International Journal of u-and e-Service, Science and Technology
Vol.7, No.4 (2014), pp.311-322
http://dx.doi.org/10.14257/ijunnesst.2014.7.4.28

# An Enhanced Negotiation-Style Framework for Requirements Change

Fengxu Wang

*Experimental Management Center of Liaocheng University*
*No. 1, Hunan Road, Dongchangfu District, Liaocheng City, Shandong, China 252059*
*wangfengxu@lcu.edu.cn*

***Abstract***

*There are many uncontrolled requirements changes existed in the software requirements engineering. Currently, under the premise of managing the software requirements changes using the logic-based technique, there is a framework that be proposed by Ke-Dian Mu etc based on Booth Negotiation-Style which managed the requirements changes conveniently and effectively. On the basis of the Mu's framework, this paper presents a selection strategy for selecting the negotiation scheme, which perfected the part that can be improved and managed the software requirements changes flexibly and effectively.*

***Keywords:*** *Negotiation-Style Framework, Requirement's Importance, Requirement's Magnitude, Balance Algorithm*

## 1. Introduction

The success of software system depends on how well it fits the needs of its users and its environment [1]. Software requirements comprise these needs, and requirements engineering (RE) is the process by which the requirements are determined and the requirements evolution are managed. The industry has a strong demand for sophisticated RE methods in order to manage the high complexity of requirements specifications for software–intensive embedded systems and ensure a high requirements quality [2, 3]. However, the inherent characteristics of software requirements, such as ambiguity, uncertainty, subjectivity and variably make RE's process too complex to gain the desired effect. Among these, requirements change is the key problem faced by RE. Requirements change throughout the entire software development life cycle. Many reasons, such as policy adjustments, market vitality, will lead the customers to adjust their actual requirements to gain the highest cost-effect. In general, appropriate requirements changes will not only enhance the quality of the software requirements specifications, but also contribute to a more perfect system. However, uncontrolled requirements changes may cause a series of modifications of all existing artifacts during the development, even many troubles of problems [4-6]. Then it's necessary to provide a flexible and effective management method to deal with the software requirements changes.

At present, the logic-based technology is widely recognized to manage the requirements changes [6]. Garcez AS *etc*. proposed that using the combination of the cycle reductive inference and the inductive learning to deduce the formation of requirements specification [13]. They believe that the development of requirements specifications must include revision and deduction and use a cycle two-stage model composed of two phases: analysis and

ISSN: 2005-4246 IJUNNEST

revision(Figure 1.1), the model supports revision and can remain the main requirements goal and nature.

