Mu KD, Liu W, Jin Z *et al.* Managing software requirements changes based on negotiation-style revision. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 26(5): 890–907 Sept. 2011. DOI 10.1007/s11390-011-0187-y

Managing Software Requirements Changes Based on Negotiation-Style Revision

Ke-Dian Mu[1] (牟克典), *Member*, *CCF*, Weiru Liu[2], Zhi Jin[3] (金　芝), *Senior Member*, *CCF*, *IEEE*, Jun Hong[2] and David Bell[2]

[1] *School of Mathematical Sciences, Peking University, Beijing 100871, China*

[2] *School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, BT7 1NN, U.K.*

[3] *Key Laboratory of High Confidence Software Technologies (Peking University), Ministry of Education*
School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China

E-mail: mukedian@math.pku.edu.cn; fw.liu@qub.ac.uk; zhijin@sei.pku.edu.cn; {j.hong, da.bellg}@qub.ac.uk

Received September 20, 2010; revised June 30, 2011.

Abstract For any proposed software project, when the software requirements specification has been established, requirements changes may result in not only a modification of the requirements specification but also a series of modifications of all existing artifacts during the development. Then it is necessary to provide effective and flexible requirements changes management. In this paper, we present an approach to managing requirements changes based on Booth's negotiation-style framework for belief revision. Informally, we consider the current requirements specification as a belief set about the system-to-be. The request of requirements change is viewed as new information about the same system-to-be. Then the process of executing the requirements change is a process of revising beliefs about the system-to-be. We design a family of belief negotiation models appropriate for different processes of requirements revision, including the setting of the request of requirements change being fully accepted, the setting of the current requirements specification being fully preserved, and that of the current specification and the request of requirements change reaching a compromise. In particular, the prioritization of requirements plays an important role in reaching an agreement in each belief negotiation model designed in this paper.

Keywords requirements change, non-prioritized belief revision, inconsistency handling, negotiation

1 Introduction

For any proposed software development project, it seems to be inevitable to confront requirements changes during the software development life cycle. Stakeholders change their minds for many reasons, including policies and legislation changes, commercial strategies updating and marketplace changes, identifying a defect in proposed requirements or missing a requirement, and realizing that they misunderstood their actual demand. Generally, the earlier the requirements are frozen (i.e., the software requirements specification has been established), the more requirements changes would occur later. Suitable requirements changes may boost satisfactions of some stakeholders to the system-to-be and enhance the quality of software requirements specification and subsequent artifacts.

But uncontrolled requirements changes must result in many troublesome problems during the development, especially when the software requirements specification has been established[1]. Uncontrolled consumption of development resources such as time, funds, and human resource for accommodating requirements changes may result in delay of delivery of the software product, overburden on developers, and difficulties in funds. Moreover, uncontrolled requirements change may result in inconsistency and other indetectable defects in the system-to-be, which will degrade the quality of software product. On the other hand, to develop software product which is both high-quality and high-value, it is unadvisable to prohibit the requirements change during the development process, even if the software requirements specification has been established. Consequently, it is necessary to provide effective and flexible approaches to managing requirements changes.

Logic-based techniques for managing requirements change have drawn significant attention recently. Garcez *et al.*[2-3] argued that the evolution of

Regular Paper

This work was partly supported by the National Natural Science Foundation of China under Grant No. 60703061, the National Basic Research 973 Program of China under Grant No. 2009CB320701, the Key Project of National Natural Science Foundation of China under Grant No. 90818026, and the NSFC & the British Royal Society China-UK Joint Project.

requirements specifications can be supported by a cycle composed of two phases: analysis and revision. The analysis phase focuses on checking whether a number of desirable properties of a system-to-be are satisfied by its partial requirements specification based on abductive reasoning[4]. Some diagnostic information are also provided when a certain property is violated by the specification. Then inductive learning-based[5] revision phase executes the change from the given specification into a new specification by making use of the diagnostic information provided by the analysis phase. Zowghi *et al.*[6-7] proposed belief revision for default theories as a formal approach for resolving inconsistencies caused by evolutionary changes of requirements. They argued that the requirements specification can be formulated as default theories where each requirement may be firm, defeasible, or discarded. Inconsistencies introduced by an evolutionary change are resolved by performing a revision operation over the entire specification. These approaches are appropriate to providing support to managing requirements changes during the requirements specification development, in which many requirements are considered as imprecise, incomplete and immature.

However, when the requirements specification has been established, most requirements are considered as reliable and relative complete. Compared to the concerns of [2-3, 6-7], developers do not focus on finding possible changes to evolve requirements at this stage. The request of requirements change is always triggered by some very particular or uncertain factors. Because the current requirements specification is viewed as a baseline for subsequent stages including design, coding and testing, any modification of the current requirements specification may cause a series of changes at each developed or developing stage in the software development life cycle. Consequently, the request of requirements change should be handled cautiously, moreover, an acceptable requirements change to this stage should not lead to a major modification to the current requirements specification in general case.

The change control board (CCB for short) is responsible for considering the necessity of each change request[1]. In particular, the CCB should also estimate the impact that the requirements change has on the current requirements specification. In other words, the CCB should consider the possible result caused by the requirements change. The process of change is not always an easy process of adding the new requirements to the current requirements specification and giving up the corresponding older requirements. Actually, the requirements changes often result in conflicts between the existing requirements and the new requirements. Thus, the process of requirements change is associated with resolving conflicts in many cases. For the CCB, the possible accommodation caused by the requirements change should be taken into account in decision making about the request of requirements change. There are three possible ways to implement a given request of requirements change as follows.

(a) The request of change is fully accepted. If the request contradicts the current requirements specification, then stakeholders should accommodate the current requirements specification to the request. That is, the current requirements specification needs to give up some existing requirements. Generally, this kind of request is very necessary to software project, such as the requests caused by policy and legislation change.

(b) If the request contradicts the current requirements specification, the CCB accommodate the request to the current requirements specification. That is, the request is partially accepted to augment the current requirements specification.

(c) If the request contradicts the current requirements specification, the current requirements specification and the new requirements accommodate themselves to each other. In other words, both the request and the current requirements specification need to make concessions.

Belief revision provides a promising way to manage the requirements changes. Informally speaking, belief revision is the process of changing the beliefs of an agent in some world when new evidence (possibly inconsistent with the existing beliefs) about that world is given. The current software requirements specification may be viewed as a set of beliefs of stakeholders in the system-to-be. If we consider each request of requirements change as an evidence about the system-to-be, then the process of sequential changes can be viewed as a process of iterated belief revision, moreover, the revised belief set can be viewed as the revised requirements specification by executing the change.

A belief revision operator is always characterized by a number of rationality criteria (called postulates). The AGM framework[8] and its most adaptations such as DP framework[9] assume that the new information is more reliable, then the new information should always be fully accepted in the revision result. This is referred to as the success postulate. In contrast, some so-called *non-prioritized* belief revisions[10-13] do not think the new information should be always fully accepted after revision. Evidently, non-prioritized belief revision is more appropriate to managing requirements change.

In this paper, we present an approach to managing the requirements changes based on the negotiation-style

belief revision. In particular, the priority level of requirements plays an important role in negotiation models. The negotiation-style revision presented by R. Booth[12] is a kind of non-prioritized belief revision, in which the result of revision is arrived at via a kind of negotiation between the existing information and new information. The belief negotiation model in [12] provides a more flexible framework to belief revision. We consider the current requirements specification and the request of requirements change as two negotiation parties in belief revision. Then for different requests of requirements change, we design different belief negotiation models to execute the requirements changes. Informally, we provide three belief negotiation models for the following settings, i.e., the request is fully accepted, the current requirements specifications is fully preserved, and the current requirements specification and the requested changes accommodate themselves to each other. The three possible results of revision may help developers to make reasonable trade-off decisions about the request of requirements change.

The rest of this paper is organized as follows. In Section 2 we introduce the logical representation of requirements specification and also give a brief overview to negotiation-style framework for belief revision presented by Booth[12]. Then in Section 3 we provide the negotiation-style revision-based approach to managing requirements changes. Section 4 gives some comparison between our approach and other related works. Finally, we conclude this paper in Section 5.

2 Preliminaries

2.1 Logical Representation of Requirements Specification

We consider the use of classical logic-based language in representation of requirements specifications in this paper. Although different software projects may use different notations and tools to represent their requirements during the requirements stage, 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[14]. That is, first order logic may be considered as a promising tool to represent requirements. Moreover, in a logic based framework for representing requirements, reasoning about requirements is always based on some facts that describe a certain scenario[14]. It implies that checking the consistency of a set of requirements only considers ground formulas① 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 $\alpha \in \mathcal{L}_{\Phi_0}$ 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 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 Software Engineering and the book is available, then Alice can borrow the book*" in a certain scenario, we can represent the requirement by

$$\texttt{Require}(\texttt{Alice}, \texttt{S\_E}) \wedge \texttt{Available}(\texttt{S\_E}) \rightarrow \texttt{Borrow}(\texttt{Alice}, \texttt{S\_E}).$$

Let $\mathcal{S}$ be the current *requirements specification*. It contains all the existing requirements statements.

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 perspectives. A common approach to prioritizing requirements collection is to group requirements statements into several priority categories, such as the most frequent three-level scale of "*High*", "*Medium*", "*Low*"[1,15] and the five-level scale of priorities used in [16].

Another technique for prioritizing requirements specifications is based on numerical estimations of value, cost and risk of each requirements statements, such as the cost-value approach[17] and the quality function deployment (QFD for short)[18]. However, K. Wiegers has pointed that few software organizations are willing to undertake the rigor of QFD in his experience[15].

In this paper, we use a common prioritization scale to group requirements into several priority categories. Let m, a natural number, be the scale of the priority level and L^m be $\{l_1, \ldots, l_m\}$, 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 higher value in L^m signifies a lower priority. Furthermore, each symbolic value in L^m could associate with a linguistic

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.

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 the paper, we adopt this three-level priority set in most examples, though it is not obligatory. From a 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, $\langle \Delta^1, \Delta^2, \ldots, \Delta^m \rangle$, where $\Delta^k = \{\alpha | \alpha \in \Delta, P(\alpha) = l_k\}$, for $k = 1, \ldots, m$. We then use $\langle \Delta^1, \Delta^2, \ldots, \Delta^m \rangle$ to denote a prioritized requirements collection in this paper.

The term of *inconsistency* in requirements collections has different definitions in requirements engineering[19]. Most logic-based works such as [14, 19-20] 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 be also concerned with the logical contradiction. Let $Cn(\Delta) = \{\alpha | \Delta \vdash \alpha\}$. It is the set of all the consequences derivable from Δ. If there is a formula α such that $\alpha \in Cn(\Delta)$ and $\neg\alpha \in Cn(\Delta)$, then we consider Δ to be *inconsistent* and abbreviate $\alpha \wedge \neg\alpha$ by $\bot$.

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

2.2 Booth's Negotiation-Style Framework for Belief Revision

In this subsection, we give an overview of the negotiation-style framework for belief revision presented by Booth[12].

The negotiation-style framework for belief revision imagines the inconsistency between the current belief set K and new information ϕ as a gap between K on one side and ϕ on the other. Then the process of revision is thought to be the process of bridging this gap[12].

We adopt the following notations, abstractive functions used in [12]. We use $\mathcal{W}$ and $\mathcal{B}$ to denote the set of all possible propositional worlds and the set of all non-empty subsets of $\mathcal{W}$, respectively. For any set of worlds $\mathcal{U}$, we use $Th(\mathcal{U})$ to denote the set of sentences true in every world in $\mathcal{U}$. Moreover, we interpret $\mathcal{U}$ as information that the actual "true" world is one of the worlds in $\mathcal{U}$ in the following negotiation models.

For any set of formulas I, $[I]$ denotes the set of worlds in which every sentence in I is true, i.e., $[I]$ is the set of models for sentences in I. If $[I] \neq \varnothing$ then I is consistent. Especially, for $\phi \in \mathcal{L}_{\Phi_0}$ we write $[\phi]$ rather than $[\{\phi\}]$. For any world w and any formula α, if α is true in the world w, then we write $w \models \alpha$. Obviously, for each $\alpha \in I$ and $w \in [I]$, $w \models \alpha$.

A belief set K is a consistent and deductively closed set of sentences, i.e., $[K] \neq \varnothing$ and $K = Cn(K)$. We use $\mathcal{L}^*_{\Phi_0}$ and $\mathcal{K}$ to denote the set of all consistent sentences and the set of all belief sets, respectively.

Let s and t be two sources of information which provide information S and T respectively, where S and T are two non-empty subsets of $\mathcal{W}$. The intuitive idea of negotiation-style merging S and T into a single piece $Merge(S, T)$ is to incrementally enlarge S or T or both until their intersection is non-empty[12]. Consider $\langle S_0, T_0 \rangle$ where $S_0 = S, T_0 = T$, if $S_0 \cap T_0 \neq \varnothing$, then we just take $Merge(S, T) = S_0 \cap T_0$. But if $S_0 \cap T_0 = \varnothing$, we start the first round of negotiation. After some concessions are made by s or t or both, we arrive at the pair $\langle S_1, T_1 \rangle$, where $S_0 \subseteq S_1, T_0 \subseteq T_1$. If $S_1 \cap T_1 \neq \varnothing$, then we just take $Merge(S, T) = S_1 \cap T_1$. Otherwise we repeat this and do the next round of negotiation.

