized as follows. Section 2 gives an introduction to the Viewpoints framework and merging operators presented in [6], respectively. Section 3 provides a merging-based approach to handling inconsistency in the Viewpoints framework. Section 4 gives some comparison and discussion about the merging-based approach. Finally, we conclude this paper in Section 5.

2 Preliminaries

2.1 Logical Representation of Viewpoints

Although heterogeneity of representation allows different viewpoints to use different representations to describe their requirements [2], 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 [7]. Moreover, in a logic-based framework for representing requirements, reasoning about requirements is always based on some facts that

describe a certain scenario [7]. It implies that checking the consistency of requirements collections only considers ground formulas[1] 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 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[2]. Let $\alpha \in \mathcal{L}_{\Phi_0}$ be a classical formula and $\Delta \subseteq \mathcal{L}_{\Phi_0}$ a 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.

An usual 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*" [8]. Let m, a natural number, be the scale of the priority and L be $\left\{l_0^m, \cdots, l_{m-1}^m\right\}$, a totally ordered finite set of m symbolic values of the priorities, i.e. $l_i^m < l_j^m$ iff $i < j$. Furthermore, each symbolic value in L could associate with a linguistic value. For example, for a three-level priority set, we have a totally ordered set L as $L = \{l_0^3, l_1^3, l_2^3\}$ where $l_0^3 : Low,\ l_1^3 : Medium,\ l_2^3 : High$. For example, if we assign l_2^3 to a requirements statement α, it means that α is one of the most important requirements statements. Prioritization over Δ is in essence to establish a prioritization function $P : \Delta \rightarrow L$ by balancing the business value of each requirements statement against its cost and technique risk. Actually, for every Δ, prioritization provides a priority-based partition of Δ, $< \Delta^{m-1}, \cdots, \Delta^1, \Delta^0 >$, where $\Delta^k = \{\alpha | \alpha \in \Delta, P(\alpha) = l_k^m\}$, for $k = m-1, \cdots, 0$. We then use $< \Delta^{m-1}, \cdots, \Delta^1, \Delta^0 >$ or (Δ, P) to denote a prioritized requirements collection in this paper. Note that different viewpoints may use different scales of the priority in the Viewpoints framework.

In the Viewpoints framework, let $V = \{v_1, \cdots, v_n\} (n \geq 2)$ be the set of viewpoints and L_i the scale of the priority used by viewpoint v_i, $i \in [1, \cdots, n]$. Then the requirements specification could be represented by a $n+1$ tuple $< (\Delta_1, P_1), \cdots, (\Delta_n, P_n), R >$, where Δ_i and P_i $(1 \leq i \leq n)$ are the set of requirements statements and the prioritization mapping of viewpoint v_i, respectively, and R is a 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 relationship between v_i and v_j could be also explicitly represented by special formulas associated with some formulas in Δ_i and Δ_j. These may be added to the requirements set $\Delta_i \cup \Delta_j$ if necessary. For example, for the relationship of total overlaps defined in [9], we may use a set of formulas in the form of $\phi(a) \leftrightarrow \psi(a)$ to denote the notation ψ and ϕ overlap totally, where

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

[2] We view each ground atomic predicate formula as a propositional atom.

a is a constant. For the sake of simplicity, we use $\Delta_{(v_{i_1},\cdots,v_{i_k})}$ to represent the relationships among viewpoints $v_{i_1},\cdots,v_{i_k}$. Moreover, we assume that we should check the consistency of $\Delta_{(v_{i_1},\cdots,v_{i_k})} \cup (\bigcup_{j=1}^{k} \Delta_{i_j})$ if $\Delta_{(v_{i_1},\cdots,v_{i_k})} \in R$.

It is not surprising that some stakeholders are more important than others. Let L_V be a r-level priority set used in prioritizing viewpoints. Then prioritizing viewpoints is to establish a prioritization mapping $P_V : V \mapsto L_V$. In the rest of this paper, we use (V, P_V) to denote a set of prioritized viewpoints.