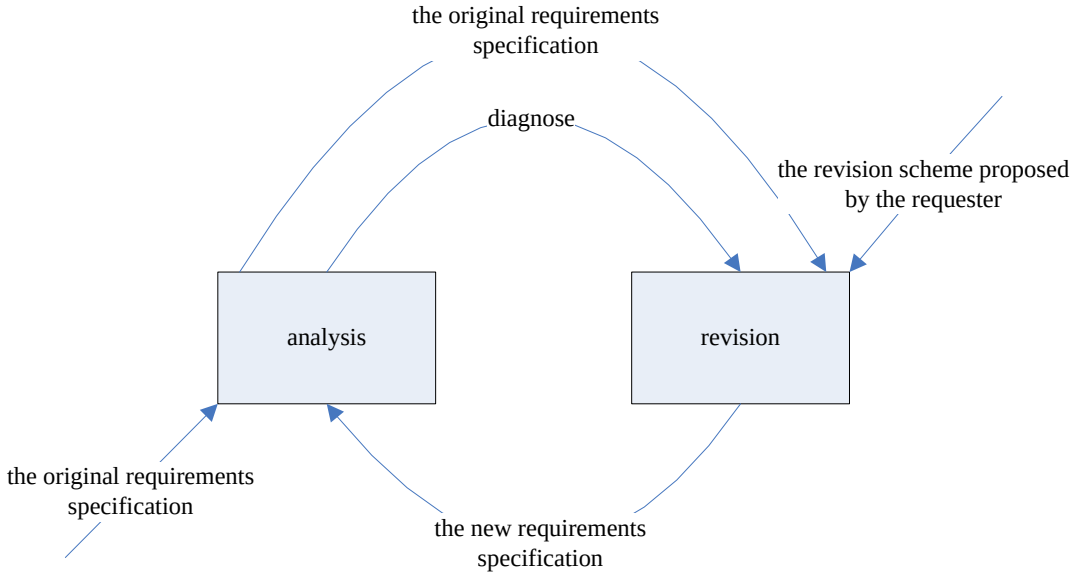


**Figure 1.1. The Cycle Deduction Process for the Requirements Specification**

Ke-Dian Mu, Zhi Jin *etc.*, proposed a series of work [14-18] to manage the changes of the software requirements. In [14] they think that the framework based on the Booth negotiating architecture[19] can be used to manage the changes of the requirements and eliminate the inconsistencies. The negotiation-style revision presented by Booth is achieved via a kind of negotiation between the existing information and new information. Mu etc. consider the current requirements specification and the request of requirements change as two negotiation parties in belief revision. They design three different belief negotiation models to execute the requirements changes:

1. The request is fully accepted and the current requirements specifications is modified. Conflicts of the two parties caused by policies and legislation changes, commercial strategies updating and marketplace changes, a defect found in proposed requirements or missing a requirement will require the stakeholders agreed to modify and extend existing requirements specifications.

2. The current requirements specification is fully accepted and request is refused. If request comes from some errors or worse communications, then the current requirements specifications should be retained.

3. The current requirements specification and the requested changes accommodate themselves to each other. In this situation, both sides need to make concessions, and communicate again to bring about the new requirements specifications.

Mu *etc.*, defined the detail negotiation rules and improved Booth's negotiation framework via Boolean algebra. But in Mu's framework they didn't tell how to choose a negotiation rule and they didn't describe the partition rules used to make a confession. We enhanced Mu's framework with a negotiation rule choosing algorithm. Our work made a more perfect negotiation process and improved the negotiation efficiency and accuracy. A case study made by us proved our method's feasibility and validity.

 

The rest of this paper is organized as follows. In Section 2 we introduce Mu's negotiation framework. Then in Section 3 we provide the enhanced negotiation framework strengthened with negotiation rule choosing algorithm. Section 4 gives a case study of the enhanced negotiation framework. Finally, we conclude this paper in Section 5.

## 2. MU'S Negotiation-Style Framework for Requirement Changes

Mu etc. improved Booth's negotiation-style framework for belief revision with a family of belief negotiation models appropriate for different processes of requirements revision. Firstly, they use classical first logic language without function symbols and existential quantities in representation of requirements specification. Then they use a three level prioritization scale to group requirements into several priority categories to increase the negotiation's granularity and reduce negotiation's workload. Truth value of all the atoms in the first logic formula was represented by a bit vector. These Vectors was partitioned into different group to form the extended set. Finally, the original set was expanded with the extended set to eliminate inconsistency among the requirements. Mu etc. defined the following three expanding models:

(1). The current set makes a concession, to extend the current requirements set $S$.

(2). The request set makes a concession, to extend the request requirements set $T$.

(3). Both of the current set and the request set make concessions, to extend the current requirements set $S$ and the request set $T$.

Let $W$ be a set of all possible worlds, Let $\Delta$ be a set of formulas and $<\Delta^1,...,\Delta^m>$ the priority-based partition of $\Delta$, a preorder relationship $\leq_\Delta$ on $W$ can be induced from $\Delta$ and get $<W_1^S,...,W_{n(\Delta)}^S>$, $W_1^{S_0}=[S_0]$, $W_1^T=[\Phi]$ []. Mu's negotiation framework can be summarized as follows:

(1). Initialize the negotiation values:

$S_0$=[$S_0$], $T_0$=[$T_0$] .

(2). $n$ is the minimum nonnegative number that meets $S_n \cap T_n \neq \varnothing$.

(3). $g(\partial i)=\{S_i,T_i\}$ *if* $S_i \cap T_i = \varnothing$.

$S_{i+1}=\nabla_{\partial i}(S_i)=S_i \cup W_{i+2}^{S_0}, T_{i+1}=\nabla_{\partial i}(T_i)=T_i \cup W_{i+2}^{T_0}$, *for all* $i<n$

$\partial=(<S_0,T_0>,\ldots,<S_n,T_n>)$, *where* $n=\min\{i|\ S_i \cap T_i \neq \varnothing\}$.