A possible stage in the negotiation process starting with S_0 and T_0 can be represented by $\sigma = (\langle S_0, T_0 \rangle, \ldots, \langle S_n, T_n \rangle)$, an *increasing* sequence of pairs of elements of $\mathcal{B}$, where $S_i \subseteq S_{i+1}, T_i \subseteq T_{i+1}$ for all $i = 0, 1, \ldots, n-1$. Let Ω denote the set of all finite sequences of pairs of elements of $\mathcal{B}$. Then we define the set of sequences $\Sigma \subseteq \Omega$ by

$$\Sigma = \{\sigma = (\langle S_0, T_0 \rangle, \cdots, \langle S_n, T_n \rangle) \in \Omega | \sigma \text{ is increasing, and } S_n \cap T_n = \varnothing\}.$$

Note that a sequence $\sigma \in \Sigma$ represents a possible stage in the *unfinished* negotiation process starting with S_0 and T_0 since $S_n \cap T_n = \varnothing$. Given a sequence σ, we abbreviate $(\langle S_0, T_0 \rangle, \cdots, \langle S_i, T_i \rangle)$ as σ_i for each $i < n$.

To assure that there is a mechanism to select which of the two parties should make concession at each negotiation stage σ, Booth[12] assumes that there exists an abstractive function $g : \Sigma \to 2^{\mathcal{B}}$ such that

$$(g0)\ \ \forall \sigma \in \Sigma,\ \varnothing \neq g(\sigma) \subseteq \{S_n, T_n\},\ \text{where } \sigma = (\langle S_0, T_0 \rangle, \ldots, \langle S_n, T_n \rangle).$$

In other words, the function g assures that there is at least one party having to make concession at stage σ, then the negotiation can proceed.

Moreover, for each $\sigma = (\langle S_0, T_0 \rangle, \cdots, \langle S_n, T_n \rangle)$, Booth[12] also assumes that the weakening of S_n (resp. T_n) is strict if S_n(resp. T_n) must be weakened at stage

σ, i.e., there exists a function $\blacktriangledown_\sigma : \{S_n, T_n\} \to \mathcal{B}$ such that:

$$(\blacktriangledown 0)\quad S_n \subseteq \blacktriangledown_\sigma(S_n) \text{ and } \blacktriangledown_\sigma(S_n) \not\subseteq S_n;$$
$$T_n \subseteq \blacktriangledown_\sigma(T_n) \text{ and } \blacktriangledown_\sigma(T_n) \not\subseteq T_n.$$

Actually, $\blacktriangledown_\sigma(S_n)$ (resp. $\blacktriangledown_\sigma(T_n)$) is the enlargement of S_n (resp. T_n), i.e., $S_n \subseteq \blacktriangledown_\sigma(S_n)$ (resp. $T_n \subseteq \blacktriangledown_\sigma(T_n)$). To avoid deadlock (i.e., $S_i \cap T_i = \varnothing$ and $S_i = S_{i+k}, T_i = T_{i+k}$, for all $k > 0$) in negotiations, this enlargement is also required to be strict according to the condition of ($\blacktriangledown 0$), i.e., $\blacktriangledown_\sigma(S_n) \not\subseteq S_n$ (resp. $\blacktriangledown_\sigma(T_n) \not\subseteq T_n$).

However, ($\blacktriangledown 0$) can be weakened as:

$$(\blacktriangledown 0)\quad S_n \subseteq \blacktriangledown_\sigma(S_n),\ T_n \subseteq \blacktriangledown_\sigma(T_n) \text{ and}$$
$$\blacktriangledown_\sigma(S_n) \cup \blacktriangledown_\sigma(T_n) \not\subseteq S_n \cup T_n.$$

That is to say, it is really only necessary that at least one party make a strict concession to avoid a deadlock.

Then a belief negotiation model is defined as follows.

Definition 1 (Belief Negotiation Model[12]). *A belief negotiation model (relative to s and t) is a pair $\mathcal{N} = \langle g, \{\blacktriangledown_\sigma\}_{\sigma\in\Sigma}\rangle$, where $g : \Sigma \to 2^{\mathcal{B}}$ is a function which satisfies ($g0$) and, for each $\sigma = (\langle S_0, T_0\rangle, \cdots, \langle S_n, T_n\rangle)$, $\blacktriangledown_\sigma : \{S_n, T_n\} \to \mathcal{B}$ is a function which satisfies ($\blacktriangledown 0$).*

As stated in [12], given any $S, T \in \mathcal{B}$, a belief negotiation model $\mathcal{N}$ can uniquely determine the complete process of negotiation between S and T, i.e., this process can be returned by the function $f^{\mathcal{N}} : \mathcal{B} \times \mathcal{B} \to \Omega$ given by

$$f^{\mathcal{N}}(S, T) = \sigma = (\langle S_0, T_0\rangle, \cdots, \langle S_n, T_n\rangle)$$

where

(a) $S_0 = S$ and $T_0 = T$,

(b) n is minimal such that $S_n \cap T_n \neq \varnothing$,

(c) for each $0 \leqslant i < n$,

$$S_{i+1} = \begin{cases} \blacktriangledown_{\sigma_i}(S_i), & \text{if } S_i \in g(\sigma_i), \\ S_i, & \text{otherwise;} \end{cases}$$
$$T_{i+1} = \begin{cases} \blacktriangledown_{\sigma_i}(T_i), & \text{if } T_i \in g(\sigma_i), \\ T_i, & \text{otherwise.} \end{cases}$$

Then we may take $Merge(S, T) = S_n \cap T_n$.

Based on $f^{\mathcal{N}}$, the following functions $f^{\mathcal{N}}_{\rightarrow}, f^{\mathcal{N}}_{\leftarrow} : \mathcal{B} \times \mathcal{B} \to \mathcal{B}$ are defined as

$$f^{\mathcal{N}}_{\rightarrow}(S, T) = S_n \text{ and } f^{\mathcal{N}}_{\leftarrow}(S, T) = T_n.$$

Evidently, $f^{\mathcal{N}}_{\rightarrow}(S, T)$ (resp. $f^{\mathcal{N}}_{\leftarrow}(S, T)$) is the result of weakening information S (resp. T) to accommodate information T(resp. S)[12].

Definition 2 (Revision Operator $\boxplus_{\mathcal{N}}$[12]). *Given a belief negotiation model $\mathcal{N}$. The revision operator $\boxplus_{\mathcal{N}}$ from $\mathcal{N}$ is defined by, for $K \in \mathcal{K}$ and $\phi \in \mathcal{L}^*_{\Phi_0}$,*

$$K \boxplus_{\mathcal{N}} \phi = Th(f^{\mathcal{N}}_{\rightarrow}([K], [\phi]) \cap f^{\mathcal{N}}_{\leftarrow}([K], [\phi])).$$

It has been shown in [12] that the revision operator $\boxplus_{\mathcal{N}}$ satisfies the basic AGM postulates for revision including *Closure*, *Extensionality*, *Inclusion*, *Vacuity* and *Consistency*.

However, if Φ is a consistent set of sentences, then we use $\bigwedge \Phi$ to denote the conjunction of all sentences in Φ. Moreover, $[\bigwedge \Phi] = [\Phi]$. Thus, we can generalize the revision operator $\boxplus_{\mathcal{N}}$ as follows:

Definition 3 (Revision Operator $\boxplus^*_{\mathcal{N}}$). *Given a belief negotiation model $\mathcal{N}$. The revision operator $\boxplus^*_{\mathcal{N}}$ from $\mathcal{N}$ is defined by, for $K \in \mathcal{K}$ and consistent $\Phi \subseteq \mathcal{L}^*_{\Phi_0}$,*

$$K \boxplus^*_{\mathcal{N}} \Phi \stackrel{\text{def}}{=} K \boxplus_{\mathcal{N}} (\bigwedge \Phi)$$
$$= Th(f^{\mathcal{N}}_{\rightarrow}([K], [\bigwedge \Phi]) \cap f^{\mathcal{N}}_{\leftarrow}([K], [\bigwedge \Phi])).$$

If there is no confusion, we often write $K \boxplus^*_{\mathcal{N}} \Phi$ as $K \boxplus_{\mathcal{N}} \Phi$.

Now we give an example to illustrate the negotiation-style revision.

Example. Consider $S = [\alpha, \neg\beta, \gamma]$ and $T = [\beta, \neg\alpha, \neg\gamma]$. Suppose that information source s is more reliable than t about the truth value of β. But t is more reliable than s about the truth value of α and γ. We start the process of negotiation-style belief revision with

$$\sigma_0 = (\langle S_0, T_0\rangle).$$

Obviously, $S_0 \cup T_0 = \varnothing$.

Suppose that $g(\sigma_0) = \{T_0\}$, $T_1 = \blacktriangledown_{\sigma_0}(T_0) = [\neg\alpha, \neg\gamma]$ and $S_1 = S_0$. Then we get

$$\sigma_1 = (\langle S_0, T_0\rangle, \langle S_1, T_1\rangle) \text{ and } S_1 \cap T_1 = \varnothing.$$

Suppose that $g(\sigma_1) = \{S_1\}$, $S_2 = \blacktriangledown_{\sigma_1}(S_1) = [\neg\beta, \gamma]$ and $T_2 = T_1$. Then we get

$$\sigma_2 = (\langle S_0, T_0\rangle, \langle S_1, T_1\rangle, \langle S_2, T_2\rangle) \text{ and } S_2 \cap T_2 = \varnothing.$$

Suppose that $g(\sigma_2) = \{S_2\}$, $S_3 = \blacktriangledown_{\sigma_2}(S_2) = [\neg\beta]$ and $T_3 = T_2$. Then we get

$$\sigma_3 = (\langle S_0, T_0\rangle, \langle S_1, T_1\rangle, \langle S_2, T_2\rangle, \langle S_3, T_3\rangle)$$

and

$$S_3 \cap T_3 = [\neg\beta, \neg\alpha, \neg\gamma].$$

Therefore, we get

$$\{\alpha, \neg\beta, \gamma\} \boxplus_{\mathcal{N}} \{\beta, \neg\alpha, \neg\gamma\} = Th([\neg\beta, \neg\alpha, \neg\gamma]).$$

Evidently, the revised belief set conforms to the intuition.

Note that the negotiation-style revision presented by Booth[12] is only an abstractive framework. Then functions g and $\{\blacktriangledown_\sigma\}_{\sigma\in\Sigma}$ need to be instantiated based on some domain knowledge from applications.

3 Handling Requirements Changes as Negotiation-Style Revision

In this section, we will present a negotiation-style revision-based approach to handling the requirements changes when the requirements specification has been established. We start with the formal representation of requests of requirements changes.

Let $\mathcal{S}$ be the current software requirements specification and $[\mathcal{S}]$ the set of worlds in which every requirements statements in $\mathcal{S}$ is true, i.e., the set of models of $\mathcal{S}$. There are three representative atomic requirements changes in $\mathcal{S}$ as follows:

(a) add a new requirement α to the current requirements specification $\mathcal{S}$;

(b) give up an existing requirement β in $\mathcal{S}$;

(c) change an existing requirements γ in $\mathcal{S}$ into a new requirement ϕ.

We use $\phi \parallel \gamma$ to denote an atomic request of changing γ into a new requirement ϕ. If Φ and Γ are two sets of formulas, we also use $\Phi \parallel \Gamma$ to denote a request of changing requirements Γ into Φ. By convention, (a) and (b) can be denoted by $\{\alpha\} \parallel \varnothing$ and $\varnothing \parallel \{\beta\}$, respectively. In particular, $\varnothing \parallel \varnothing$ denotes there is no change.

Definition 4 (Request of Requirements Change). *Let $\mathcal{S}$ be a requirements specification. A request of requirements changes $\mathcal{R}$ with regard to $\mathcal{S}$ is defined as*

$$\mathcal{R} = \{\Phi_1 \parallel \Gamma_1, \ldots, \Phi_n \parallel \Gamma_n\},$$

where each Φ_i is a new requirements set (possibly empty), and $\varnothing \subseteq \cup_{i=1}^n \Gamma_i \subset \mathcal{S}$, $\Gamma_i \cap \Gamma_j = \varnothing$, for all $i \neq j$.

Without loss of generality, we assume that $\cup_{i=1}^n \Phi_i$ is consistent and $(\cup_{i=1}^n \Phi_i) \cap (\cup_{i=1}^n \Gamma_i) = \varnothing$. Moreover, each of requirements in Φ_i is given a level of priority. Let $\Phi = \cup_{i=1}^n \Phi_i$ and $\Gamma = \cup_{i=1}^n \Gamma_i$, then we abbreviate a request of requirements change as $\Phi \parallel \Gamma$.

Let $\mathcal{W}$ be a set of all possible worlds. Let Δ be a set of formulas and $\langle \Delta^1, \ldots, \Delta^m \rangle$ the priority-based partition or stratification of Δ. We may provide a total pre-order relationship $\preceq_\Delta$ on $\mathcal{W}$ that is induced from Δ by the *leximin* ordering strategy[21] as follows:

• *Leximin Ordering*[21]. For each i $(1 \leqslant i \leqslant m)$, let $\kappa^i(w) = \{\varphi \in \Delta^i : w \models \varphi\}$, for $w \in \mathcal{W}$. Then a leximin ordering $\preceq_\Delta$ on $\mathcal{W}$ is defined as: $w \preceq_\Delta w'$ iff

1) $|\kappa^i(w)| = |\kappa^i(w')|$ for all i, or

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

For simplicity, we abbreviate $(|\kappa^1(w)|, \ldots, |\kappa^m(w)|)$ as $|\kappa(w)|$.

Note that the *leximin* ordering over $\mathcal{W}$ is based on the lexicographical relation on $\kappa(w) = \{\varphi \in \Delta : w \models \varphi\}$ for each $w \in \mathcal{W}$. As one of the widely used ordering strategies, it considers the number of all formulas satisfied by a given interpretation w as well as the priority of each formula satisfied by w.