A logical contradiction is any situation in which some fact α and its negation $\neg\alpha$ can be simultaneously derived from the same set of formulas Δ. Some works about inconsistency handling in requirements engineering [7,10] refer to *the logical contradiction* as *the inconsistency*. In this paper, we only consider this type of inconsistency in requirements engineering.

Now we give an example to illustrate this representation.

Example 1. Consider the following scenario in eliciting demands about an user interface of a game system. Just for convenience, we assume that the three-level priority set is adopted to prioritize viewpoints as well as requirements of each viewpoint. *Alice* is a delegate of players of an earlier game system. She gives three requirements as follows:

(a1) *The style (sty) of user interface should be more fashionable(FAS) than that of the earlier system.*;
(a2) *The user interface should provide flexible (FLE) choice of settings(set) to players*;
(a3) *The elements(ele) of user interface should be familiar(FAM) to all the players.*

Then she assigns the level of *high* and the level of *medium* to (a1-2) and (a3), respectively.

Bob is a delegate of potential players of the system-to-be. He gives three demands as follows:

(b1) *The style of user interface should be very fashionable*;
(b2) *The user interface should provide flexible choice of settings to players*;
(b3) *The elements of user interface should be unexpected (UNE) to all players.*

He assigns the level of *high* to (b1) and (b2). (b3) is viewed as a requirements statement with the level of *medium*.

John is a consultant in the user interface of game systems. He gives the same demands as Alice. The main difference between *Alice* and *John* is that he assigns the level of *low* to the third. In addition, *Bob* is one of the most important stakeholders. *Alice* and *John* are two important stakeholders. Their priorities are *High*, *Medium*, and *Medium*, respectively. Obviously, the three stakeholders should reach agreement on the user interface.

Let v_A be the viewpoint of *Alice*, then $P_V(v_A) = l_1^3$ and

$$\Delta_A = \{\text{FAS(sty)}, \text{FLE(set)}, \text{FAM(ele)}\}.$$
$$P_A(\text{FAS(sty)}) = l_2^3,\ P_A(\text{FLE(set)}) = l_2^3,\ P_A(\text{FAM(ele)}) = l_1^3.$$
$$(\Delta_A, P_A) =< \{\text{FAS(sty)},\text{FLE(set)}\}, \{\text{FAM(ele)}\}, \emptyset > .$$

Let v_B be the viewpoint of *Bob*, then $P_V(v_B) = l_2^3$ and

$$\begin{aligned}
&\Delta_B = \{\text{FAS(sty)},\ \text{FLE(set)}, \text{UNE(ele)}\}.\\
&P_B(\text{FAS(sty)}) = l_2^3,\ P_B(\text{FLE(set)}) = l_2^3,\ P_B(\text{UNE(ele)}) = l_1^3.\\
&(\Delta_B, P_B) =< \{\text{FAS(sty)},\ \text{FLE(set)}\}, \{\text{UNE(ele)}\}, \emptyset > .
\end{aligned}$$

Let v_J be the viewpoint of *John*, then $P_V(v_J) = l_1^3$ and

$$\begin{aligned}
&\Delta_J = \{\text{FAS(sty)},\ \text{FLE(set)}, \text{FAM(ele)}\}.\\
&P_J(\text{FAS(sty)}) = l_2^3,\ P_J(\text{FLE(set)}) = l_2^3,\ P_J(\text{FAM(ele)}) = l_0^3.\\
&(\Delta_J, P_J) =< \{\text{FAS(sty)},\ \text{FLE(set)}\}, \emptyset, \{\text{FAM(ele)}\} > .
\end{aligned}$$

Obviously, $R = \{\Delta_{(v_A,v_B,v_J)}\}$ and $\Delta_{(v_A,v_B,v_J)} = \{\text{FAM(ele)} \leftrightarrow \neg\text{UNE(ele)}\}$. Then the partial requirements specification comprising viewpoints $\{v_A, v_B, v_J\}$ is $< (\Delta_A, P_A), (\Delta_B, P_B), (\Delta_J, P_J), R >$. Moreover, we can conclude that

$$\Delta_A \cup \Delta_B \cup \Delta_J \cup \Delta_{(v_A,v_B,v_J)} \vdash \text{UNE(ele)} \wedge \neg\text{UNE(ele)}.$$

We will come back to this example in section 3.