$\partial_i=(<S_0,T_0>,\ldots,<S_i,T_i>)$, *for all* $i<n$.

The recurrence relation of $S_{i+1}$ and $T_{i+1}$ can be executed selectively. Model 1 or model 2 corresponds to execute $S_{i+1}$ or $T_{i+1}$ separately. Model 3 corresponds to execute $S_{i+1}$ and $T_{i+1}$ at the same time.

In which, $\partial$ is a finite sequence, $\nabla_{\partial i}(S_i)$ represents the expanding operation of set $S_i$ under sequence $\partial$. Defined as under a functional relationship $g:\sum \to 2^{\aleph}$ which meets g0, there is ordered pair $N=\{g,\{\nabla_\partial\}_{\partial\in\sum}\}$, then to each $\partial=(<S_0,T_0>,\ldots,<S_n,T_n>)$, there is a functional relationship $\nabla_\partial:\{S_n,T_n\}\to\aleph$ under $\nabla_0$.

Now we give an example to illustrate Mu's negotiation-style change process.

**Example 1**. Consider $S =< \{a\},\{b\},\{c\} >, T =< \varnothing,\{-b\},\{c\} >$.Then truth value set of vector $(a,b,c)$ is:

$$W = \begin{Bmatrix} w_1 = 111\ w_2 = 110\ w_3 = 101 \\ w_4 = 100\ w_5 = 011\ w_6 = 010 \\ w_7 = 001\ w_8 = 000 \end{Bmatrix}$$

Partition of $W$ induced by $S_0$ s given as follows：

$$< \{w_1\}^{S_0}, \{w_2\}^{S_0}, \{w_3\}^{S_0}, \{w_4\}^{S_0}, \{w_5\}^{S_0}, \{w_6\}^{S_0}, \{w_7\}^{S_0}, \{w_8\}^{S_0} >,$$

Partition of induced by is given as follows：

$$< \{w_3, w_7\}^{T_0}, \{w_1, w_5\}^{T_0}, \{w_2, w_8\}^{T_0}, \{w_4, w_6\}^{T_0} >$$

As is shown in the known condition: $S_0 = [S_0] = \{w_1\}$, $T_0 = [T_0] = \{w_3, w_7\}$.

We find that: $S_0 \cap T_0 = \varnothing$.

There is an inconsistent in the system. According to model 3, Let and confess at the same time.

That is, $S_1 = S_0 \cup \{w_2\}^{S_0} = \{w_1, w_2\}, T_1 = T_0 \cup \{w_1, w_5\}^{T_0} = \{w_3, w_7, w_1, w_5\}$.

Obviously, agreement is reached, since $S_1 \cap T_1 = \{w_1\}$ .Then the revised requirements specification is $< \{a\},\{b\},\{c\} >$.

## 3. An Enhanced Negotiation-Style Framework

Mu etc. defined the detail negotiation rules and improved Booth's negotiation framework via Boolean algebra. But in Mu's framework they didn't tell how to choose a negotiation rule and they didn't describe the partition rules used to make a confession. In this paper we improved Mu's framework to be more operable and effective with a negotiation rule choosing algorithm.

The goal of requirements management is balancing the relationship between the system requirements and the customer needs. So system and customer's initiative should be considered and play an important role in choosing which negotiation model. Therefore, we proposed a model choosing algorithm based on attention degree of system and customer to the conflicting requirements and amount of the conflicting requirements. The process of model choosing algorithm is shown in Figure 2.

 