From the pre-order relationship $\preceq_\Delta$ over $\mathcal{W}$ induced from Δ, $\mathcal{W}$ can be stratified as

$$\langle \mathcal{W}_1, \ldots, \mathcal{W}_{n(\Delta)} \rangle,$$

where $\mathcal{W}_i$ contains all the minimal worlds of set $\cup_{j=i}^{n(\Delta)} \mathcal{W}_j$ with regard to $\preceq_\Delta$, and $n(\Delta)$ is a positive number associated with Δ. Evidently, if Δ is consistent, then $\mathcal{W}_1 = [\Delta]$.

Informally speaking, this stratification groups $\mathcal{W}$ into several partitions so as to satisfy $|\kappa(w)| = |\kappa(w')|$ for any two interpretations w and w' in the same partition.

In particular, let $\langle \mathcal{S}^1, \ldots, \mathcal{S}^m \rangle$ be the priority-based partition of $\mathcal{S}$, then we can get a stratification of $\mathcal{W}$ as

$$\langle \mathcal{W}_1^s, \ldots, \mathcal{W}_{n(\mathcal{S})}^s \rangle$$

from the pre-order relation $\preceq_\mathcal{S}$ over $\mathcal{W}$ induced from $\mathcal{S}$, in which $\mathcal{W}_1^s = [\mathcal{S}]$.

Now we give an example to illustrate this stratification of the set of worlds.

Example. Consider $\Delta = \langle \Delta^1, \Delta^2, \Delta^3 \rangle$, where $\Delta^1 = \{\alpha\}$, $\Delta^2 = \{\beta \wedge \gamma\}$, and $\Delta^3 = \{\neg\alpha \vee \gamma\}$.

We denote each possible world by a bit vector consisting of truth values of (α, β, γ), then

$$\mathcal{W} = \left\{ \begin{array}{lll} \boldsymbol{w}_1 = 111, & \boldsymbol{w}_2 = 110, & \boldsymbol{w}_3 = 101, \\ \boldsymbol{w}_4 = 100, & \boldsymbol{w}_5 = 011, & \boldsymbol{w}_6 = 010, \\ \boldsymbol{w}_7 = 001, & \boldsymbol{w}_8 = 000. & \end{array} \right\}$$

and

$$\begin{array}{ll} |\kappa(\boldsymbol{w}_1)| = (1,1,1), & |\kappa(\boldsymbol{w}_2)| = (1,0,0), \\ |\kappa(\boldsymbol{w}_3)| = (1,0,1), & |\kappa(\boldsymbol{w}_4)| = (1,0,0), \\ |\kappa(\boldsymbol{w}_5)| = (0,1,1), & |\kappa(\boldsymbol{w}_6)| = (0,0,1), \\ |\kappa(\boldsymbol{w}_7)| = (0,0,1), & |\kappa(\boldsymbol{w}_8)| = (0,0,1). \end{array}$$

We can get the stratification of $\mathcal{W}$ from $\preceq_\Delta$ over $\mathcal{W}$ induced from Δ as follows:

$$\langle \mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3, \mathcal{W}_4, \mathcal{W}_5 \rangle,$$

where $\mathcal{W}_1 = \{\boldsymbol{w}_1\} = [\Delta]$ and $\mathcal{W}_2 = \{\boldsymbol{w}_3\}$, $\mathcal{W}_3 = \{\boldsymbol{w}_2, \boldsymbol{w}_4\}$, $\mathcal{W}_4 = \{\boldsymbol{w}_5\}$, $\mathcal{W}_5 = \{\boldsymbol{w}_6, \boldsymbol{w}_7, \boldsymbol{w}_8\}$.

Given a request of requirements change $\mathcal{R} = \Phi \parallel \Gamma$, we can also define a total pre-order relationship $\preceq_\Phi$ on

$\mathcal{W}$ induced from Φ. We may write $\preceq_\Phi$ as $\preceq_\mathcal{R}$ if there is no confusion.

As mentioned earlier, there are three possible ways to execute a given request of requirements change. Intuitively, each of possible ways may be viewed as a special kind of negotiation-style revision. That is, we need to design different belief negotiation models for different ways of executing the requirements change.

3.1 Fully Accepting the Request of Requirements Change $\mathcal{R}$

Given an obligatory request of requirements change $\mathcal{R} = \Phi \parallel \Gamma$, it should be fully accepted by the current requirements specification $\mathcal{S}$. Then $\mathcal{S}$ need to

a) give up all the requirements in Γ;

b) absorb all the new requirements in Φ;

c) give up some other requirements to keep consistent with Φ if there exists inconsistency caused by absorbing Φ.

In such a process of revision, negotiation focuses on the subprocess of (c). Let s and r stand for the current requirements specification and the request of requirements change in the negotiation, respectively.

Let $\mathcal{S}_0 = \mathcal{S}\backslash\Gamma$. Suppose that the stratification of $\mathcal{W}$ induced by $\mathcal{S}_0$ is

$$\langle \mathcal{W}_1^{s_0}, \ldots, \mathcal{W}_{n(\mathcal{S}_0)}^{s_0} \rangle,$$

where $\mathcal{W}_1^{s_0} = [\mathcal{S}_0]$. Then we design a belief negotiation model appropriate to this kind of requirements change as follows.

Definition 5. *The belief negotiation model relative to s and r, denoted as $\mathcal{N}^1 = \langle g^1, \{\blacktriangledown_\sigma^1\}_{\sigma\in\Sigma}\rangle$, is defined by*

$$\begin{aligned}
S_0 &= [\mathcal{S}_0],\\
R_0 &= [\Phi];\\
g^1(\sigma_i) &= \{S_i\} \text{ if } S_i \cap R_i = \varnothing;\\
S_{i+1} &= \blacktriangledown_{\sigma_i}^1(S_i) = S_i \cup \mathcal{W}_{i+2}^{s_0},\\
R_{i+1} &= \blacktriangledown_{\sigma_i}^1(R_i) = R_i, \text{ for all } i < n;\\
\sigma &= (\langle S_0, R_0\rangle, \ldots, \langle S_n, R_n\rangle), \text{ where}\\
&\quad n = \min\{i | S_i \cap R_i \neq \varnothing\};\\
\sigma_i &= (\langle S_0, R_0\rangle, \ldots, \langle S_i, R_i\rangle), \text{ for all } i < n.
\end{aligned}$$

Furthermore, we define a revision operator $\boxplus_{\mathcal{N}^1}$ to merge $\mathcal{S}_0$ and Φ as follows:

$$\mathcal{S}_0 \boxplus_{\mathcal{N}^1} \Phi = Th(S_n \cap R_n).$$

In this belief negotiation model, $g^1(\sigma_i) = \{S_i\}$ implies that the current requirements specification is a loser of negotiation round at each stage σ_i. Note that we weaken S_i by $S_i \cup \mathcal{W}_{i+2}^{s_0}$ at the i-th round of negotiation in the belief negotiation model $\mathcal{N}^1$. A problem is whether the negotiation between s and r can end within $n(\mathcal{S}_0)$ steps. The answer is positive.

Proposition 1. *Let $\sigma = (\langle S_0, R_0\rangle, \ldots, \langle S_n, R_n\rangle)$ be the negotiation procedure in the belief negotiation model $\mathcal{N}^1$. Let $\langle \mathcal{W}_1^{s_0}, \ldots, \mathcal{W}_{n(\mathcal{S}_0)}^{s_0}\rangle$ be the stratification of $\mathcal{W}$ induced by $\mathcal{S}_0$. Then $n \leqslant n(\mathcal{S}_0) - 1$.*

Proof. $[\Phi] \neq \varnothing$ since Φ is consistent. Then there exists i $(1 \leqslant i \leqslant n(\mathcal{S}_0))$ such that $[\Phi] \cap \mathcal{W}_i^{s_0} \neq \varnothing$. Therefore, $S_0 \cap R_0 \neq \varnothing$ and $\sigma = (\langle S_0, R_0\rangle)$ if $i = 1$; $S_{i-1} \cap T_{i-1} \neq \varnothing$ and $\sigma = (\langle S_0, R_0\rangle, \ldots, \langle S_{i-1}, R_{i-1}\rangle)$, i.e., $n = i - 1$ if $i > 1$. Therefore, $n \leqslant n(\mathcal{S}_0) - 1$. □

Evidently, we can also get the following property of $\boxplus_{\mathcal{N}^1}$.

Proposition 2. *Let $\mathcal{S}$ be the current requirements specification and $\mathcal{R} = \Phi \parallel \Gamma$ a request of requirements change with regard to $\mathcal{S}$. Then*

$$\Phi \subseteq \mathcal{S}_0 \boxplus_{\mathcal{N}^1} \Phi,$$

where $\mathcal{S}_0 = \mathcal{S}\backslash\Gamma$.

Proof. Let $\sigma = (\langle S_0, R_0\rangle, \ldots, \langle S_n, R_n\rangle)$ be the negotiation procedure in the belief negotiation model $\mathcal{N}^1$, then $R_0 = \cdots = R_n = [\Phi]$ and $S_n \cap [\Phi] \neq \varnothing$. Thus,

$$\Phi \subseteq Th(S_n \cap [\Phi]) = \mathcal{S}_0 \boxplus_{\mathcal{N}^1} \Phi.$$ □

According to this proposition, Φ is fully accepted by the revised belief set.

In belief revision, $\mathcal{S}_0 \boxplus_{\mathcal{N}^1} \Phi$ is a revised belief set. However, in requirements engineering, we hope that $\mathcal{S}_0 \boxplus_{\mathcal{N}^1} \Phi$ is given in the form of subset(s) of $\mathcal{S}_0 \cup \Phi$ rather than $Th(S_n \cap R_n)$.

Let $\langle \Delta^1, \ldots, \Delta^m\rangle$ and $\langle \Theta^1, \ldots, \Theta^m\rangle$ be the priority-based partitions of Δ and Θ, respectively. We define $\Theta \sqsubseteq \Delta$ iff $\Theta^i \subseteq \Delta^i$ for all $i = 1, \ldots, m$.

Let $\langle \mathcal{S}_0^1, \ldots, \mathcal{S}_0^m\rangle$ be the priority-based partition of $\mathcal{S}_0$. We use $2^{\mathcal{S}_0}$ to denote the set of all the subsets of $\mathcal{S}_0$ in the sense of $\sqsubseteq$. We provide the leximin order relation $\prec_{leximin}$[21-22] over $2^{\mathcal{S}_0}$ as follows:

• for $S, S' \in 2^{\mathcal{S}_0}$, define $S' \prec_{leximin} S$ iff $\exists k$ such that

(a) $|\mathcal{S}_0^k \cap S'^k| > |\mathcal{S}_0^k \cap S^k|$, and

(b) for all $i < k$, $|\mathcal{S}_0^i \cap S'^i| = |\mathcal{S}_0^i \cap S^i|$.

Essentially, $\prec_{leximin}$ gives an ordering relationship of relative importance of requirements sets.

Let $Sub(\mathcal{S}_0) = \{S \sqsubseteq \mathcal{S}_0 | [S] \cap (S_n \cap R_n) \neq \varnothing\}$, then for $Sub(\mathcal{S}_0)$ we denoted by $Min(\prec_{leximin}, Sub(\mathcal{S}_0))$ the set of undominated elements of $Sub(\mathcal{S}_0)$ with respect to $\prec_{leximin}$, i.e.,

$$Min(\prec_{leximin}, Sub(\mathcal{S}_0)) = \{S \in Sub(\mathcal{S}_0) | \text{ there is no } S' \in Sub(\mathcal{S}_0) \text{ such that } S' \prec_{leximin} S\}.$$

Furthermore, we define a change operator $\uplus_{\mathcal{N}^1}$ induced by $\boxplus_{\mathcal{N}^1}$ as follows.

Definition 6. (Change Operator $\uplus_{\mathcal{N}^1}$). *Let $\mathcal{S}$ be the current requirements specification and $\mathcal{R} = \Phi \parallel \Gamma$ a request of requirements change with regard to $\mathcal{S}$. Then the change operator $\uplus_{\mathcal{N}^1}$ induced by $\boxplus_{\mathcal{N}^1}$ is defined as*

$$\mathcal{S} \uplus_{\mathcal{N}^1} \mathcal{R} = \{\Phi \cup S | S \in Min(\prec_{leximin}, Sub(\mathcal{S}_0))\},$$

where $\mathcal{S}_0 = \mathcal{S} \backslash \Gamma$ and $Sub(\mathcal{S}_0) = \{S \sqsubseteq \mathcal{S}_0 | [S] \cap (S_n \cap R_n) \neq \varnothing\}$.

Note that each $\Psi \in \mathcal{S} \uplus_{\mathcal{N}^1} \mathcal{R}$ provides a possible revised requirements specification according to the belief negotiation model $\mathcal{N}^1$. Moreover, these revised specifications are viewed as "equivalent" to each other in the sense of $\prec_{leximin}$.

Evidently, we get the following property of $\mathcal{S} \uplus_{\mathcal{N}^1} \mathcal{R}$.

Proposition 3. *Let $\mathcal{S}$ be the current requirements specification and $\mathcal{R} = \Phi \parallel \Gamma$ a request of requirements change with regard to $\mathcal{S}$. Then*

$$\forall \gamma \in \Gamma,\ \forall \Psi \in \mathcal{S} \uplus_{\mathcal{N}^1} \mathcal{R},\ \Phi \sqsubseteq \Psi \text{ and } \gamma \notin \Psi.$$

That is, the request $\mathcal{R}$ is fully accepted by $\mathcal{S}$.

Now we give an example to illustrate this negotiation-style change process.

