International Journal of Hybrid Information Technology
Vol.6, No.6 (2013), pp.249-258
http://dx.doi.org/10.14257/ijhit.2013.6.6.22

An Improved Framework for Managing Requirements Changes

Jianxiu Bai[1], Ying Jin[2], Yirui Zhang[3] and Jing Zhang[4]

[1,2,3]*College of Computer Science and Technology, JiLin University*
No 2699 Qianjin Street, Changchun, Jilin Province, China
[1]*baijianxiu@126.com,* [2] *jinying@jlu.edu.cn,* [3] *zhangyr11@jlu.edu.cn*

[4] *Corresponding author: zhangjing99@jlu.edu.cn*

Abstract

The management of requirements changes is an important task in requirement engineering, and the request for changing requirements occurs after the establishment of the requirements specification should be handled more charily. The logic-based techniques have attracted great attention, Ke-Dian Mu and others adopted the negotiation-style revision proposed by Booth to manage the requirements changes effectively. Based on Mu's work, this paper improves the method of equivalence classes division, and proposes the "4 equivalence classes" division method, which improved the negotiation efficiency to some extent, perfected the original framework.

Keywords: *Requirements changes management, Negotiation-style revision,"4 equivalence classes" division method*

1. Introduction

Requirement engineering is the first phase in the software development process, and the requirements obtained in this phase relates to the quality of the final software products [1-3]. Appropriate changes of the requirements can improve the software products and enhance the satisfaction of the stakeholders, but once the changes occurs after the establishment of the requirements specification, the work in the other phases will also be affected [4]. An effective requirements management method will play an active role in the whole software development process.

In recent years, the logic-based techniques have been used in the requirements management continually. AGM framework thinks that the new information is always more reliable than the old, and the new information should be fully accepted [5]. Generally speaking, belief revision [6] is a process in which the original belief is modified when new evidence appears. The opposite one is called the non-priority belief revision [7-9], which doesn't think the new information can always replace the old information. R. Booth proposed a negotiation-style framework which based on the non-priority belief revision [7]. Ke-Dian Mu *et al.*, refined Booth framework and applied it into the requirements management field [10].

Booth framework is an abstract structure, whose functions and definitions should be given specific interpretations when facing specific knowledge domain. Mu etc. defined the forms of the current requirements specification and the request of requirements change, and also defined an ordering method by which all the possible propositional worlds can be divided into some equivalence classes. They also proposed three negotiation models, respectively are: (1) fully accept the change request, (2) fully retain the current requirements, (3) both sides make

ISSN: 1738-9968 IJHIT

concessions. The priority levels of the requirements are the basis of the negotiation, the original requirements specification is seen as the current set of beliefs, and the request for changing requirements as the new evidence, and the goal is to reach an agreement between the two parties.

In the framework of Booth and Mu, the requirements priority levels are necessary, which need the predefinition of the negotiation parties. The rationality and standard of the priority setting will affect the negotiation result, especially when the conflicting requirements have the same priority level or the requirements are vaguely defined, this method loses advantage. Based on Mu's work, this paper proposes a negotiation method in which the requirements priority levels are ignored. This method redefines the forms of the current requirements specification and the request of requirements changes, improves the method of equivalence classes division, proposes the "4 equivalence classes" division method that divides all the possible propositional worlds into at most 4 classes, thus the negotiation can be finished in finite rounds, this method accelerates the negotiation, perfects the original framework to some extent.

The rest of this paper is organized as follows. Section 2 introduces Mu's work briefly. Section 3 proposes an improved scheme for the equivalence classes division. Section 4 gives a case to illustrate the improved framework. Section 5 gives the comparison with the related work and summarizes our work.

2. Mu's Framework for Requirements Changes Management

In the stage of the logical expression of the requirements, Mu etc. translate different forms of requirements into the first-order logic expressions, and define the requirements collection Δ, and each formula $\alpha \in \Delta$ represents a requirement state. The requirements set is expressed as $<\Delta^1, \Delta^2, \Delta^3>$, which has the priority "*High*", "*Medium*" and "*Low*" orderly [11]. If a fact α and its negation $\neg\alpha$ can be derived meanwhile, there is an inconsistency.