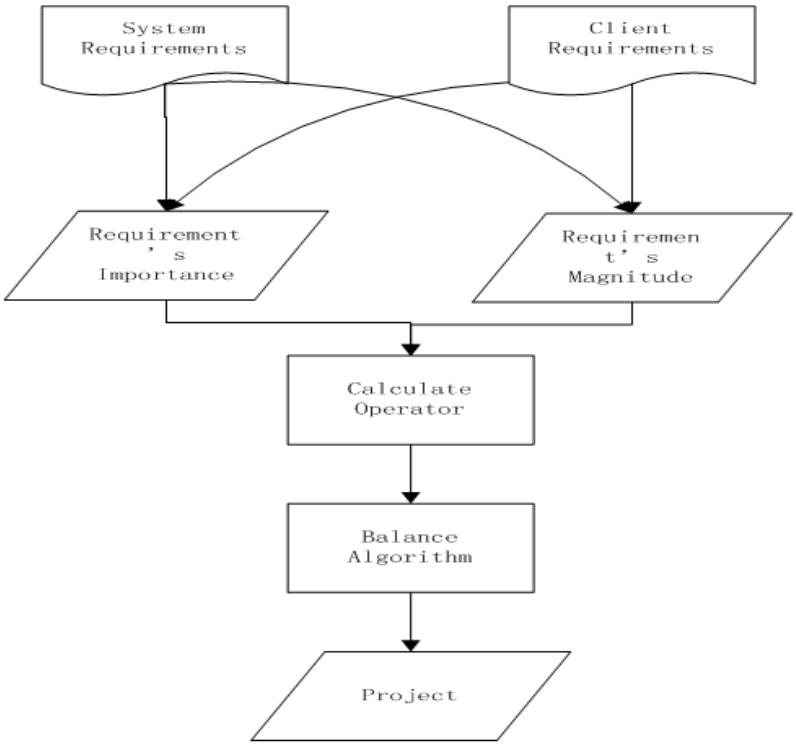


**Figure 2. Model Choosing Algorithm Process**

We use coefficient $\sigma_i$ to refer to attention degree of system to some requirement, $\sigma_i \in (0,1)$. The more $\sigma_i$ is closer to 1, means that the requirement is more important to the system. Similarly, coefficient $\tau_i$ means attention degree of customer to some requirements, $\tau_i \in (0,1)$ . The more $\tau_i$ is closer to 1, means that the requirement is more important to the customer.

The coefficient $\sigma_{\to\tau}$ is the global evaluation factor of the system to the customer requirements, $\sigma_{\to\tau} \in (0,1)$, if $\sigma_{\to\tau} \to 1$ ,the system thinks that the request of the customer for changing the requirements is more important to the whole situation. The coefficient $\tau_{\to\sigma}$ is the global evaluation factor of the customer to the system requirements, $\tau_{\to\sigma} \in (0,1)$, if $\tau_{\to\sigma} \to 1$, the customer thinks that the request of the system for changing the requirements is more important to the whole situation.

We suppose that m is the total number of system requirements, and n is the total number of customer requirements.

Then we can get the selection operator ב of the selection balance algorithm calculated as follows:

$$\hat{u}_{\sigma} = \frac{\Sigma_{i=1}^{m}\sigma_i}{m}, \hat{u}_{\tau} = \frac{\Sigma_{i=1}^{n}\tau_i}{n}.$$

$$\hat{u}_{\sigma\cup\tau} = (\sigma_{\to\tau} + \tau_{\to\sigma})\frac{\Sigma_{i=1}^{m}\sigma_i + \Sigma_{i=1}^{n}\tau_i}{m+n}$$

in which： $i \in (1,2,...), m \geq 0, n \geq 0.$

We can see from the above formula, if $\hat{u}_l \to 1, l \in (\sigma, \tau, \sigma + \tau)$ it means that the requirement that $l$ stands for is more important to the one who put forward the requirement. Therefore, the establishment of the selection algorithm BA is as follows:

$\hat{u}_{\sigma \cup \tau}$ can be seen as a global selection operator, if $\hat{u}_\sigma$ and $\hat{u}_\tau$ meet:

a. $\hat{u}_\sigma > \hat{u}_{\sigma \cup \tau}, \hat{u}_\tau < \hat{u}_{\sigma \cup \tau}$ , then to the attention degree under the whole situation, the system's request is higher than the customer's request, so the customer should make a concession, namely the negotiation model 1 in Mu's framework should be selected.

b. $\hat{u}_\sigma < \hat{u}_{\sigma \cup \tau}, \hat{u}_\tau > \hat{u}_{\sigma \cup \tau}$ , then to the attention degree under the whole situation, the customer's request is higher than the system's request, so the system should make a concession, namely the negotiation model 2 in Mu's framework should be selected.

c. $\hat{u}_\sigma \geq \hat{u}_{\sigma \cup \tau}, \hat{u}_\tau \geq \hat{u}_{\sigma \cup \tau}$ or $\hat{u}_\sigma \leq \hat{u}_{\sigma \cup \tau}, \hat{u}_\tau \leq \hat{u}_{\sigma \cup \tau}$ ,then to the attention degree under the whole situation, the customer's request has the same level with the system's request, so both of them should make concessions, namely the negotiation model 3 in Mu's framework should be selected.