2.2 Knowledge Bases Merging

Merging is an usual approach to fusing a set of heterogeneous information. The gist of knowledge base merging is to derive an overall belief set from a collection of 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 exists at least one 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 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 $\bigcup_{j=i}^{n} S_j$ w.r.t $\preceq$. Each S_i is called a stratum of K and is non-empty. We denote $\bigcup K = \bigcup_{j=1}^{n} S_j$. A prioritized knowledge profile E is a multi-set of stratified 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 lossing 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 retain all the original knowledge and may also introduce additional formulas. Most merging operators just generate a flat base as the result. At present, only the merging operators presented in [6] can be used to construct a stratified merged knowledge base. In this paper, we adopt the syntax-based operators presented in [6] to merge inconsistent requirements collections.

Given a stratified knowledge base K, 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 the *best out* ordering [11], the *maxsat* ordering [12] and the *leximin* ordering [11]. In this paper, we use the *maxsat* ordering, though it is not obligatory.

Definition 1 (Maxsat Ordering [12]). *Given* $K=(S_1, \cdots, S_n)$. *Let* $r_{MO}(\omega)=\min\{i : \omega \models S_i\}$, *for* $\omega \in \Omega$. *By convention,* $\min\{\emptyset\} = +\infty$. *Then the maxsat ordering* $\preceq_{maxsat}$ *on* Ω *is defined as:* $\omega \preceq_{maxsat} \omega'$ *iff* $r_{MO}(\omega) \leq r_{MO}(\omega')$.

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

Yue et al. [6] 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 (Relative Preference Relation [6]). *Let* $\{\Omega_{K_1,X_1}, \cdots, \Omega_{K_n,X_n}\}$ *be a multi-set. A binary relative preference relation* $R \subseteq \Omega \times \Omega$ *is defined as:* $R(\omega, \omega')$ *iff* $|\{\Omega_{K_i,X_i} \text{ s.t. } \omega \prec_i \omega'\}| > |\{\Omega_{K_i,X_i} \text{ s.t. } \omega' \prec_i \omega\}|$, *where* $\prec_i$ *is the strict partial order relation induced from* Ω_{K_i,X_i}.

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

Definition 3 (Undominated Set [6]). *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* $\forall \omega \in Q$, $\forall \omega' \in \Omega \setminus Q$, $R(\omega', \omega)$ *does not hold.* Q *is a minimal undominated set of* Ω *if for any undominated set* P *of* Ω, $P \subset Q$ *does not hold.*

We denote the set of minimal undominated sets of Ω w.r.t R as U^R_Ω. Then we can stratify the interpretations as follows:

Definition 4 (Stratification of Ω Obtained from R [6]). *Let* R *be a relative preference relation. A stratification of interpretations* $\Omega = (\Omega_1, \cdots, \Omega_n)$ *can be obtained from* R *such that* $\Omega_i = \cup Q$, *where* $Q \in U^R_{\Omega - \cup_{j=1}^{i-1} \Omega_j}$.

Definition 5 (Maxsat-Dominated Construction [6]). *Let* $\Omega=(\Omega_1, \cdots, \Omega_n)$ *be a stratification of interpretation and* S *be a set of propositions. A stratified knowledge base* $K_S^{maxsat,\Omega} = (S_1, \cdots, S_m)$ *is a maxsat-dominated construction from* S *w.r.t* Ω *if* $\bigcup_{i=1}^{m} S_i \subseteq S$ *and* $\Omega_{K_S^{maxsat,\Omega},maxsat} = \Omega$.

Yue et al. [6] has also shown how to construct a maxsat-dominated construction as a stratified merged result from the original bases based on the stratification of Ω obtained from R.