Example. Consider $\mathcal{S} = \langle \{\alpha\}, \{\beta\}, \{\gamma, \delta\} \rangle$. Suppose that $\mathcal{R} = \{\langle \varnothing, \{\neg\beta\}, \varnothing \rangle \parallel \{\delta\}\}$ is a request of requirements change with regard to $\mathcal{S}$. Intuitively, the request $\mathcal{R}$ means that the requirements δ should be changed into the new requirements $\neg\beta$ with priority of *Medium*. If we fully accept this request, it will result in an inconsistency $\beta \wedge \neg\beta$. Now we adopt the belief negotiation model $\mathcal{N}^1$ to handle this request.

Let $\mathcal{S}_0 = \mathcal{S} \backslash \{\delta\} = \langle \{\alpha\}, \{\beta\}, \{\gamma\} \rangle$ and $\Phi = \langle \varnothing, \{\neg\beta\}, \varnothing \rangle$. We denote each possible world by a bit vector consisting of truth values of $(\alpha, \beta, \gamma, \delta)$, then

$$\mathcal{W} = \left\{ \begin{array}{lll} \boldsymbol{w}_1 = 1111, & \boldsymbol{w}_2 = 1110, & \boldsymbol{w}_3 = 1101, \\ \boldsymbol{w}_4 = 1100, & \boldsymbol{w}_5 = 1011, & \boldsymbol{w}_6 = 1010, \\ \boldsymbol{w}_7 = 1001, & \boldsymbol{w}_8 = 1000, & \boldsymbol{w}_9 = 0111, \\ \boldsymbol{w}_{10} = 0110, & \boldsymbol{w}_{11} = 0101, & \boldsymbol{w}_{12} = 0100, \\ \boldsymbol{w}_{13} = 0011, & \boldsymbol{w}_{14} = 0010, & \boldsymbol{w}_{15} = 0001, \\ \boldsymbol{w}_{16} = 0000. & & \end{array} \right\}.$$

The stratification of $\mathcal{W}$ induced by $\mathcal{S}_0$ is given as follows:

$$\langle \mathcal{W}_1^{s_0}, \mathcal{W}_2^{s_0}, \ldots, \mathcal{W}_8^{s_0} \rangle,$$

where

$$\begin{array}{ll} \mathcal{W}_1^{s_0} = \{\boldsymbol{w}_1, \boldsymbol{w}_2\},\ \mathcal{W}_2^{s_0} = \{\boldsymbol{w}_3, \boldsymbol{w}_4\}, \\ \mathcal{W}_3^{s_0} = \{\boldsymbol{w}_5, \boldsymbol{w}_6\},\ \mathcal{W}_4^{s_0} = \{\boldsymbol{w}_7, \boldsymbol{w}_8\}, \\ \mathcal{W}_5^{s_0} = \{\boldsymbol{w}_9, \boldsymbol{w}_{10}\},\ \mathcal{W}_6^{s_0} = \{\boldsymbol{w}_{11}, \boldsymbol{w}_{12}\}, \\ \mathcal{W}_7^{s_0} = \{\boldsymbol{w}_{13}, \boldsymbol{w}_{14}\},\ \mathcal{W}_8^{s_0} = \{\boldsymbol{w}_{15}, \boldsymbol{w}_{16}\}. \end{array}$$

Let $S_0 = [\mathcal{S}_0], R_0 = [\Phi]$ and $\sigma_0 = (\langle S_0, R_0 \rangle)$, where

$$\begin{array}{l} [\mathcal{S}_0] = \mathcal{W}_1^{s_0} = \{\boldsymbol{w}_1, \boldsymbol{w}_2\}, \\ [\Phi] = \{\boldsymbol{w}_5, \boldsymbol{w}_6, \boldsymbol{w}_7, \boldsymbol{w}_8, \boldsymbol{w}_{13}, \boldsymbol{w}_{14}, \boldsymbol{w}_{15}, \boldsymbol{w}_{16}\}. \end{array}$$

Obviously,

$$S_0 \cap R_0 = \varnothing.$$

Then

$$\begin{array}{l} g^1(\sigma_0) = \{S_0\}, \\ S_1 = \blacktriangledown_{\sigma_0}^1(S_0) = S_0 \cup \mathcal{W}_2^{s_0} = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3, \boldsymbol{w}_4\}, \\ R_1 = \blacktriangledown_{\sigma_0}^1(R_0) = R_0, \\ \sigma_1 = (\langle S_0, R_0 \rangle, \langle S_1, R_1 \rangle). \end{array}$$

However, no agreement is reached, since

$$S_1 \cap R_1 = \varnothing.$$

Then

$$\begin{array}{l} g^1(\sigma_1) = \{S_1\}, \\ S_2 = \blacktriangledown_{\sigma_1}^1(S_1) = S_1 \cup \mathcal{W}_3^{s_0} \\ \quad = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3, \boldsymbol{w}_4, \boldsymbol{w}_5, \boldsymbol{w}_6\}, \\ R_2 = \blacktriangledown_{\sigma_1}^1(R_1) = R_1, \\ \sigma_2 = (\langle S_0, R_0 \rangle, \langle S_1, R_1 \rangle, \langle S_2, R_2 \rangle). \end{array}$$

Now $S_2 \cap R_2 = \{\boldsymbol{w}_5, \boldsymbol{w}_6\}$, the agreement is reached. So,

$$\sigma = (\langle S_0, R_0 \rangle, \langle S_1, R_1 \rangle, \langle S_2, R_2 \rangle)$$

and

$$\mathcal{S} \uplus_{\mathcal{N}^1} \mathcal{R} = \{\langle \{\alpha\}, \{\neg\beta\}, \{\gamma\} \rangle\}.$$

The revised requirements specification is

$$\langle \{\alpha\}, \{\neg\beta\}, \{\gamma\} \rangle.$$

The request of requirements change $\mathcal{R}$ is fully accepted.

3.2 Augmenting the Current Requirements Specification $\mathcal{S}$

When the requirements specification has been established, especially in the later stage of software development life cycle, abandoning some existing requirements means abandoning most the artifacts associated with the requirements. In such cases, developers tend to only accept the request of augmenting the requirements specification rather than the request of giving up some existing requirements. Then we need to construct a negotiation-style revision that preserves the current requirements specification.

Suppose that there is a request of requirements change $\mathcal{R} = \{\Phi_1 \parallel \varnothing, \varnothing \parallel \Gamma_2, \Phi_3 \parallel \Gamma_3\}$, where

- Φ_1 is the set of new requirements to be added to the current specification $\mathcal{S}$;

• Γ_2 is the set of requirements to be abandoned by the current specification $\mathcal{S}$;

• $\Phi_3 \parallel \Gamma_3$ means that the existing requirements in Γ_3 should be changed into Φ_3.

Evidently, given $\mathcal{R}$, to preserve the current requirements specification $\mathcal{S}$,

(a) the request of $\varnothing \parallel \Gamma_2$ and $\Phi_3 \parallel \Gamma_3$ should be rejected;

(b) $\mathcal{S}$ absorbs the new requirements in Φ_1;

(c) Φ_1 gives up some new requirements to keep consistent with $\mathcal{S}$ if there exists inconsistency caused by absorbing Φ_1.

In such a process of revision, negotiation focuses on the subprocess of (c). Let s and r stand for the current requirements specification and the request of requirements change in the negotiation, respectively.

Suppose that the stratification of $\mathcal{W}$ induced by Φ_1 is

$$\langle \mathcal{W}_1^r, \ldots, \mathcal{W}_{n(\Phi_1)}^r \rangle,$$

where $\mathcal{W}_1^r = [\Phi_1]$. Then we design a belief negotiation model appropriate to this kind of requirements change as follows.

Definition 7. *The belief negotiation model relative to s and r, denoted as $\mathcal{N}^2 = \langle g^2, \{\blacktriangledown_\sigma^2\}_{\sigma\in\Sigma}\rangle$, is defined by*

$$\begin{aligned}
S_0 &= [\mathcal{S}],\\
R_0 &= [\Phi_1];\\
g^2(\sigma_i) &= \{R_i\} \text{ if } S_i \cap R_i = \varnothing;\\
S_{i+1} &= \blacktriangledown_{\sigma_i}^2(S_i) = S_i,\\
R_{i+1} &= \blacktriangledown_{\sigma_i}^2(R_i) = R_i \cup \mathcal{W}_{i+2}^r, \text{ for all } i < n;\\
\sigma &= (\langle S_0, R_0\rangle, \ldots, \langle S_n, R_n\rangle), \text{ where}\\
&\quad n = \min\{i | S_i \cap R_i \neq \varnothing\};\\
\sigma_i &= (\langle S_0, R_0\rangle, \ldots, \langle S_i, R_i\rangle), \text{ for all } i < n.
\end{aligned}$$

Furthermore, we define a revision operator $\boxplus_{\mathcal{N}^2}$ to merge $\mathcal{S}$ and Φ_1 as follows:

$$\mathcal{S} \boxplus_{\mathcal{N}^2} \Phi_1 = Th(S_n \cap R_n).$$

In this belief negotiation model, $g^2(\sigma_i) = \{R_i\}$ implies that the request of requirements change is a loser of negotiation round at each stage σ_i. Note that we weaken R_i by $R_i \cup \mathcal{W}_{i+2}^r$ at the i-th round of negotiation in the belief negotiation model $\mathcal{N}^2$. Similar to $\mathcal{N}^1$, we can get the following propositions about $\mathcal{N}^2$.

Proposition 4. *Let $\sigma = (\langle S_0, R_0\rangle, \ldots, \langle S_n, R_n\rangle)$ be the negotiation procedure in the belief negotiation model $\mathcal{N}^2$. Let $\langle \mathcal{W}_1^r, \ldots, \mathcal{W}_{n(\Phi_1)}^r\rangle$ be the stratification of $\mathcal{W}$ induced by Φ_1. Then $n \leqslant n(\Phi_1) - 1$.*

Proposition 5. *Let $\mathcal{S}$ be the current requirements specification and $\mathcal{R} = \{\Phi_1 \parallel \varnothing, \varnothing \parallel \Gamma_2, \Phi_3 \parallel \Gamma_3\}$ a request of requirements change with regard to $\mathcal{S}$. Then*

$$\mathcal{S} \subseteq \mathcal{S} \boxplus_{\mathcal{N}^2} \Phi_1.$$

This proposition implies that the current requirements specification $\mathcal{S}$ is fully preserved during the revision process. Then the revised result can be viewed as an augmentation of $\mathcal{S}$.

Let $Sub(\Phi_1) = \{\Theta \sqsubseteq \Phi_1 | [\Theta] \cap (S_n \cap R_n) \neq \varnothing\}$, then for $Sub(\Phi_1)$ we denoted by $Min(\prec_{leximin}, Sub(\Phi_1))$ the set of undominated elements of $Sub(\Phi_1)$ with respect to $\prec_{leximin}$, i.e.,

$$\begin{aligned}
&Min(\prec_{leximin}, Sub(\Phi_1)) = \{\Theta \in Sub(\Phi_1) | \text{ there}\\
&\text{is no } \Theta' \in Sub(\Phi_1) \text{ such that } \Theta' \prec_{leximin} \Theta\}.
\end{aligned}$$

We define a change operator $\uplus_{\mathcal{N}^2}$ induced by $\boxplus_{\mathcal{N}^2}$ as follows.

Definition 8. (Change Operator $\uplus_{\mathcal{N}^2}$). *Let $\mathcal{S}$ be the current requirements specification and $\mathcal{R} = \{\Phi_1 \parallel \varnothing, \varnothing \parallel \Gamma_2, \Phi_3 \parallel \Gamma_3\}$ a request of requirements change with regard to $\mathcal{S}$. Then the change operator $\uplus_{\mathcal{N}^2}$ induced by $\boxplus_{\mathcal{N}^2}$ is defined as*

$$\mathcal{S} \uplus_{\mathcal{N}^2} \mathcal{R} = \big\{\mathcal{S} \cup \Theta | \Theta \in Min(\prec_{leximin}, Sub(\Phi_1))\big\},$$

where $Sub(\Phi_1) = \{\Theta \sqsubseteq \Phi_1 | [\Theta] \cap (S_n \cap R_n) \neq \varnothing\}$.

Note that each $\Psi \in \mathcal{S} \uplus_{\mathcal{N}^2} \mathcal{R}$ provides a possible revised requirements specification according to $\mathcal{N}^2$. Moreover, these revised specifications are viewed as "equivalent" to each other in the sense of $\prec_{leximin}$.

The following proposition shows that the change operator $\uplus_{\mathcal{N}^2}$ is appropriate to augmenting the current requirements specification.

Proposition 6. *Let $\mathcal{S}$ be the current requirements specification and $\mathcal{R} = \{\Phi_1 \parallel \varnothing, \varnothing \parallel \Gamma_2, \Phi_3 \parallel \Gamma_3\}$ a request of requirements change with regard to $\mathcal{S}$. Then*

$$\forall \Psi \in \mathcal{S} \uplus_{\mathcal{N}^2} \mathcal{R}, \ \mathcal{S} \sqsubseteq \Psi \sqsubseteq \mathcal{S} \cup \Phi_1.$$

Now we give an example to illustrate the augmentation of the current requirements specification.

Example. Consider $\mathcal{S} = \langle\{\alpha\}, \{\beta\}, \{\gamma\}\rangle$. Suppose that

$$\mathcal{R} = \{\langle \varnothing, \{\phi\}, \{\neg\gamma\}\rangle \parallel \varnothing\}$$

is a request of requirements change with regard to $\mathcal{S}$. Intuitively, according to the request $\mathcal{R}$, the new requirements ϕ with the priority of *Medium* and $\neg\gamma$ with the priority of *Low* should be added to the current requirements specification $\mathcal{S}$. If we want to fully preserve the current requirements, then we adopt the belief negotiation model $\mathcal{N}^2$ to handle this request.