Thus, we can make a more reasonable and correct selection scheme for the system requirements and customer requirements which can make the two sides to negotiate more fairly, eventually achieve the global balance.

Now we give an example to illustrate the negotiation framework enhanced with model choosing rules.

**Example 2**. Suppose that the system $S = <\{a\}, \{b\}, \{-c\}, \{d\}>$ , and the customer $T = <\{a\}, \{-b\}, \{c \vee d\}>$ .Then vector $(x, y, r, s)$ can be used to represent the set of the truth value domain.

$$W = \left\{ \begin{array}{l} w_1 = 1111 \quad w_2 = 1110 \quad w_3 = 1101 \quad w_4 = 1100 \\ w_5 = 1011 \quad w_6 = 1010 \quad w_7 = 1001 \quad w_8 = 1000 \\ w_9 = 0111 \quad w_{10} = 0110 \quad w_{11} = 0101 \quad w_{12} = 0100 \\ w_{13} = 0011 \quad w_{14} = 0010 \quad w_{15} = 0001 \quad w_{16} = 0000 \end{array} \right\}$$

The partition of $W$ induced by $T_0$ is:

$$<\{w_1, w_6, w_7\}^{T_0}, \{w_3, w_4, w_{11}, w_5, w_{12}\}^{T_0}, \{w_9, w_8, w_6, w_{10}, w_{13}\}^{T_0}, \{w_{14}, w_{15}, w_{16}\}^{T_0}>$$

Suppose that the attention degree coefficient of the system S to the requirements is:

$$\sigma = \{0.1, 0.6, 0.4, 0.1\},$$

The attention degree coefficient of the customer to the requirements is: $\tau = \{0.1, 02, 0.3\}$ ,

and $\sigma_{\to\tau} = 0.21, \tau_{\to\sigma} = 0.76.$

As it can been seen from the known conditions: $S_0 = [S_0] = \{w_3\}$, $T_0 = [T_0] = \{w_1, w_6, w_7\}.$

We find that $S_0 \cap T_0 = \varnothing.$

There exists inconsistency in the system. So we calculate the selection operator $\hat{u}$ :

$$\hat{u}_\sigma = \frac{\Sigma_{i=1}^4 \sigma_i}{4} = 0.30, \hat{u}_\tau = \frac{\Sigma_{i=1}^3 \tau_i}{3} = 0.20.$$

$$\hat{u}_{\sigma \cup \tau} = (\sigma_{\to\tau} + \tau_{\to\sigma}) \frac{3 * \hat{u}_\sigma + 4 * \hat{u}_\tau}{4+3} \approx 0.24.$$

 

Obviously,

$$\hat{u}_{\sigma} > \hat{u}_{\sigma\cup\tau}, \hat{u}_{\tau} < \hat{u}_{\sigma\cup\tau}.$$

So negotiation model 1 in Mu's framework should be selected. That is, $S$ remains the same and $T$ makes a concession.

Namely,

$$S_1 = S_0 = \{w_3\}, T_1 = \nabla^2_{\partial_0}(T_0) = T_0 \cup W_2^{T_0} = \{w_1, w_6, w_7, w_3, w_4, w_{11}, w_5, w_{12}\}.$$

Obviously, $S_1 \cap T_1 = \{w_3\}$, the negotiation succeeds.

Then requirements specification turns to be:

$$<\{a\},\{b\},\{-c\},\{d\}>$$

## 4. A Case Study

The Access Control System is widely used and known. In this section, the Access Control System will be used as an example to illustrate the feasibility and validness of the framework we proposed.

(1)The requirements specification of the Access Control System of an area which manages the parking is as follows:

a. The high priority requirements: The car is not allowed to enter the residential area without a specific permission. The car is allowed to enter the residential area with a specific permission. The situation when a car tries to enter the residential area without a specific permission will trigger the alarm.

b. The medium priority requirements: If the alarm is triggered, the owner of the car will not be able to press the button for entering the residential area again.

c. The low priority requirements: None.