W is used to denote the set of all the possible propositional worlds and they defined a pre-order relationship $\leq_\Delta$ on W, $<\Delta^1, \Delta^2, \Delta^3>$ is the requirements set, for each $i(1 \leq i \leq 3)$, $k^i(w) = \{\varphi \in \Delta^i : w \models \varphi\}$ and $w \in W$, define $w \leq_\Delta w'$ iff 1) $|k^i(w)| = |k^i(w')|$ *for all* i or 2) exists i that meets $|k^i(w)| > |k^i(w')|$ and for all $j < i$, $|k^i(w)| = |k^i(w')|$. $\leq_\Delta$ is based on the lexicographical relation, by which W is divided into $<W_1, ..., W_{n(\Delta)}>$, so as to each w and w' which meet $|k(w)| = |k(w')|$ are in the same partition. S and T are the two negotiation parties, the final goal of the negotiation is to get intersection between the two sides by expanding $S(T)$ or both of them continually. Here σ is used to denote the different stages of the negotiation, and $\sigma_i = (<S_0, T_0>, ..., <S_i, T_i>)$, a function g is defined, $g(\sigma_i) = \{S_i, T_i\}$, namely, when the negotiation is at stage σ_i, there is at least one party making concession between S_i and T_i, ∇_σ is concession function which provides the specific concession rules:

In brief, the negotiation-style belief revision can be seen as a process that consists of the function g and the concession operation ∇_σ, and it can be summarized as the following steps:

(1) Initialization, set $S_0 = S, T_0 = T$.

(2) If $S_0 \cap T_0 \neq \varnothing$, take $Merge(S,T) = S_0 \cap T_0$, else go on negotiating until there is intersection between S and T, where n is minimal that meets $S_n \cap T_n \neq \varnothing$.

(3) For each $i \in [0, n)$,

$$S_{i+1}=\begin{cases}\nabla_{\sigma_i}(S_i), if\ S_i \in g(\sigma_i)\\ S_i \quad\quad ,otherwise\end{cases},\quad T_{i+1}=\begin{cases}\nabla_{\sigma_i}(T_i), if\ T_i \in g(\sigma_i)\\ T_i \quad\quad ,otherwise\end{cases}$$

Where $\sigma_i=(<S_0,T_0>,...,<S_i,T_i>)$.

(4) The specific concession rules:

$$\nabla_{\sigma_i}(S_i)=\begin{cases}S_{\mathrm{i}} \quad\quad , model(2)\\ S_{\mathrm{i}} \cup W_{i+2}^{S_0}, otherwise\end{cases},\quad \nabla_{\sigma_i}(T_i)=\begin{cases}T_{\mathrm{i}} \quad\quad , model(1)\\ T_{\mathrm{i}} \cup W_{i+2}^{T_0}, otherwise\end{cases}$$

(5) The final negotiation result is: $Merge(S,T)=S_{\mathrm{n}} \cap T_{\mathrm{n}}$,in which S_n is the extending of S , and T_n is the extending of T .

Here given a case that uses model (3) to handle the inconsistency caused by the requirements changes.

Example .Consider $S=<\{a\},\{b\},\{\neg c\}>, T=<\{c\},\varnothing,\{\neg b\}>$, the forms of them correspond to $<\Delta^1,\Delta^2,\Delta^3>$, we use a bit vector consisting of truth values of (a,b,c) to denote each possible world:

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

$S_0=S, T_0=T$, then we can get the stratification of W based on S_0 :

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

The stratification of W based on T_0 :

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

$S_0=[S_0]=\{w_1\}, T=T_0=\{w_3,w_7\}$, as $S_0 \cap T_0 \neq \varnothing$, there exists inconsistency, the model (3) is selected to negotiate, the two sides both make concession, and we get:

$S_1=S_0 \cup W_2^{S_0}=\{w_1,w_2\},\ T_1=T_0 \cup W_2^{T_0}=\{w_3,w_7,w_1,w_5\}$

Obviously, $S_1 \cap T_1=\{w_1\}$, the negotiation finishes, and $Merge(S,T)=\{w_1\}$,then the modified requirements specification is: $<\{a\},\{b\},\{c\}>$.

3. An Improved Equivalence Classes Division Method

Mu *et al.*, refined the Booth framework based on the knowledge of the requirements domain. In which, the stratification of W (equivalence classes division) plays an important role to the final negotiation result. S and T both defined priority levels for their requirements, so the equivalence classes division is also based on the priority ,and generally speaking, the quantity of the equivalence classes is uncertain.