Let $\Phi_1 = \langle \varnothing, \{\phi\}, \{\neg\gamma\}\rangle$. We denote each possible world by a bit vector consisting of truth values of

$(\alpha, \beta, \gamma, \phi)$, then

$$\mathcal{W} = \left\{ \begin{array}{lll} \boldsymbol{w}_1 = 1111, & \boldsymbol{w}_2 = 1110, & \boldsymbol{w}_3 = 1101, \\ \boldsymbol{w}_4 = 1100, & \boldsymbol{w}_5 = 1011, & \boldsymbol{w}_6 = 1010, \\ \boldsymbol{w}_7 = 1001, & \boldsymbol{w}_8 = 1000, & \boldsymbol{w}_9 = 0111, \\ \boldsymbol{w}_{10} = 0110, & \boldsymbol{w}_{11} = 0101, & \boldsymbol{w}_{12} = 0100, \\ \boldsymbol{w}_{13} = 0011, & \boldsymbol{w}_{14} = 0010, & \boldsymbol{w}_{15} = 0001, \\ \boldsymbol{w}_{16} = 0000. & & \end{array} \right\}.$$

The stratification of $\mathcal{W}$ induced by Φ_1 is given as follows:

$$\langle \mathcal{W}_1^r, \mathcal{W}_2^r, \mathcal{W}_3^r, \mathcal{W}_4^r \rangle,$$

where

$$\begin{aligned} \mathcal{W}_1^r &= \{\boldsymbol{w}_3, \boldsymbol{w}_7, \boldsymbol{w}_{11}, \boldsymbol{w}_{15}\}, \\ \mathcal{W}_2^r &= \{\boldsymbol{w}_1, \boldsymbol{w}_5, \boldsymbol{w}_9, \boldsymbol{w}_{13}\}, \\ \mathcal{W}_3^r &= \{\boldsymbol{w}_4, \boldsymbol{w}_8, \boldsymbol{w}_{12}, \boldsymbol{w}_{16}\}, \\ \mathcal{W}_4^r &= \{\boldsymbol{w}_2, \boldsymbol{w}_6, \boldsymbol{w}_{10}, \boldsymbol{w}_{14}\}. \end{aligned}$$

Let $S_0 = [\mathcal{S}], R_0 = [\Phi_1]$ and $\sigma_0 = (\langle S_0, R_0 \rangle)$, where

$$\begin{aligned} &[\mathcal{S}] = \{\boldsymbol{w}_1, \boldsymbol{w}_2\}, \\ &[\Phi_1] = \mathcal{W}_1^r = \{\boldsymbol{w}_3, \boldsymbol{w}_7, \boldsymbol{w}_{11}, \boldsymbol{w}_{15}\}. \end{aligned}$$

We find that

$$S_0 \cap R_0 = \varnothing.$$

According to the belief negotiation model $\mathcal{N}^2$,

$$\begin{aligned} &g^2(\sigma_0) = \{R_0\}, \\ &S_1 = \blacktriangledown_{\sigma_0}^2(S_0) = S_0, \\ &R_1 = \blacktriangledown_{\sigma_0}^2(R_0) = R_0 \cup \mathcal{W}_2^r \\ &\quad = \{\boldsymbol{w}_3, \boldsymbol{w}_7, \boldsymbol{w}_{11}, \boldsymbol{w}_{15}, \boldsymbol{w}_1, \boldsymbol{w}_5, \boldsymbol{w}_9, \boldsymbol{w}_{13}\}, \\ &\sigma_1 = (\langle S_0, R_0 \rangle, \langle S_1, R_1 \rangle). \end{aligned}$$

An agreement has been reached, since

$$S_1 \cap R_1 = \{\boldsymbol{w}_1\}.$$

So,

$$\sigma = (\langle S_0, R_0 \rangle, \langle S_1, R_1 \rangle)$$

and

$$\mathcal{S} \uplus_{\mathcal{N}^2} \mathcal{R} = \{\langle \{\alpha\}, \{\beta, \phi\}, \{\gamma\} \rangle\}.$$

The revised requirements specification is

$$\langle \{\alpha\}, \{\beta, \phi\}, \{\gamma\} \rangle.$$

Moreover,

$$\mathcal{S} \sqsubseteq \langle \{\alpha\}, \{\beta, \phi\}, \{\gamma\} \rangle.$$

Evidently, the requirements specification $\mathcal{S}$ is augmented by adding ϕ, whilst the request of adding $\neg\gamma$ is rejected.

3.3 Reaching a Compromise Between S and $\mathcal{R}$

The belief negotiation model $\mathcal{N}^1$ forces the party of the current requirements specification to make concessions at each stage of the negotiation procedure over executing the requirements change. In contrast, the belief negotiation model $\mathcal{N}^2$ forces the party of the request of requirements change to make concessions at each stage of the negotiation process. However, it is necessary to make some concessions for both the current requirements specification and the request of requirements change in many cases.

For a request of giving up some current requirements $\mathcal{R} = \{\varnothing \parallel \Gamma\}$ with regard to $\mathcal{S}$, there is no inconsistency caused by fully or partially acceptance of the request. Thus, we focus on the typical kinds of request of requirements change $\mathcal{R} = \{\Phi \parallel \Gamma\}$:

1) if $\Gamma = \varnothing$, $\mathcal{R}$ is to add the new requirements in Φ to $\mathcal{S}$;

2) if $\Gamma \neq \varnothing$, $\mathcal{R}$ is to change the existing requirements in Γ into the new requirements in Φ.

Just for simplicity, we assume that $\mathcal{S}$ is willing to abandon requirements in Γ. That is, $\mathcal{S}$ focuses on negotiation over abandoning some other requirements to accommodate the requirements in Φ.

Let $\mathcal{S}_0 = \mathcal{S} \backslash \Gamma$. Suppose that the stratification of $\mathcal{W}$ induced by $\mathcal{S}_0$ is

$$\langle \mathcal{W}_1^{s_0}, \ldots, \mathcal{W}_{n(\mathcal{S}_0)}^{s_0} \rangle,$$

where $\mathcal{W}_1^{s_0} = [\mathcal{S}_0]$. The stratification of $\mathcal{W}$ induced by Φ is

$$\langle \mathcal{W}_1^r, \ldots, \mathcal{W}_{n(\Phi)}^r \rangle,$$

where $\mathcal{W}_1^r = [\Phi]$.

Then we design a belief negotiation model appropriate to partially accepting $\mathcal{R} = \{\Phi \parallel \Gamma\}$ as follows.

Definition 9. *The belief negotiation model relative to s and r, denoted as $\mathcal{N}^3 = \langle g^3, \{\blacktriangledown_\sigma^3\}_{\sigma \in \Sigma} \rangle$, is defined by*

$$\begin{aligned} S_0 &= [\mathcal{S}_0], \\ R_0 &= [\Phi]; \\ g^3(\sigma_i) &= \{S_i, R_i\} \ \textit{if} \ S_i \cap R_i = \varnothing; \\ S_{i+1} &= \blacktriangledown_{\sigma_i}^3(S_i) = S_i \cup \mathcal{W}_{i+2}^{s_0}, \\ R_{i+1} &= \blacktriangledown_{\sigma_i}^3(R_i) = R_i \cup \mathcal{W}_{i+2}^r, \ \textit{for all} \ i < n; \\ \sigma &= (\langle S_0, R_0 \rangle, \cdots, \langle S_n, R_n \rangle), \ \textit{where} \\ &\quad n = \min\{i | S_i \cap R_i \neq \varnothing\}; \\ \sigma_i &= (\langle S_0, R_0 \rangle, \ldots, \langle S_i, R_i \rangle), \ \textit{for all} \ i < n. \end{aligned}$$

Furthermore, we define a revision operator $\boxplus_{\mathcal{N}^3}$ to merge $\mathcal{S}_0$ and Φ as follows:

$$\mathcal{S}_0 \boxplus_{\mathcal{N}^3} \Phi = Th(S_n \cap R_n).$$

Compared to $\mathcal{N}^1$ and $\mathcal{N}^2$, $g^3(\sigma_i)=\{S_i,R_i\}$ implies that both the current requirements specification and the request need to make concessions at each stage in $\mathcal{N}^3$ before reaching an agreement.

Proposition 7. *Let $\sigma=(\langle S_0,R_0\rangle,\ldots,\langle S_n,R_n\rangle)$ be the negotiation procedure in the belief negotiation model $\mathcal{N}^3$. Let $\langle \mathcal{W}_1^{s_0},\ldots,\mathcal{W}_{n(\mathcal{S}_0)}^{s_0}\rangle$ and $\langle \mathcal{W}_1^{r},\ldots,\mathcal{W}_{n(\Phi)}^{r}\rangle$, be the stratifications of $\mathcal{W}$ induced by $\mathcal{S}_0$ and Φ, respectively. Then*

$$n\leqslant \min\{n(\mathcal{S}_0)-1,n(\Phi)-1\}.$$

According to this proposition, the agreement will be reached within $\min\{n(\mathcal{S}_0)-1,n(\Phi)-1\}$ steps.

For each $w\in S_n\cap R_n$, suppose that $w\in\mathcal{W}_i^{s_0}$ and $w\in\mathcal{W}_j^{r}$, then we define $s(w)$ and $r(w)$ by

$$s(w)=i,\ r(w)=j.$$

Then we define

$$\begin{aligned}W_1=&\{w\in S_n\cap R_n|\text{there is no } w'\in S_n\cap R_n\\&\text{such that } s(w')<s(w)\},\\W_2=&\{w\in S_n\cap R_n|\text{there is no } w'\in S_n\cap R_n\\&\text{such that } r(w')<r(w)\}.\end{aligned}$$

Further, we define

$$\begin{aligned}&Sub_1(\mathcal{S}_0\cup\Phi)=\{\Psi\sqsubseteq\mathcal{S}_0\cup\Phi|[\Psi]\cap W_1\neq\varnothing\},\\&Sub_2(\mathcal{S}_0\cup\Phi)=\{\Psi\sqsubseteq\mathcal{S}_0\cup\Phi|[\Psi]\cap W_2\neq\varnothing\},\\&Sub_3(\mathcal{S}_0\cup\Phi)=\{\Psi\sqsubseteq\mathcal{S}_0\cup\Phi|[\Psi]\cap (S_n\cap R_n)\neq\varnothing\}.\end{aligned}$$

If the developers tend to support the current requirements specification $\mathcal{S}$, then we define a change operator $\uplus_{\mathcal{N}_1^3}$ as follows.

Definition 10 (Change Operator $\uplus_{\mathcal{N}_1^3}$). *Let $\mathcal{S}$ be the current requirements specification and $\mathcal{R}=\{\Phi\parallel\Gamma\}$ a request of requirements change with regard to $\mathcal{S}$. Then the change operator $\uplus_{\mathcal{N}_1^3}$ induced by $\boxplus_{\mathcal{N}^3}$ is defined as*

$$\mathcal{S}\uplus_{\mathcal{N}_1^3}\mathcal{R}=\{\Psi|\Psi\in Min(\prec_{leximin},Sub_1(\mathcal{S}_0\cup\Phi))\}.$$

If the developers tend to support the request of requirements change $\mathcal{R}$, then we define the change operator $\uplus_{\mathcal{N}_2^3}$ as follows.

Definition 11 (Change Operator $\uplus_{\mathcal{N}_2^3}$). *Let $\mathcal{S}$ be the current requirements specification and $\mathcal{R}=\{\Phi\parallel\Gamma\}$ a request of requirements change with regard to $\mathcal{S}$. Then the change operator $\uplus_{\mathcal{N}_2^3}$ induced by $\boxplus_{\mathcal{N}^3}$ is defined as*

$$\mathcal{S}\uplus_{\mathcal{N}_2^3}\mathcal{R}=\{\Psi|\Psi\in Min(\prec_{leximin},Sub_2(\mathcal{S}_0\cup\Phi))\}.$$

If the developers tend to balance $\mathcal{S}$ against $\mathcal{R}$, then we define the change operator $\uplus_{\mathcal{N}_3^3}$ as follows.

Definition 12 (Change Operator $\uplus_{\mathcal{N}_3^3}$). *Let $\mathcal{S}$ be the current requirements specification and $\mathcal{R}=\{\Phi\parallel\Gamma\}$ a request of requirements change with regard to $\mathcal{S}$. Then the change operator $\uplus_{\mathcal{N}_3^3}$ induced by $\boxplus_{\mathcal{N}^3}$ is defined as*

$$\mathcal{S}\uplus_{\mathcal{N}_3^3}\mathcal{R}=\{\Psi|\Psi\in Min(\prec_{leximin},Sub_3(\mathcal{S}_0\cup\Phi))\}.$$

These change operators are appropriate to reaching different compromises between the current requirements specification and the request of requirements change.

It is not hard to get the following property of the three change operators.