(2) In the Access Control System of an area that manages the fire engines, three priority requirements should be included:

a. The high priority requirements: The fire engines are regarded as emergency vehicles. The emergency engine is allowed to enter the residential area without a specific permission. In addition to the emergency engines, the others are not allowed to enter the residential area without a specific permission.

b. The medium priority requirements: None.

c. The low priority requirements: None.

Then we first transform the requirements specification in (1) and (2) to the requirements specification in logic language. We can use the following symbols to denote the natural language.

●We use the predicate symbol to denote that is awarded a special license.

●We use the predicate symbol to denote that can enter the area.

●We use the predicate symbol to denote that if tries to enter the residential area, the alarm will be triggered.

●We use the predicate symbol to denote that pushes the button .

●We use the predicate symbol  to denote that  is the emergency engine.

●We use constant  to denote that the button for entering the area.

●We use constant  to denote the fire engines.

Then we can get the logic-based requirements set as follows:

(3) Three priority requirements are included in the management of the access:

$$S=\begin{cases} High:\begin{cases} -Aut(fire\_e)\rightarrow -Ent(fire\_e) \\ Aut(fire\_e)\rightarrow Ent(fire\_e) \\ -Aut(fire\_e)\rightarrow -Ala(fire\_e) \end{cases} \\ Medium: Ala(fire\_e)\rightarrow -Push(fire\_e,entr) \\ Low:\varnothing \end{cases}$$

(4) Three priority requirements are included in the management of the fire engines:

$$T=\begin{cases} High:\begin{cases} Eme(fire\_e) \\ Eme(fire\_e)\rightarrow Ent(fire\_e)\wedge -Aut(fire\_e) \\ -Aut(fire\_e)\wedge -Eme(fire\_e)\rightarrow -Ent(fire\_e) \end{cases} \\ Medium:\varnothing \\ Low:\varnothing \end{cases}$$

The presentation of the logic requirements is completed.

As the Access Control System of the area that manages the fire engines belongs to the Access Control System of the area that manages the access, we should regard the requirements set T as the customer that requests for changing requirements, and the requirements set S as the system that has the requirements specification.

Seen from the above formulation, $S\cap T=\varnothing$. So we use our framework to eliminate the inconsistency and get a perfect new system.

To simplify, we use letters to replace the requirements in the requirements set:

We use $a$ to denote the requirement $Aut(fire\_e)$, use $b$ to denote the requirement $Ent(fire\_e)$, use $c$ to denote the requirement $Ala(fire\_e)$, use $d$ to denote the requirement $Push(fire\_e,entr)$, and use $e$ to denote the requirement $Eme(fire\_e)$.

Then we can get the requirements sets:

$S=\{a\vee -l,-a\vee l,a\vee c,-c\vee -d\}$ and $T=\{e,-e\vee(-a\wedge l),a\vee e\vee -l\}$.

Then vector $(x,y,r,s,t)$ can be used to represent the set of the truth-value domain.

 

$$W=\left\{\begin{array}{llllll} w_1=11111 & w_2=11110 & w_3=11101 & w_4=11100 & w_5=11011 & w_6=11010 \\ w_7=11001 & w_8=11000 & w_9=10111 & w_{10}=10110 & w_{11}=10101 & w_{12}=10100 \\ w_{13}=10011 & w_{14}=10010 & w_{15}=10001 & w_{16}=10000 & w_{17}=01111 & w_{18}=01110 \\ w_{19}=01101 & w_{20}=01100 & w_{21}=01011 & w_{22}=01010 & w_{23}=01001 & w_{24}=01000 \\ w_{25}=00111 & w_{26}=00110 & w_{27}=00101 & w_{28}=00100 & w_{29}=00011 & w_{30}=00010 \\ w_{31}=00001 & w_{32}=00000 & & & & \end{array}\right\}$$

The division of that induced by $S_0$ is:

$$\begin{aligned} <W_1^{S_0}&=\{w_2,w_4,w_5,w_6,w_7,w_8\},\\ W_2^{S_0}&=\{w_1,w_3,w_{15},w_{26},w_{27},w_{28}\},\\ W_3^{S_0}&=\{w_{10},w_{12},w_{13},w_{14},w_{16},w_{18},w_{20},w_{25},w_{29},w_{30},w_{31},w_{32}\},\\ W_4^{S_0}&=\{w_9,w_{11},w_{17},w_{19},w_{21},w_{22},w_{23},w_{24}\}> \end{aligned}$$