On the basis of Mu's framework, this paper improves equivalence classes division method, redefines the forms of the current requirements specification and the request of requirements changes, and S and T are seen as completely equal negotiating parties, the priorities of the requirements are ignored in the process of negotiations. "4 equivalence classes" division method is defined, and to any negotiation, W will be divided in to no more than 4 stratifications. S and T negotiate based on the division method until they reach an agreement. The quantity of the equivalence classes is finite, thus it will have less negotiation rounds that accelerate the negotiation, and finally get the negotiation result satisfied by both sides in finite steps. The overall framework is as follows:

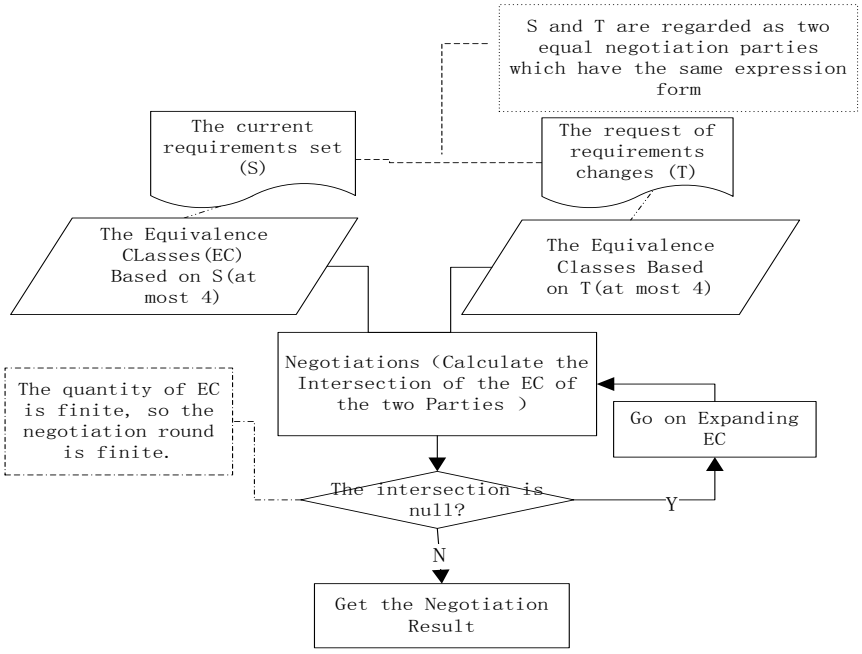


Figure 1. The Advanced Framework

Firstly, this paper defines the forms of the current requirements specification and the request of requirements changes, still uses the first-order logic to express the requirements, S represents the current requirements specification and T represents the request of requirements changes, where S and T have the same interpretation, they are completely equal. As the negotiating parties give priorities to the requirements from their own point of view, in the process of negotiation, the priorities are ignored, in short, S and T negotiate completely equally under the conditions of ignoring the priorities so as to get the negotiation result quickly and improve the negotiation efficiency.

Next, dividing the equivalence classes for W, the "4 equivalence classes" division method is based on an "extending operation", and the definition of "extending operation" is as follows:

*D*efine 1.(extending operation) To one fact, the extending operation is a process that gets the negation-value of this fact, and to several facts, the extending operation is a process that gets all the possible values except for the truth-value of all the facts and the negation-value of all the facts.

According to the extending operation, the requirements set $S(or\ T)$ can be divided into 4 equivalence classes $< W_1, W_2, W_3, W_4 >$, and the specific division method is as follows:

$W_1 = \{the\ vector\ representation\ of\ the\ truth\ value\ of\ all\ the\ formulas\}$;

$W_2 = \{retain\ all\ the\ formulas\ just\ like\ \alpha\ and\ the\ first\ fact\ of\ which\ just\ like\ \alpha,$
$and\ extend\ all\ the\ formulas\ just\ like\ \neg\beta\ and\ the\ first\ fact\ of\ which\ just\ like\ \neg\beta\}$;

$W_3 = \{extend\ all\ the\ formulas\ just\ like\ \alpha\ and\ the\ first\ fact\ of\ which\ just\ like\ \alpha,$
$and\ retain\ all\ the\ formulas\ just\ like\ \neg\beta\ and\ the\ first\ fact\ of\ which\ just\ like\ \neg\beta\}$;

$W_4 = \{all\ the\ other\ vector\ representations\}$.

Here are some notices, when facing some specific situations, the requirements set can also be divided into three or two stratifications. For example, if all the formulas and all the first facts of the formulas have the form of α, or all the formulas and all the first facts of the formulas have the form of $\neg\beta$,then the requirements set S can be divided into the following two stratifications:

$W_1 = \{the\ vector\ representation\ of\ the\ truth\ value\ of\ all\ the\ formulas\ in\ S\}$;

$W_2 = \{all\ the\ other\ vector\ representations\}$.