Proposition 8. *Let $\mathcal{S}$ be the current requirements specification and $\mathcal{R}=\{\Phi\parallel\Gamma\}$ a request of requirements change with regard to $\mathcal{S}$. Let $\mathcal{S}_0=\mathcal{S}\backslash\Gamma$. If $\mathcal{S}_0\cup\Phi$ is consistent, then*

$$\mathcal{S}\uplus_{\mathcal{N}_i^3}\mathcal{R}=\{\mathcal{S}_0\cup\Phi\},\ \textit{for } i=1,2,3.$$

Example. Consider $\mathcal{S}=\langle\{\alpha\},\{\beta\},\{\gamma,\delta\}\rangle$ and $\mathcal{R}=\{\langle\{\neg\gamma\},\varnothing,\{\neg\beta\}\rangle\parallel\{\delta\}\}$. Then $\mathcal{S}_0=\langle\{\alpha\},\{\beta\},\{\gamma\}\rangle$ and $\Phi=\langle\{\neg\gamma\},\varnothing,\{\neg\beta\}\rangle$. We denote each possible world by a bit vector consisting of truth values of (α,β,γ), then

$$\mathcal{W}=\left\{\begin{array}{lll}\boldsymbol{w}_1=111, & \boldsymbol{w}_2=110, & \boldsymbol{w}_3=101,\\ \boldsymbol{w}_4=100, & \boldsymbol{w}_5=011, & \boldsymbol{w}_6=010,\\ \boldsymbol{w}_7=001, & \boldsymbol{w}_8=000. & \end{array}\right\}.$$

Then the stratification of $\mathcal{W}$ induced by $\mathcal{S}_0$ is given as follows:

$$\langle\{\boldsymbol{w}_1\},\{\boldsymbol{w}_2\},\{\boldsymbol{w}_3\},\{\boldsymbol{w}_4\},\{\boldsymbol{w}_5\},\{\boldsymbol{w}_6\},\{\boldsymbol{w}_7\},\{\boldsymbol{w}_8\}\rangle.$$

The stratification of $\mathcal{W}$ induced by Φ is given as follows:

$$\langle\{\boldsymbol{w}_4,\boldsymbol{w}_8\},\{\boldsymbol{w}_2,\boldsymbol{w}_6\},\{\boldsymbol{w}_3,\boldsymbol{w}_7\},\{\boldsymbol{w}_1,\boldsymbol{w}_5\}\rangle.$$

Let $S_0=[\mathcal{S}_0],R_0=[\Phi]$ and $\sigma_0=(\langle S_0,R_0\rangle)$, where

$$\begin{aligned}&[\mathcal{S}_0]=\{\boldsymbol{w}_1\},\\&[\Phi]=\{\boldsymbol{w}_4,\boldsymbol{w}_8\}.\end{aligned}$$

We find that

$$S_0\cap R_0=\varnothing.$$

According to the belief negotiation model $\mathcal{N}^3$,

$$\begin{aligned}&g^3(\sigma_0)=\{S_0,R_0\},\\&S_1=\blacktriangledown_{\sigma_0}^3(S_0)=\{\boldsymbol{w}_1,\boldsymbol{w}_2\},\\&R_1=\blacktriangledown_{\sigma_0}^3(R_0)=\{\boldsymbol{w}_4,\boldsymbol{w}_8,\boldsymbol{w}_2,\boldsymbol{w}_6\},\\&\sigma_1=(\langle S_0,R_0\rangle,\langle S_1,R_1\rangle).\end{aligned}$$

An agreement has been reached, since

$$S_1\cap R_1=\{\boldsymbol{w}_2\}.$$

Further, the revised requirements specification is

$$\mathcal{S} \uplus_{\mathcal{N}_i^3} \mathcal{R} = \langle \{\alpha, \neg\gamma\}, \{\beta\}, \varnothing \rangle \text{ for each } i.$$

This result means that the original requirements specification abandoned the requirement γ, and the request of requirements change abandoned the new requirements $\neg\beta$. This compromise is intuitive.

Generally, when a request of requirements change is submitted to the Change Control Board, the Change Control Board may use different change operators to simulate the different change processes. These possible results of revision can help developers and CCB to make reasonable trade-off decisions about the request of requirements change.

3.4 A Case Study

In order to illustrate the models for managing requirements changes based on negotiation-style revision, we have provided a series of small examples. Although we do not expect our approach can be immediately applied to scenarios in an industrial setting, we can demonstrate how complexity can be reduced using our approaches by restricting them to smaller partial specifications, and show its potential usefulness when conjuncted with a host of other techniques in requirements engineering. From this perspective, we provide a simple but explanatory case study below.

As argued in [20], most of requirements specifications in industrial setting are expressed in natural language rather than logical formulas directly. Therefore we start our case study with requirements specification in natural language. Then we formulate these requirements and the request of requirements change in logical formulas. Following this, we illustrate how to negotiate over requirements changes by use of appropriate models. Finally, we discuss some potential supports for application of our approaches to requirements engineering.

Example. Consider the following requirements specification of a computer-aided close residential area management system, which is concentrated on managing the entrance of vehicles to the residential area.

• Requirements assigned to the priority level of *High*:

(h1) the vehicles without special authorization of the Management Board of the residential area cannot be allowed to enter the area;

(h2) the vehicles with a special authorization of the Management Board of the residential area can enter the area;

(h3) the system should trigger warning alarm if a vehicle without authorization enters the area.

• Requirements assigned to the priority level of *Medium*:

(m1) If the system trigger warning alarm, the vehicle cannot push the button for entrance again.

At first, we translate these requirements into logical formulas. Suppose that we use

• the predicate $\mathtt{Aut(x)}$ to denote that $\mathtt{x}$ is authorized by the Management Board of the residential area;

• the predicate $\mathtt{Ent(x)}$ to denote that $\mathtt{x}$ can enter the residential area;

• the predicate $\mathtt{Ala(x)}$ to denote that the system triggers alarm if $\mathtt{x}$ enters the area;

• the predicate $\mathtt{Pus(x, y)}$ to denote that $\mathtt{x}$ pushes the button $\mathtt{y}$;

• the constant $\mathtt{entr}$ to denote the button of entrance.

Then we can use the following set of formulas to represent the preliminary requirements specification:

$$\mathcal{S} = \langle \mathcal{S}_{\mathtt{High}}, \mathcal{S}_{\mathtt{Medium}}, \mathcal{S}_{\mathtt{Low}} \rangle,$$

where

$$\mathcal{S}_{\mathtt{High}} = \left\{ \begin{array}{l} (\forall \mathtt{x})(\neg \mathtt{Aut(x)} \rightarrow \neg \mathtt{Ent(x)}), \\ (\forall \mathtt{x})(\mathtt{Aut(x)} \rightarrow \mathtt{Ent(x)}), \\ (\forall \mathtt{x})(\neg \mathtt{Aut(x)} \rightarrow \mathtt{Ala(x)}) \end{array} \right\},$$

$$\mathcal{S}_{\mathtt{Medium}} = \{(\forall \mathtt{x})(\mathtt{Ala(x)} \rightarrow \neg \mathtt{Pus(x, entr)})\},$$

$$\mathcal{S}_{\mathtt{Low}} = \varnothing.$$

During the development, the emergency manager told the vehicles entrance manager that the system should allow the vehicle of emergency such as fine engines to enter the area. To guarantee this, they provided the following constraints that must be met:

(r1) The fire engine should be viewed as the vehicle of emergency.

(r2) The vehicle of emergency can enter the area, and need not to be authorized by the Management Board of the residential area in advance.

(r3) The vehicles without authorization except that for emergency cannot enter the area.

Suppose that we use

• the constant $\mathtt{fire\_e}$ to denote the fire engine;

• the predicate $\mathtt{Eme(x)}$ to denote that $\mathtt{x}$ is a vehicle for emergency.

Then we can instantiate the requirements about the fire engine as follows:

$$\mathcal{S}_{\mathcal{F}} = \langle \mathcal{S}_{\mathcal{F}\mathtt{High}}, \mathcal{S}_{\mathcal{F}\mathtt{Medium}}, \mathcal{S}_{\mathcal{F}\mathtt{Low}} \rangle,$$

where

$$\mathcal{S}_{\mathcal{F}\mathtt{High}} = \left\{ \begin{array}{c} \neg \mathtt{Aut(fire\_e)} \rightarrow \neg \mathtt{Ent(fire\_e)}, \\ \mathtt{Aut(fire\_e)} \rightarrow \mathtt{Ent(fire\_e)}, \\ \neg \mathtt{Aut(fire\_e)} \rightarrow \mathtt{Ala(fire\_e)} \end{array} \right\},$$

$$\mathcal{S}_{\mathcal{F}\mathtt{Medium}} = \{\mathtt{Ala(fire\_e)} \rightarrow \neg \mathtt{Pus(fire\_e, entr)}\},$$

$$\mathcal{S}_{\mathcal{F}\mathtt{Low}} = \varnothing.$$

Correspondingly, the request for requirements change can be formulated by

$$\mathcal{R}_1 = \{\Phi_1 \parallel \varnothing\},$$

where

$$\Phi_1 = \left\langle \left\{ \begin{array}{l} \texttt{Eme(fire\_e)}, \\ \texttt{Eme(fire\_e)} \\ \rightarrow \texttt{Ent(fire\_e)} \wedge \neg\texttt{Aut(fire\_e)}, \\ \neg\texttt{Aut(fire\_e)} \wedge \neg\texttt{Eme(fire\_e)} \\ \rightarrow \neg\texttt{Ent(fire\_e)} \end{array} \right\}, \varnothing, \varnothing \right\rangle.$$

Because this is an obligatory request of requirements change, all the requirements belonging to Φ_1 should be added to the requirements specification $\mathcal{S}_\mathcal{F}$. However,

$$\mathcal{S}_\mathcal{F} \cup \Phi_1 \vdash \texttt{Ent(fire\_e)},$$

and

$$\mathcal{S}_\mathcal{F} \cup \Phi_1 \vdash \neg\texttt{Ent(fire\_e)},$$

i.e., $\mathcal{S}_\mathcal{F} \cup \Phi_1$ is inconsistent.

To maintain consistency of requirements specification after change, the original requirements specification $\mathcal{S}_\mathcal{F}$ has to make concession. So, we adopt the belief negotiation model $\mathcal{N}^1$. We denote each possible world by a bit vector consisting of truth values of

$$\begin{array}{l}(\texttt{Aut(fire\_e)}, \texttt{Ent(fire\_e)}, \texttt{Ala(fire\_e)}, \\ \texttt{Eme(fire\_e)}, \texttt{Pus(fire\_e, entr)}),\end{array}$$

then

$$\mathcal{W} = \left\{ \begin{array}{lll} \boldsymbol{w}_1 = 11111, & \boldsymbol{w}_2 = 11110, & \boldsymbol{w}_3 = 11101, \\ \boldsymbol{w}_4 = 11100, & \boldsymbol{w}_5 = 11011, & \boldsymbol{w}_6 = 11010, \\ \boldsymbol{w}_7 = 11001, & \boldsymbol{w}_8 = 11000, & \boldsymbol{w}_9 = 10111, \\ \boldsymbol{w}_{10} = 10110, & \boldsymbol{w}_{11} = 10101, & \boldsymbol{w}_{12} = 10100, \\ \boldsymbol{w}_{13} = 10011, & \boldsymbol{w}_{14} = 10010, & \boldsymbol{w}_{15} = 10001, \\ \boldsymbol{w}_{16} = 10000, & \boldsymbol{w}_{17} = 01111, & \boldsymbol{w}_{18} = 01110, \\ \boldsymbol{w}_{19} = 01101, & \boldsymbol{w}_{20} = 01100, & \boldsymbol{w}_{21} = 01011, \\ \boldsymbol{w}_{22} = 01010, & \boldsymbol{w}_{23} = 01001, & \boldsymbol{w}_{24} = 01000, \\ \boldsymbol{w}_{25} = 00111, & \boldsymbol{w}_{26} = 00110, & \boldsymbol{w}_{27} = 00101, \\ \boldsymbol{w}_{28} = 00100, & \boldsymbol{w}_{29} = 00011, & \boldsymbol{w}_{30} = 00010, \\ \boldsymbol{w}_{31} = 00001, & \boldsymbol{w}_{32} = 00000 & \end{array} \right\}.$$

The stratification of $\mathcal{W}$ induced by $\mathcal{S}_\mathcal{F}$ is given as follows:

$$\langle \mathcal{W}_1^{s_\mathcal{F}}, \mathcal{W}_2^{s_\mathcal{F}}, \ldots, \mathcal{W}_5^{s_\mathcal{F}} \rangle,$$

where

$$\begin{array}{l} \mathcal{W}_1^{s_\mathcal{F}} = \{\boldsymbol{w}_2, \boldsymbol{w}_4, \boldsymbol{w}_5, \boldsymbol{w}_6, \boldsymbol{w}_7, \boldsymbol{w}_8, \boldsymbol{w}_{26}, \boldsymbol{w}_{28}\}, \\ \mathcal{W}_2^{s_\mathcal{F}} = \{\boldsymbol{w}_1, \boldsymbol{w}_3, \boldsymbol{w}_{25}, \boldsymbol{w}_{27}\}, \\ \mathcal{W}_3^{s_\mathcal{F}} = \{\boldsymbol{w}_{10}, \boldsymbol{w}_{12}, \boldsymbol{w}_{13}, \boldsymbol{w}_{14}, \boldsymbol{w}_{15}, \boldsymbol{w}_{16}, \boldsymbol{w}_{18}, \\ \qquad \boldsymbol{w}_{20}, \boldsymbol{w}_{29}, \boldsymbol{w}_{30}, \boldsymbol{w}_{31}, \boldsymbol{w}_{32}\}, \\ \mathcal{W}_4^{s_\mathcal{F}} = \{\boldsymbol{w}_9, \boldsymbol{w}_{11}, \boldsymbol{w}_{17}, \boldsymbol{w}_{19}\}, \\ \mathcal{W}_5^{s_\mathcal{F}} = \{\boldsymbol{w}_{21}, \boldsymbol{w}_{22}, \boldsymbol{w}_{23}, \boldsymbol{w}_{24}\}. \end{array}$$

The detailed negotiation process is given as follows.