Proposition 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.*

$\Omega_{K,maxsat} = \Omega$ *and* $\bigcup K \subseteq S$*, then* $K_S^{maxsat,\Omega} = (S_1, \cdots, S_n)$ *is a maxsat-dominated construction from* S *w.r.t* Ω*, where* $S_i = \{\varphi \in S | \forall \omega \in \Omega_i, \omega \models \varphi\} - \bigcup_{j=1}^{i-1} S_j$ *and* $S_i \neq \emptyset$.

Actually, if there is an integrity constraint μ during the merging process, then we only need to use Ω^μ instead of Ω in the definitions above, where Ω^μ is the set of all the models of μ.

3 A Merging-Based Approach to Handling Inconsistent Requirements with Local Prioritization

We start this section with consideration of *Example* 1. Intuitively, developers should persuade someone to abandon some requirements statements so as to retain more important requirements from a global perspective. Consider the local priorities of the two requirements involved in the inconsistency:

$$P_A(\text{FAM(ele)}) = l_1^3,\ P_J(\text{FAM(ele)}) = l_0^3,\ P_B(\text{UNE(ele)}) = l_1^3.$$

As a shared requirements statement, $FAM(ele)$ has two different priorities given by *Alice* and *John*, respectively. To determine whether $UNE(ele)$ is more important than $FAM(ele)$ from a global perspective, it is necessary to derive a merged requirements collection with global prioritization based on the requirements collections with local prioritization.

3.1 Merging an Ordered Knowledge Profile

Merging provides a promising way to extract an overall view from distributed viewpoints. Intuitively, 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 some viewpoints are more important than others.

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 $\leq_E$. Generally, the pre-order relation on E should be considered during the merging process. Actually, as mentioned in [6], only the *relative preference relation* over interpretations is meaningful in the merging process. Consequently, we will integrate the pre-order relationship over a profile into the relative preference relation defined in *Definition 1*.

Definition 6 (Level Vector Function). *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* $j \in [1, m],\ j \neq i$.

Given a total ordering relation $\leq_s$ on N^m as follows: $\forall(a_1, \cdots, a_m), (b_1, \cdots, b_m) \in \{0,1\}^m$, $(a_1, \cdots, a_m) \leq_s (b_1, \cdots, b_m)$ iff $a_i = b_i$ for all i, or $\exists i$ s.t $a_i > b_i$ and $a_j = b_j$ for all $j < i$. Further, $(a_1, \cdots, a_m) <_s (b_1, \cdots, b_m)$ iff $(a_1, \cdots, a_m) \leq_s (b_1, \cdots, b_m)$ and $(b_1, \cdots, b_m) \not\leq_s (a_1, \cdots, a_m)$. Obviously,$K_i \leq_E K_j$ iff $s(K_i) \leq_s s(K_j)$. It means $s(K)$ gives a numerical measure of the relative importance of K w.r.t $\leq_E$.

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

Definition 7 (Relative Preference Relation). *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*

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

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

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

Example 2. Consider an ordered knowledge profile $E = (K_1, K_2)$, where $K_1 = (\{p\}, \{\neg p\})$ and $K_2 = (\{\neg p\}, \{p\})$. The set of interpretations is $\Omega = \{\omega_1 = 1, \omega_2 = 0\}$. Then $r_{MO,K_1}(\omega_1) = 1$, $r_{MO,K_1}(\omega_2) = 2$; $r_{MO,K_2}(\omega_1) = 2$, $r_{MO,K_2}(\omega_2) = 1$. So, $\omega_1 \prec_{K_1,maxsat} \omega_2$ and $\omega_2 \prec_{K_2,maxsat} \omega_1$. If we do not consider $\leq_E$, 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. In contrast, if we consider $\leq_E$ on E, then $s(K_1) = (1, 0)$ and $s(K_2) = (0, 1)$. So, $R_s(\omega_1, \omega_2)$ holds. The stratification of interpretations is $\Omega = (\{\omega_1\}, \{\omega_2\})$. We get a maxsat-dominated construction $K = (\{p\}, \{\neg p\})$. It is an intuitive result of merging.