The division of $W$ that induced by $T_0$ is:

$$\begin{aligned} <W_1^{T_0}&=\{w_{15},w_{18},w_{21},w_{22}\},\\ W_2^{T_0}&=\{w_{12},w_{13},w_{14},w_{16},w_{17}\},\\ W_3^{T_0}&=\{w_3,w_4,w_5,w_{10},w_{11},w_{19},w_{23},w_{24}\},\\ W_4^{T_0}&=\{w_1,w_2,w_6,w_{20},w_{25},w_{26},w_{27},w_{28},w_{29},w_{30},w_{31},w_{32}\},\\ W_5^{T_0}&=\{w_7,w_8,w_9\}> \end{aligned}$$

In which, the degree of attention coefficient to the requirements of the system $S$ : $\sigma=\{0.3,0.8,0.6,0.1\}$ , and the degree of attention coefficient to the requirements of the customer $T$ : $\tau=\{0.4,0.3,0.3\}$ ,and $\sigma_{\to\tau}=0.80,\tau_{\to\sigma}=0.66.$

Derived from the known conditions:

$S_0=[S_0]=\{w_2,w_4,w_5,w_6,w_7,w_8\}$, $T_0=[T_0]=\{w_{15},w_{18},w_{21},w_{22}\}$, because $S_0\cap T_0=\varnothing$ ,there exists inconsistency in the system. So we calculate the selection operator $\hat{u}$ :

$$\hat{u}_\sigma=\frac{\Sigma_{i=1}^4\sigma_i}{4}=0.45,\hat{u}_\tau=\frac{\Sigma_{i=1}^3\tau_i}{3}\approx 0.33.$$

$$\hat{u}_{\sigma\cup\tau}=(\sigma_{\to\tau}+\tau_{\to\sigma})\frac{3*\hat{u}_\sigma+4*\hat{u}_\tau}{4+3}\approx 0.56.$$

Obviously, $\hat{u}_\sigma\ll\hat{u}_{\sigma\cup\tau},\hat{u}_\tau<\hat{u}_{\sigma\cup\tau}$ the scheme 3 of negotiation in Ke-Dian Mu's framework should be selected. So, $S$ and $T$ both make concessions.

Namely,

$$S_1=S_0\cup W_2^{S_0}=\{w_2,w_4,w_5,w_6,w_7,w_8,w_1,w_3,w_{15},w_{26},w_{27},w_{28}\},$$

$$T_1=\nabla_{\partial_0}^2(T_0)=T_0\cup W_2^{T_0}=\{w_{15},w_{18},w_{21},w_{22},w_{12},w_{13},w_{14},w_{16},w_{17}\}.$$

Obviously, $S_1\cap T_1=\{w_{15}\}$ , the negotiation succeeds.

Then requirements specification turns to be:

$$<\{a \vee -b\},\{a \vee c\},\{-c \vee -d\},\{e\},\{a \vee e \vee -b\}>$$

Then the whole requirements specification can be described in natural language as follows:

a. The high priority requirements: The car is allowed to enter the residential area with a specific permission. The situation when a car tries to enter the residential area without a specific permission will trigger the alarm. The fire engines are regarded as emergency vehicles. The emergency engine is allowed to enter the residential area without a specific permission. In addition to the emergency engines, the others are not allowed to enter the residential area without a specific permission.

b. The medium priority requirements: If the alarm is triggered, the owner of the car will not be able to press the button for entering the residential area again.

c. The low priority requirements: None.

## 6. Conclusion

In the software development process, the management for the requirements changes is very critical [13]. The belief revision has been successfully applied in requirements management, AGM framework [11] thinks that the new information is always more reliable than the old, and the new information should be fully accepted. In Mu's framework [14], the negotiation is based on the priority, and the negotiation parties should predefine the priority levels for their requirements, but in some circumstances, the clients have the inaccurate definitions for the requirements priorities, or the conflicting requirements have the same priority level, Mu's framework didn't give schemes to these circumstances.

This paper introduced and analyzed various existing methods of managing the changes of requirements, and evaluated the typical management framework. An improved framework with model choosing algorithm based on Mu's negotiation framework was proposed. Our work improved the efficiency of Mu's negotiation framework and made negotiation process more reasonable. A case study for an access control system was conducted and proved our method's feasibility and validness. In the future studies, we will consider joining a reasonable method for dividing priority orders and better scheme for managing the requirements changes.