• Consider the initial stage:

Let $S_0 = [\mathcal{S}_\mathcal{F}], R_0 = [\Phi_1]$ and $\sigma_0 = (\langle S_0, R_0 \rangle)$, where

$$\begin{array}{l} [\mathcal{S}_\mathcal{F}] = \mathcal{W}_1^{s_\mathcal{F}} \\ \qquad = \{\boldsymbol{w}_2, \boldsymbol{w}_4, \boldsymbol{w}_5, \boldsymbol{w}_6, \boldsymbol{w}_7, \boldsymbol{w}_8, \boldsymbol{w}_{26}, \boldsymbol{w}_{28}\}, \\ [\Phi_1] = \{\boldsymbol{w}_{17}, \boldsymbol{w}_{18}, \boldsymbol{w}_{21}, \boldsymbol{w}_{22}\}. \end{array}$$

Obviously,

$$S_0 \cap R_0 = \varnothing.$$

• Then consider the first round of negotiation:

$$\begin{array}{l} g^1(\sigma_0) = \{S_0\}, \text{ i.e., } S_0 \text{ is the party having} \\ \qquad\qquad \text{to make concession.} \\ S_1 = \blacktriangledown^1_{\sigma_0}(S_0) = S_0 \cup \mathcal{W}_2^{s_\mathcal{F}} \\ \qquad = \{\boldsymbol{w}_2, \boldsymbol{w}_4, \boldsymbol{w}_5, \boldsymbol{w}_6, \boldsymbol{w}_7, \boldsymbol{w}_8, \boldsymbol{w}_{26}, \boldsymbol{w}_{28}, \\ \qquad \boldsymbol{w}_1, \boldsymbol{w}_3, \boldsymbol{w}_{25}, \boldsymbol{w}_{27}\}, \\ R_1 = \blacktriangledown^1_{\sigma_0}(R_0) = R_0, \\ \sigma_1 = (\langle S_0, R_0 \rangle, \langle S_1, R_1 \rangle). \end{array}$$

Obviously,

$$S_1 \cap R_1 = \varnothing.$$

• Then consider the second round of negotiation:

$$\begin{array}{l} g^1(\sigma_1) = \{S_1\}, \text{ i.e., } S_1 \text{ is the party having} \\ \qquad\qquad \text{to make concession.} \\ S_2 = \blacktriangledown^1_{\sigma_1}(S_1) = S_1 \cup \mathcal{W}_3^{s_\mathcal{F}} \\ \qquad = \left\{ \begin{array}{l} \boldsymbol{w}_2, \boldsymbol{w}_4, \boldsymbol{w}_5, \boldsymbol{w}_6, \boldsymbol{w}_7, \boldsymbol{w}_8, \\ \boldsymbol{w}_{26}, \boldsymbol{w}_{28}, \boldsymbol{w}_1, \boldsymbol{w}_3, \boldsymbol{w}_{25}, \\ \boldsymbol{w}_{27}, \boldsymbol{w}_{10}, \boldsymbol{w}_{12}, \boldsymbol{w}_{13}, \\ \boldsymbol{w}_{14}, \boldsymbol{w}_{15}, \boldsymbol{w}_{16}, \boldsymbol{w}_{18}, \\ \boldsymbol{w}_{20}, \boldsymbol{w}_{29}, \boldsymbol{w}_{30}, \boldsymbol{w}_{31}, \boldsymbol{w}_{32} \end{array} \right\}, \\ R_2 = \blacktriangledown^1_{\sigma_1}(R_1) = R_1, \\ \sigma_2 = (\langle S_0, R_0 \rangle, \langle S_1, R_1 \rangle, \langle S_2, R_2 \rangle). \end{array}$$

An agreement is reached, since

$$S_2 \cap R_2 = \{\boldsymbol{w}_{18}\}.$$

Then the process of negotiation over requirements changes under this obligatory request is given as follows:

$$\sigma = \sigma_2.$$

Moreover,

$$\mathcal{S}_\mathcal{F} \uplus_{\mathcal{N}^1} \mathcal{R}_1 = \{\langle \mathcal{S}_{1_{\texttt{High}}}, \mathcal{S}_{1_{\texttt{Medium}}}, \mathcal{S}_{1_{\texttt{Low}}} \rangle\},$$

where

$$\mathcal{S}_{1_{\texttt{High}}} = \left\{ \begin{array}{l} \texttt{Aut(fire\_e)} \rightarrow \texttt{Ent(fire\_e)}, \\ \neg\texttt{Aut(fire\_e)} \rightarrow \texttt{Ala(fire\_e)}, \\ \texttt{Eme(fire\_e)}, \\ \texttt{Eme(fire\_e)} \rightarrow \\ \texttt{Ent(fire\_e)} \wedge \neg\texttt{Aut(fire\_e)}, \\ \neg\texttt{Aut(fire\_e)} \wedge \neg\texttt{Eme(fire\_e)} \\ \rightarrow \neg\texttt{Ent(fire\_e)} \end{array} \right\},$$

$$\begin{aligned} \mathcal{S}_{1_{\texttt{Medium}}} &= \mathcal{S}_{\mathcal{F}\texttt{Medium}} \\ &= \{\texttt{Ala(fire\_e)} \rightarrow \neg\texttt{Pus(fire\_e, entr)}\}, \\ \mathcal{S}_{1_{\texttt{Low}}} &= \mathcal{S}_{\mathcal{F}\texttt{Low}} = \varnothing. \end{aligned}$$

Therefore, the revised instantiated requirements specification is

$$\mathcal{S}_{\mathcal{F}1} = \langle \mathcal{S}_{1_{\texttt{High}}}, \mathcal{S}_{1_{\texttt{Medium}}}, \mathcal{S}_{1_{\texttt{Low}}} \rangle.$$

Compared to the original instantiated requirements specification, the request of requirements change $\mathcal{R}_1$ is fully accepted. To maintain consistency of requirements specification, $\mathcal{S}_{\mathcal{F}}$ makes concession and abandons the following requirement:

$$\neg\texttt{Aut(fire\_e)} \rightarrow \neg\texttt{Ent(fire\_e)}.$$

When the emergency manager verified the revised requirements specification, he found the entrance of fine engines can trigger warning alarm. So the emergency manager and the emergency manager want to replace the original requirement about triggering alarm by the following requirements with *High* level to requirements specification:

(r4) The vehicles without authorization except the fine engine should trigger warning alarm.

Then we can formulate this request of requirements change as follows:

$$\mathcal{R}_2 = \{\Phi_2 \parallel \Gamma_2\},$$

where

$$\begin{aligned} \Phi_2 &= \left\langle \left\{ \begin{array}{l} \neg\texttt{Aut(fire\_e)} \wedge \neg\texttt{Eme(fire\_e)} \\ \rightarrow \texttt{Ala(fire\_e)} \end{array} \right\}, \varnothing, \varnothing \right\rangle, \\ \Gamma_2 &= \{\neg\texttt{Aut(fire\_e)} \rightarrow \texttt{Ala(fire\_e)}\}. \end{aligned}$$

Suppose that stakeholders want to reach a compromise between $\mathcal{S}_{\mathcal{F}_1}$ and $\mathcal{R}_2$. Then we adopt the belief negotiation model $\mathcal{N}^3$. Let $\mathcal{S}_0 = \mathcal{S}_{\mathcal{F}_1} \setminus \Gamma_2$. We consider the following negotiation over requirements changes under the request $\mathcal{R}_2$:

• The initial stage:

Let $S_0 = [\mathcal{S}_0], R_0 = [\Phi_2]$ and $\sigma_0 = (\langle S_0, R_0 \rangle)$, where

$$[\mathcal{S}_0] = \{w_{18}\},$$

$$[\Phi_2] = \mathcal{W} \setminus \{\boldsymbol{w}_{23}, \boldsymbol{w}_{24}, \boldsymbol{w}_{31}, \boldsymbol{w}_{32}\}.$$

Obviously,

$$S_0 \cap R_0 = \{\boldsymbol{w}_{18}\}.$$

An agreement is reached.

Correspondingly,

$$\mathcal{S}_{\mathcal{F}1} \uplus_{\mathcal{N}_i^3} \mathcal{R}_2 = \{\mathcal{S}_0 \cup \Phi_2\}.$$

The revised instantiated requirements specification is

$$\mathcal{S}_{\mathcal{F}2} = \langle \mathcal{S}_{2_{\texttt{High}}}, \mathcal{S}_{2_{\texttt{Medium}}}, \mathcal{S}_{2_{\texttt{Low}}} \rangle,$$

where

$$\mathcal{S}_{2_{\texttt{High}}} = \left\{ \begin{array}{l} \texttt{Aut(fire\_e)} \rightarrow \texttt{Ent(fire\_e)}, \\ \neg\texttt{Aut(fire\_e)} \wedge \neg\texttt{Ele(fire\_e)} \rightarrow \\ \texttt{Ala(fire\_e)}, \\ \texttt{Eme(fire\_e)}, \\ \texttt{Eme(fire\_e)} \rightarrow \\ \texttt{Ent(fire\_e)} \wedge \neg\texttt{Aut(fire\_e)}, \\ \neg\texttt{Aut(fire\_e)} \wedge \neg\texttt{Eme(fire\_e)} \rightarrow \\ \neg\texttt{Ent(fire\_e)} \end{array} \right\},$$

$$\begin{aligned} \mathcal{S}_{2_{\texttt{Medium}}} &= \mathcal{S}_{\mathcal{R}1_{\texttt{Medium}}} \\ &= \{\texttt{Ala(fire\_e)} \rightarrow \neg\texttt{Pus(fire\_e, entr)}\}, \\ \mathcal{S}_{2_{\texttt{Low}}} &= \mathcal{S}_{\mathcal{R}1_{\texttt{Low}}} = \varnothing. \end{aligned}$$

Evidently, the request of requirements change $\mathcal{R}_2$ is fully accepted, i.e., $\mathcal{S}_{\mathcal{F}_1}$ replaced

$$\neg\texttt{Aut(fire\_e)} \rightarrow \texttt{Ala(fire\_e)}$$

by

$$\neg\texttt{Aut(fire\_e)} \wedge \neg\texttt{Ele(fire\_e)} \rightarrow \texttt{Ala(fire\_e)}.$$

Finally, based on the revised instantiated requirements specification $\mathcal{S}_{\mathcal{F}2}$, the revised preliminary requirements specification can be formulated by

$$\mathcal{S}_{\mathcal{R}} = \langle \mathcal{S}_{\mathcal{R}\texttt{High}}, \mathcal{S}_{\mathcal{R}\texttt{Medium}}, \mathcal{S}_{\mathcal{R}\texttt{Low}} \rangle,$$

where

$$\mathcal{S}_{\mathcal{R}\texttt{High}} = \left\{ \begin{array}{l} \forall\texttt{x}(\neg\texttt{Ele(x)} \rightarrow \texttt{Ent(x)} \wedge \neg\texttt{Aut(x)}), \\ \forall\texttt{x}(\texttt{Aut(x)} \rightarrow \texttt{Ent(x)}), \\ \forall\texttt{x}(\neg\texttt{Aut(x)} \wedge \neg\texttt{Ele(x)} \rightarrow \texttt{Ala(x)}), \\ \texttt{Ele(fire\_e)} \end{array} \right\},$$

$$\begin{aligned} \mathcal{S}_{\mathcal{R}\texttt{Medium}} &= \{\forall\texttt{x}(\texttt{Ala(x)} \rightarrow \neg\texttt{Pus(x, entr)})\}, \\ \mathcal{S}_{\mathcal{R}\texttt{Low}} &= \varnothing. \end{aligned}$$

That is, after negotiation between original requirements specification and the request of requirements change,

• the requirement (h1) was replaced by (r2) and (r3) together;

• the requirement (h3) was replaced by (r4);

• the new requirement (r1) was added to the requirements specification;

• requirements (h2) and (m1) remain unchanged.

This case study has illustrated how to use our models in requirements engineering step by step. The approach presented in this paper is in the form of first order logic. Actually, first order logic can be considered as a preliminary of formal methods used in requirements engineering, such as in [14, 20]. Requirements analysts with only preliminaries of first order logic are able to use these approaches easily. On the other hand, to further facilitate the application of our logic-based approaches, we need consider some tools support for translating requirements expressed in natural language into formal logic as well as translating formal logic into natural language sentences. As such, a prototype tool termed CARL for this task has been presented in [20]. Then requirements analysts may combine or integrate our models with available tools such as CARL flexibly according to their needs in requirements management.

4 Discussion and Comparison

This paper focuses on management of software requirements changes based on negotiation-style belief revision. The negotiation-style framework presented by Booth[12] does not emphasize the postulate of success. It provides a more flexible way to revise the current belief set when new information is given. The belief negotiation model defined in [12] is still a general description except that the outcomes in abstractive negotiation formalization are specified as the possible worlds or models. Specifying the belief negotiation model presented in [12] needs to focus on the problem of how to make concessions. Of course, it depends on the application domain. However, it has been increasingly recognized that the priority of requirements can help requirements analysts resolve conflicts and make some necessary trade-off decision[15-16]. Allowing for this, we take the priority of requirements into account in making concessions, and then design a family of more specified belief negotiation models appropriate for different processes of executing the requirements changes. It is embodied by stratifying the set of possible worlds $\mathcal{W}$ and adopting the strum $\mathcal{W}^s_{i+2}$ (resp. $\mathcal{W}^r_{i+2}$) to weaken S_i (resp. R_i) at stage σ_i during a negotiation process.

As mentioned earlier, viewing the evolution of requirements specification as a process of revision has been considered in [2-3, 6-7]. The analysis-revision cycle presented in [2-3] aimed to evolve the unreliable and imprecise requirements into a relative mature requirements specification. The analysis-revision cycle is interested in finding requirements to be changed as well as how to change them. Informally, the abductive reasoning is adopted to find the problematic requirements to be changeable, moreover, the inductive learning is adopted to look for the way to modify the current problematic requirements. Generally, in this process, the requirements change is fully accepted to revise the current requirements. It agrees with the first model of negotiation presented in this paper in acceptance of the request of change. In contrast, our negotiation-based models concentrate on managing the requirements changes at the development stages, in which the requirements specification has been established. In such cases, the request of requirements change is always caused by very particular or uncertain factors. The acceptance of a request of requirements change may boost the satisfactions of related stakeholders. But it must result in a series of modifications at each existing development stage. Then the current requirements specification and the request of requirements change need to reach a compromise. So, requirement specification is not always considered as the party having to make concession to accommodate the newest requirements. This makes our flexible models for formulating the requirements change necessary when requirements specification has been established.