3.2 Handling Inconsistent Requirements Collections with Local Prioritization

The gist of this paper is to provide a merging-based approach to handling inconsistent viewpoints, as shown in figure 1. Informally speaking, we first transform each requirements collection with a local prioritization involved in inconsistencies 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 based on the merging operators presented in [6]. The merged result can be considered as a overall view of these viewpoints. Moreover, the ordering relation over the merged knowledge base could be viewed as a global prioritization on the merged requirements collection. Finally, we derive proposal candidates for handling inconsistency from the stratified merged knowledge base. The global prioritization as well as the local prioritization may be used to argue these proposals and help developers make some trade-off decisions. If a proposal is acceptable to all the viewpoints involved in inconsistencies, the viewpoints will be modified according to the proposal.

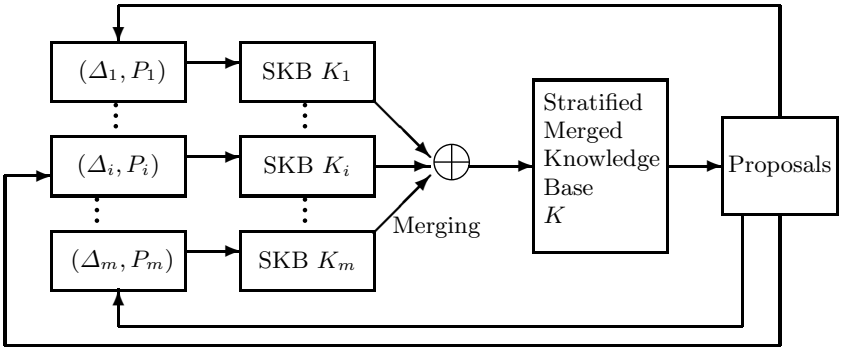

Fig. 1. A Merging-base Approach to Handling Inconsistency

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

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

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

- 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 $\Delta_i^{m-1}, \cdots, \Delta_i^0$.

Constructing A Stratified Merged Knowledge Base: Suppose that $v_{i_1}, \cdots$, v_{i_k} are the viewpoints involved in inconsistency. Let $E = \{K_{i_1}, \cdots, K_{i_k}\}$ be a knowledge profile, where K_{i_l} is the stratified knowledge base induced by (Δ_{i_l}, P_{i_l}) for all $1 \leq i_l \leq i_k$. Let Ω be the set of interpretations. Then we

- define an ordering relation $\leq_E$ on E s.t. $K_{i_l} \leq_E K_{i_j}$ iff $P_V(v_{i_l}) \geq P_V(v_{i_j})$;
- compute the level vector function s based on stratification of E w.r.t $\leq_E$.
- consider $\Delta_{(v_{i_1}, \cdots, v_{i_k})}$ as an integrity constraint μ and compute $\Omega^\mu = \{\omega \in \Omega, \omega \models \mu\}$;
- find $\Omega^u_{K_{i_l}, maxsat}$ for all i_l.
- based on $\{\Omega^u_{K_{i_1}, maxsat}, \cdots, \Omega^u_{K_{i_k}, maxsat}\}$, construct stratification of interpretations $\Omega^u = (\Omega^u_1, \cdots, \Omega^u_m)$ by using relative preference relation R_s over Ω^u.
- get a maxsat-dominated construction K based on *Proposition* 1.

Deriving A Proposal Candidate To Handling Inconsistency: The preference relation on the maxsat-dominated construction K describes the relative importance of requirements from a global perspective. Then it naturally derives proposals that the requirements with lower global priorities should be abandoned so as to resolve the inconsistencies. However, these proposals are just recommendations. Stakeholders will make further trade-off decisions based on the global prioritization as well as the local prioritization.

We give an example to illustrate how to apply the merging-based approach to handling inconsistent requirements collections with a local prioritization.