## References

[1] E. Sikora, B. Tenbergen and K. Pohl, "Requirements Engineering", vol. 17, **(2012)**, pp. 57-78.

[2] "Software development report by the Standish group", **(1995)**.

[3] S. Easterbrook, "Requirements engineering: social and technical issues", vol. 1, **(1994)**, pp. 41-65.

[4] J. Zhi, L. Lin and J. Ying, "Software Requirements Engineering: Principles and methods", Science Press, **(2008)**.

[5] M. Jarke, R. Klamma, K. Pohl and E. Sikora, "Requirements engineering in complex domains", in: Graph transformations and model-driven engineering, Springer, **(2010)**, pp. 602-620.

[6] S. Hussain, N. Ehsan and S. Nauman, "A strategic framework for requirements change in technical projects: Case study of a R\&D project", Computer Science and Information Technology, 2010 3rd IEEE International Conference on, IEEE, **(2010)**, pp. 354-358.

[7] A. K. Jallow, P. Demian, A. N. Baldwin and C. J. Anumba, "BPM-driven construction client requirements change management", Publisher, City, **(2010)**.

[8] R. Al-Khaldi, "Inconsistency Management in Software Functional Requirements: A Machine Learning System", Publisher, City, **(2012)**.

[9] D. Zowghi and R. Offen, "A logical framework for modeling and reasoning about the evolution of requirements, in: Requirements Engineering", 1997, Proceedings of the Third IEEE International Symposium on, IEEE, **(1997)**, pp. 247-257.

 

[10] D. Zowghi, "A requirements engineering process model based on defaults and revisions, in: Database and Expert Systems Applications", 2000, Proceedings 11th International Workshop on, IEEE, **(2000)**, pp. 966-970.

[11] C. Alchourron, P. G. Aardenfors and M. D, "On the logic of theory change: Partial meeting contraction and revision functions", Publisher, City, **(1985)**.

[12] A. d'Avila Garcez, A. Russo, B. Nuseibeh and J. Kramer, "An analysis-revision cycle to evolve requirements specifications, in: Automated Software Engineering", 2001, (ASE 2001), Proceedings, 16th Annual International Conference on, IEEE, **(2001)**, pp. 354-358.

[13] A. D. A. Garcez, A. Russo, B. Nuseibeh and J. Kramer, "Combining abductive reasoning and inductive learning to evolve requirements specifications", Publisher, City, **(2003)**.

[14] K. Mu, Z. Jin, R. Lua and Y. Peng, "Handling non-canonical software requirements based on annotated predicate calculus", Publisher, City, **(2007)**.

[15] K. Mu, Z. Jin and R. Lu, "Inconsistency-based strategy for clarifying vague software requirements, in: AI 2005: Advances in Artificial Intelligence", Springer, **(2005)**, pp. 39-48.

[16] K. Mu, Q. Zhang and Z. Jin, "Verifying software requirements based on answer set programming, in: Knowledge Science", Engineering and Management, Springer, **(2009)**, pp. 263-274.

[17] K. Mu, W. Liu, Z. Jin, R. Lu, A. Yue and D. Bell, "Handling inconsistency in distributed software requirements specifications based on prioritized merging", Publisher, City, **(2009)**.

[18] K. Mu, Z. Jin, R. Lu and W. Liu, "Measuring inconsistency in requirements specifications, in: Symbolic and Quantitative Approaches to Reasoning with Uncertainty", Springer, **(2005)**, pp. 440-451.

[19] R. Booth, "A negotiation-style framework for non-prioritised revision, in: Proceedings of the 8th conference on Theoretical aspects of rationality and knowledge", Morgan Kaufmann Publishers Inc., **(2001)**, pp. 137-150.

[20] K.-D. Mu, W. Liu, Z. Jin, J. Hong and D. Bell, "Managing software requirements changes based on negotiation-style revision", Publisher, City, **(2011)**.

## Authors

**Fengxu Wang**, male. He was born on October 11, 1978 in Liaocheng, Shandong, China. Now he works for Liaocheng University, and his research directions are computer lab management, software engineering etc.