The logic framework for modeling the evolution of requirements presented in [6-7] also concentrated on evolving underdeveloped requirements specifications into relative mature requirements specifications. At a meta-level, the requirements model is viewed as a default theory. The belief revision formalized in the AGM theory[8] is adopted to resolve inconsistencies caused by evolutionary changes. The postulate of success always makes the new requirements prioritized when it contradicts with the current requirements. Roughly speaking, under guidance of this postulate, the current requirements specification has to make concession to accept the new requirements when the union of the current requirements and the new requirements is inconsistent. In this sense, the principle of handling the request of requirement changes of this approach in accordance with that of our first negotiation model. It is useful to improve the quality of requirements especially at earlier requirements stage. But when requirements specification has established, handling requests of requirements changes often needs to balance the advantage of the current requirements specification against the disadvantage of the requirements change. Our flexible models are more appropriate to managing the request of requirements change. To illustrate this, consider $\mathcal{S} = \{a, b, c\}$ and $\mathcal{R} = \{\{\neg b, \neg c\}||\varnothing\}$. Suppose that disadvantage caused by accepting $\{\neg b, \neg c\}$ is more than advantage caused by only accepting $\neg b$, an intuitive result should be $\{a, \neg b, c\}$. Actually, it can be obtained by using the third negotiation model presented in this paper. But if we adopt the belief revision operators subject to the postulate of success such as [9, 23],

the revision must result in fully acceptance of the requirements change, i.e., the revised requirements is $\{a, \neg b, \neg c\}$ rather than $\{a, \neg b, c\}$.

On the other hand, we do not claim that negotiation should be considered as a silver bullet for managing requirements changes. Allowing for the complexity of requirements changes, some other strategies such as combinatorial vote[24] and game theory should be considered in resolving inconsistencies resulting from requirements changes if the result of negotiation is undesirable.

With regard to the computational issue about the approach presented in this paper, it may be divided into three sub-problems, i.e., the computation of the stratification of $\mathcal{W}$ induced by the original requirements $\mathcal{S}$, the computation of the stratification of $\mathcal{W}$ induced by the request of change $\mathcal{R}$, and the computation of a compromise between $\mathcal{S}$ and $\mathcal{R}$. Evidently, the computation of the stratification of $\mathcal{W}$ induced by S and the computation of the stratification of $\mathcal{W}$ induced by R play a dominant role in the whole computation process. That is, the core of potential implementation or tool support for the negotiation-based approach is to compute the stratification of possible worlds induced by a set of prioritized formulas, i.e., to find formulas satisfied by each given interpretation. Additionally, to provide practical tool support for the approach presented in this paper, we need to integrate a tool for translating requirements in natural language into logic formulas with the potential negotiation-based system. This will be the main direction for future work.

Finally, we assume that there are three representative changes in this paper, including, adding a new requirement, abandoning an existing requirement, and changing an existing requirement into a new requirement. However, in practical software projects, the requirements change may be more complex. For example, consider $\mathcal{S} = \{a, \neg a \vee b, c, b, d\}$ and $\mathcal{R} = \{\{\neg d\} \parallel \{b\}\}$. Suppose that we accept fully this request of change, then the revised set $\mathcal{S}_1 = \mathcal{S} \biguplus_{\mathcal{N}^1} \mathcal{R} = \{a, \neg a \vee b, c, \neg d\}$. Evidently, $\mathcal{S}_1$ is consistent and $b \in Cn(\mathcal{S}_1)$. How to modify the negotiation models for such cases will be also left for future work.

5 Conclusion

When the requirements specification has been established, the request of requirements change should be handled cautiously. The acceptance of the request of requirements change may boost the satisfactions of stakeolders and enhance the quality of software product. But it must also result in a series of modifications at each existing development stage. Generally, it is really necessary to reach a compromise between the current requirements specification and the request of requirements change.

We have presented an approach to managing the requirements changes based on negotiation-style belief revision. Informally speaking, the current requirements specification is viewed as an existing set of beliefs of stakeholder about the system-to-be, whilst the request of requirements change is viewed as new information about the system-to-be. Then we consider the current requirements specification and the request of requirements change as two parties in negotiation over revising the beliefs about the system-to-be. We design a family of belief negotiation models appropriate to different processes of requirements revision, including the setting of the request being fully accepted, the setting of the current requirements specification being fully preserved, and that of the two parties reaching a compromise. In particular, we consider the priority of requirements in making concessions by stratifying the possible worlds set and weakening the loser(s) of the contest with a stratum of the possible worlds set at each negotiation stage.

Acknowledgment The authors are grateful to anonymous reviewers for their valuable comments.

References

[1] Wiegers K E. Software Requirements, 2nd Ed. Portland: Microsoft Press, USA, 2003.

[2] Garcez A S, Russo A, Nuseibeh B, Kramer J. An analysis-revision cycle to evolve requirements specifications. In *Proc. the 16th IEEE Conference on Automated Software Engineering* (*ASE2001*), Coronado, USA, Nov. 26-29, 2001, pp.354-358.

[3] Garcez A S, Russo A, Nuseibeh B, Kramer J. Combining abductive reasoning and inductive learning to evolve requirements specifications. *IEE Proceedings of Software*, 2003, 150(1): 25-38.

[4] Kakas A C, Kowalski R, Toni F. The Role of Abduction in Logic Programming. Handbook of Logic in Artificial Intelligence and Logic Programming, Gabbay Dov M, Hogger C J, Robinson J A (eds.), Volume 5., 1998, pp.235-324.

[5] Mitchell T M. Machine Learning. New York: McGraw-Hill, 1997.

[6] Zowghi D, Offen R. A logical framework for modeling and reasoning about the evolution of requirements. In *Proc. the 3rd IEEE International Symposium on Requirements Engineering* (*RE 1997*), Jan. 5-8, 1997, Annapolis, USA, 1997, pp.247-257.

[7] Zowghi D. A requirements engineering process model based on defaults and revisions. In *Proc. the 11th International Workshop on Database and Expert Systems Applications* (*DEXA 2000*), Greenwich, UK, Sept. 6-8, 2000, pp.966-970.

[8] Alchourrón C, Gärdenfors P, Makinson D. On the logic of theory change: Partial meeting contraction and revision functions. *Journal of Symbolic Logic*, 1985, 50(2): 510-530.

[9] Darwiche A, Pearl J. On the logic of iterated belief revision. *Artificial Intelligence*, 1997, 89(1/2): 1-29.

[10] Delgrande J, Dubois D, Lang J. Iterated revision as priorited merging. In *Proc. the 10th International Conference on*

Principles of Knowledge Representation and Reasoning, Lake District, UK, Jun. 2-5, 2006, pp.210-220.
[11] Fermé E, Hansson S. Selective revision. *Studia Logica*, 1999, 63(3): 331-342.
[12] Booth R. A negotiation-style framework for non-prioritised revision. In *Proc. the 8th Conference on Theoretical Aspects of Rationality and Knowledge* (*TARK 2001*), Siena, Italy, Jul. 8-10, 2001, pp.137-150.
[13] Hansson S. A survey of non-prioritized belief revision. *Erkenntnis*, 1999, 50(2/3): 413-427.
[14] Hunter A, Nuseibeh B. Managing inconsistent specification: Reasoning, analysis, and action. *ACM Transactions on Software Engineering and Methodology*, 1998, 7(4): 335-367.
[15] Wiegers K. First things first: Prioritizing requirements. *Software Development*, 1999, 7(9): 48-53.
[16] Davis A. Just Enough Requirements Management: Where Software Development Meets Marking. New York: Dorset House Publishing, 2005.
[17] Karlsson J, Ryan K. A cost-value approach for prioritizing requirements. *IEEE Software*, 1997, 14 (5): 67-74.
[18] Pardee W J. To Satisfy and Delight Your Customer: How to Manage for Customer Value. New York: Dorset House Publishing, 1996.
[19] Zowghi D, Gervasi V. On the interplay between consistency, completeness, and correctness in requirements evolution. *Information and Software Technology*, 2003, 45(14): 993-1009.
[20] Gervasi V, Zowghi D. Reasoning about inconsistencies in natural language requirements. *ACM Transaction on Software Engineering and Methodologies*, 2005, 14(3): 277-330.
[21] Benferhat S, Cayrol C, Dobois D, Lang J, Prade H. Inconsistency management and prioritized syntax-based entailment. In *Proc. the 13th International Joint Conference on Artificial Intelligence*, Chambèry, France, Aug. 28-Sept. 3, 1993, pp.640-647.
[22] Lehmann D. Another perspective on default reasoning. *Annals of Mathematics and Artificial Intelligence*, 1995, 15(1): 61-82.
[23] Jin Y, Thielscher M. Iterated belief revision, revised. *Artificial Intelligence*, 2007, 171(1): 1-18.
[24] Mu K, Jin Z. Identifying acceptable common proposals for handling inconsistent software requirements. In *Proc. 27th IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems* (*FORTE2007*), Tallinn, Estonia, Jun. 27-29, 2007, pp.296-308.

Ke-Dian Mu received his B.Sc. degree in applied mathematics from Beijing Institute of Technology, China, in 1997, his M.Sc. degree in probability and mathematical statistics from Beijing Institute of Technology, China, in 2000, and his Ph.D. degree in applied mathematics from Peking University, Beijing, China, in 2003. From 2003 to 2005, he was a postdoctoral researcher at Institute of Computing Technology, Chinese Academy of Sciences, China. He is currently an associate professor at School of Mathematical Sciences, Peking University, Beijing, China. His research interests include uncertain reasoning in artificial intelligence, knowledge engineering and science, and requirements engineering.

Weiru Liu is a professor at the School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast. She received her B.Sc. and M.Sc. degrees in computer science from Jilin University, China, and her Ph.D. degree in artificial intelligence from the University of Edinburgh. Her main research interests include reasoning under uncertainty, uncertain and inconsistent knowledge and information fusion, belief and epistemic revision, and knowledge discovery in databases. She has published over 120 journal/conference papers in these areas.

Zhi Jin received the M.S. and Ph.D. degrees in computer science from Changsha Institute of Technology, China, in 1987 and 1992, respectively. She is currently a professor of computer science at Peking University, Beijing, China. Before joined Peking University, she has been a professor at the Academy of Mathematics and System Science at the Chinese Academy of Sciences since 2001. Her research interests include software requirements engineering and knowledge engineering. She has published a coauthored monograph by Kluwer Academic Publishers and more than 50 referred journal/conference papers in these areas. She has won various nation-class awards/honors in China, mainly including the Natural Science Foundation for Distinguished Young Scholars of China (2006), the Award for Distinguished Women IT Researcher of China (2004), and the Zhongchuang Software Talent Award (1997). She is the leader of more than 10 national competitive grants, including three China NSF grants, two China 973 program grants, and two China 863 program grants. She is a standing senior member of the China Computer Federation (CCF) and a grant review panelist for China NSF (Information Science Division). She is serving as an executive editor-in-chief for the Journal of Software, an editorial board member for the Expert Systems, and Chinese Journal of Computers; and served as a PC co-chair, area chair, or PC member for various conferences. She is a senior member of the IEEE and the IEEE Computer Society.

Jun Hong is a senior lecturer of computer science at Queen's University Belfast in the UK. He received his B.Sc. and M.Sc. degrees both in computer science and Ph.D. in artificial intelligence. His research is mainly in the area of artificial intelligence and its intersections with databases and Web technology, with a wide range of research interests including AI planning, data integration, Web data extraction and integration, Web mining, reasoning under uncertainty and intelligent tutoring systems. He has published around 60 research papers, most of which are in top-quality, peer-reviewed international journals, books, and international conference proceedings published by ICDE, AAAI, ECAI, IEEE, ACM, Springer, etc. He published some of the highly prestigious papers, cited by some of the best known researchers in AI and databases in the world. He has secured 11 research grants as either the principal investigator or investigator from the European Commission and other funding bodies. He is on the Steering Committee of the British National Conference on Databases (BNCOD), and co-chaired BNCOD 2006. In addition, he has been PC members of more than 30 international conferences, including WWW2010. He is on the editorial boards of two international journals. He has been regular reviewers of many top-quality international journals and research funding bodies, including IEEE Transactions, ACM Transactions, EPSRC, and the Netherlands Organisation for Scientific Research.

David Bell graduated in 1969 in Pure Mathematics, and has three research degrees in computing topics. He has been a full professor since 1986 at Queen's University Belfast since 2002, where he is now the director of research in knowledge and data engineering. He has produced several hundred publications, and supervised about 35 Ph.D.s to completion. He was prime investigator on about 18 EU-funded projects (eg MAP, ESPRIT, DELTA, COST, AIM, ...) and on many national projects in IT since 1981. His activities have included: PC chairmanships (e.g., joint programme committee chair of VLDB'93 and of ICDE'97), conference PC memberships and journal editing/editorial board memberships, e.g., for Computer Journal and North-Holland's Information Systems. He has been guest editor of well-known journals such as IEEE Trans. KDE, on (e.g.,) Knowledge Discovery, Data Driven Data Mining and Semantic Web. He is author/editor of several books. He served on the UK Technology Foresight Panel for several years, and was member of a number of national and international advisory and funding groups. His research interests are centred on data and knowledge management — the linking of reasoning under uncertainty, machine learning, and other artificial intelligence techniques with advances in database systems. This involves the exploration of other aspects of computing, including, at the present time, aspects of agent awareness and innateness.