Example 3. Consider *Example* 1 again. We may get the following stratified knowledge bases induced by v_A, v_B, and v_J respectively:

$$K_A = (\{\text{FAS(sty)}, \text{FLE(set)}\}, \{\text{FAM(ele)}\});$$
$$K_B = (\{\text{FAS(sty)}, \text{FLE(set)}\}, \{\text{UNE(ele)}\});$$
$$K_J = (\{\text{FAS(sty)}, \text{FLE(set)}\}, \{\text{FAM(ele)}\}).$$

Then $E = (\{K_B\}, \{K_A, K_J\})$ and $s(K_B) = (1, 0)$, $s(K_A) = s(K_J) = (0, 1)$. The integrity constraint is $\mu = \{\text{FAM(ele)} \leftrightarrow \neg\text{UNE(ele)}\}$. We denote each model by a bit vector consists of truth values of (FAM(ele), UNE(ele), FAS(sty), FLE(set)). Then $\Omega^\mu = \{\omega_1 = 0100, \omega_2 = 0101, \omega_3 = 0110, \omega_4 = 0111, \omega_5 = 1000, \omega_6 = 1001, \omega_7 = 1010, \omega_8 = 1011\}$. r_{MO} is given in table 1.

Table 1. Ranks of interpretations given by the maxsat ordering strategy

ω	K_A	K_B	K_J
0100	$+\infty$	2	$+\infty$
0101	$+\infty$	2	$+\infty$
0110	$+\infty$	2	$+\infty$
0111	1	1	1
1000	2	$+\infty$	2
1001	2	$+\infty$	2
1010	2	$+\infty$	2
1011	1	1	1

Then we can get

$$\Omega^\mu_{K_A, maxsat} = (\{\omega_8, \omega_4\}, \{\omega_5, \omega_6, \omega_7\}, \{\omega_1, \omega_2, \omega_3\});$$
$$\Omega^\mu_{K_B, maxsat} = (\{\omega_8, \omega_4\}, \{\omega_1, \omega_2, \omega_3\}, \{\omega_5, \omega_6, \omega_7\});$$
$$\Omega^\mu_{K_J, maxsat} = (\{\omega_8, \omega_4\}, \{\omega_5, \omega_6, \omega_7\}, \{\omega_1, \omega_2, \omega_3\}).$$

Furthermore, we get a stratification of Ω^μ based on the relative preference relation R_s on Ω^μ as $\Omega^\mu = (\{\omega_8, \omega_4\}, \{\omega_1, \omega_2, \omega_3\}, \{\omega_5, \omega_6, \omega_7\})$. According to Proposition 1, we get a maxsat-dominated stratified construction

$$K = (\{\text{FAS(sty)}, \text{FLE(set)}\}, \{\text{UNE(ele)}\}, \{\text{FAM(ele)}\}).$$

This merged result implies that FAM(ele) is *less important than* UNE(ele) from a global perspective.

The proposal for handling inconsistency derived from this result of merging, denoted π, is that *the developer had better persuade Alice and John to abandon their shared demand about elements of user interface.* If the proposal π is acceptable to the three stakeholders, then

$$\Delta^\pi_A = \Delta_A - \{\text{FAM(ele)}\},\ \Delta^\pi_B = \Delta_B,\ \Delta^\pi_J = \Delta_J - \{\text{FAM(ele)}\},$$

where Δ_i^π is the modification of Δ_i by performing π. The inconsistency disappears in $\Delta_A^\pi \cup \Delta_B^\pi \cup \Delta_J^\pi \cup \Delta_{(v_A, v_B, v_J)}$.

4 Discussion and Comparison

The disagreement in the local prioritization over shared requirements often leads inconsistency handling to a dilemma. It may be viewed as a promising way to identify appropriate proposals for handling inconsistency from a global perspective. But this does not mean that the global prioritization is more crucial than the local prioritization. We argue that both the global prioritization and the local prioritization play important roles in resolving inconsistencies. For example, the proposal π in *Example* 3 is considered appropriate to handling the inconsistency *from a global perspective*. Obviously, *John* maybe accept the proposal, since for the demand to be abandoned, $P_J\{\text{FAM(ele)}\} = Low$. But we can't assure that *Alice* agrees to abandon the shared demand since $P_A\{\text{FAM(ele)}\} = Medium$. In summary, the local prioritization has an impact on the acceptance of merged result to viewpoints. How to identify appropriate common proposals for inconsistency handling based on the local prioritization as well as the merged preference is still one of issues in our future work.

On the other hand, we adopt the syntax-based merging operators presented in [6] 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 we can not get a stratified merged 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.

5 Conclusions

Identifying appropriate actions or proposals for handling inconsistency is still a big problem in requirements engineering. The relative priority of requirements is considered as a useful clue to resolving conflicts and making trade-off decisions. However, in distributed development of requirements specifications such as the Viewpoints framework, the disagreement in local priorities of shared requirements statements often leads inconsistency handling to a dilemma.

The main contribution of this paper is to provide a merging-based approach to handling inconsistency in locally prioritized software requirements. Given an inconsistency, each viewpoint involved in the inconsistency is transformed into a stratified knowledge base, whilst the relationship between these viewpoints is considered as an integrity constraint. Based on the merging operators presented in [6], we construct a stratified merged knowledge base as an overall view of these inconsistent viewpoints. The ordering relationship over this stratified merged knowledge base could be considered as a global prioritization over the requirements specification. Generally, the requirements with lower merged preference may be considered as requirements to be abandoned. Then we may

derive some proposals for handling the inconsistency from the merged result. The global prioritization as well as the local prioritization may be used to argue these proposals and help developers identifying acceptable common proposals.

Acknowledgements

This work was partly supported by the National Natural Science Fund for Distinguished Young Scholars of China under Grant No. 60625204, the Key Project of National Natural Science Foundation of China under Grant No. 60496324, the National Key Research and Development Program of China under Grant No. 2002CB312004, the National 863 High-tech Project of China under Grant No. 2006AA01Z155, the Knowledge Innovation Program of the Chinese Academy of Sciences, and the NSFC and the British Royal Society China-UK Joint Project.

References

1. Finkelsetin, A., Kramer, J., Nuseibeh, B., Finkelstein, L., Goedicke, M.: Viewpoints: A framework for integrating multiple perspectives in system development. International Journal of Software Engineering and Knowledge Engineering 2, 31–58 (1992)
2. Nuseibeh, B., Kramer, J., Finkelstein, A.: Viewpoints: meaningful relationships are difficult? In: Proceedings of the 25th International Conference on Software Engineering, pp. 676–681. IEEE CS Press, Los Alamitos (2003)
3. Finkelstein, A., Gabbay, D., Hunter, A., Kramer, J., Nuseibeh, B.: Inconsistency handling in multiperspective specifications. IEEE Transactions on Software Engineering 20, 569–578 (1994)
4. Wiegers, K.: First things first:prioritizing requirements. Software Development 7, 48–53 (1999)
5. Davis, A.: Just Enough Requirements Management: Where Software Development Meets Marking. Dorset House, New York (2005)
6. Yue, A., Liu, W., Hunter, A.: Approaches to constructiing a stratified merged knowledge base. In: ECSQARU 2007. LNCS, Springer, Heidelberg (2007)
7. Hunter, A., Nuseibeh, B.: Managing inconsistent specification. ACM Transactions on Software Engineering and Methodology 7, 335–367 (1998)
8. Wiegers, K.: Software Requirements, 2nd edn. Microsoft Press, Redmond (2003)
9. Spanoudakis, G., Finkelstein, A., Till, D.: Overlaps in requirements engineering. Automated Software Engineering 6, 171–198 (1999)
10. Gervasi, V., Zowghi, D.: Reasoning about inconsistencies in natural language requirements. ACM Transactions on Software Engineering and Methodologies 14, 277–330 (2005)
11. Benferhat, S., Cayrol, C., Dobois, D., Lang, J., Prade, H.: Inconsistency management and prioritized syntax-based entailment. In: Proceedings of IJCAI 1993, pp. 640–647. Morgan Kaufmann, San Francisco (1993)
12. Brewka, G.: A rank-based description language for qualitative preferences. In: Proc. of ECAI 2004, pp. 303–307. IOS Press, Amsterdam (2004)