Then an example will be used to illustrate the application of the division method in the negotiation framework:

Consider $S = <\{a\},\{\neg b\},\{\neg c\}>, T = <\{a\},\{b\},\{\neg c\}>$, though the requirements in $S\ and\ T$ have priorities, they are ignored here, and we use a bit vector consisting of truth values of (a,b,c) to denote each possible world:

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

Model (3) is selected, and can get the stratification of W based on S:

$W_1^S = \{w_4\}, W_2^S = \{w_2, w_3\}, W_3^S = \{w_8\}, W_4^S = \{w_1, w_5, w_6, w_7\}$,

The stratification of W based on T:

$W_1^T = \{w_2\}, W_2^T = \{w_1\}, W_3^T = \{w_4, w_6\}, W_4^T = \{w_3, w_5, w_7, w_8\}$,

$S_0 = S = \{w_4\}, T_0 = T = \{w_2\}, S_0 \cap T_0 \neq \varnothing$, so there exists inconsistency, then start the negotiation, $S_1 = S_0 \cup W_2^S = \{w_2, w_3, w_4\}$, $T_1 = T_0 \cup W_2^T = \{w_1, w_2\}$,and $S_1 \cap T_1 = \{w_2\}$,the negotiation finishes, and $Merge(S,T) = \{w_2\}$.

Then the modified requirements specification is: $<\{a\},\{b\},\{\neg c\}>$.This is the result that both parties make concessions, namely, they reach an agreement in finite steps.

4. An Example Study

This paper still uses the case that Mu used in his paper [10] to illustrate the division method and prove the correctness and efficiency of the method.

Most of the requirements specifications are described in natural language [12], in a residential area, the requirements of the access control system in natural language can be described as follows:

(1) The requirements with the priority of "$High$":

The vehicles without permission cannot be allowed to enter this area;

The vehicles with permission can be allowed to enter this area;

The condition that a vehicle without permission tries to enter the area will trigger the warning alarm.

(2) The requirements with the priority of "$Medium$":

If the warning alarm is triggered, the vehicle cannot push the entrance button again.

In order to translate the above requirements into first-order logical expressions, the following predicates and constant are introduced:

We use $Per(x)$ to denote that x is a vehicle with permission, $Ent(x)$ to denote that x can enter the area, $Tri(x)$ to denote that when x tries to enter the area the warning alarm is triggered, $Push(x, y)$ to denote that x pushes the button y, $entr$ to denote the entrance button.

Corresponding, the logic expressions of the requirements can be described as follows:

$$S_{High}=\{(\forall x)(\neg Per(x)\rightarrow\neg Ent(x)),(\forall x)(Per(x)\rightarrow Ent(x)),(\forall x)(Per(x)\rightarrow Tri(x))\}$$

$$S_{Medium}=\{(\forall x)(Tri(x)\rightarrow\neg Push(x,entr))\}$$

$$S_{Low}=\varnothing$$

In the process of making requirements, the emergency managing system asks the access control system to take the emergency into consideration, for example the fire engine should enter the area successfully, so we get the request of requirements changes:

The requirements with the priority of "$High$":

(T1) The fire engine should be seen as the emergency vehicle.

(T2) The emergency vehicles without permission can be allowed to enter this area;

(T3) Except for the emergency vehicles, the vehicles without permission cannot be allowed to enter this area;

In order to translate the above requirements into first-order logical expressions, the following predicate and constant are introduced again: We use $fire\_e$ to denote that x is a fire engine, $Emer(x)$ to denote that x is an emergency vehicle. When checking the consistency of requirements, only the ground formulas should be considered, so the original requirements are expressed in the following forms:

$$S^1_{High}=\{(\neg Per(fire\_e)\rightarrow\neg Ent(fire\_e)),(Per(fire\_e)\rightarrow Ent(fire\_e)),(Per(fire\_e)\rightarrow Tri(fire\_e))\}$$

$$S^1_{Medium}=\{(Tri(fire\_e)\rightarrow\neg Push(fire\_e,entr))\}$$

$$S^1_{Low}=\varnothing$$

Corresponding, the logic expressions of the changing request can be described as follows:

$$T_{High}=\{Emer(fire\_e),Emer(fire\_e)\rightarrow Ent(fire\_e)\wedge -Per(fire\_e),-Per(fire\_e)\wedge -Emer(fire\_e)\rightarrow -Ent(fire\_e)\}$$

$$T_{Medium}=\varnothing$$

$$T_{Low}=\varnothing$$

$S^1_{High}\cup\neg Ent(fire\_e)$, $T_{High}\cup\neg Ent(fire\_e)$, so there exists inconsistency, as the emergency is necessary to be considered, the request should be fully accepted, model (1) is selected. As the request only asks to change the "$High$" level requirements, we should retain the "$Medium$" level requirements, and set:

$$S=\{(\neg Per(fire\_e)\rightarrow\neg Ent(fire\_e)),(Per(fire\_e)\rightarrow Ent(fire\_e)),(Per(fire\_e)\rightarrow Tri(fire\_e))\},$$

$T=\{Emer(fire\_e), Emer(fire\_e)\rightarrow Ent(fire\_e)\wedge -Per(fire\_e), -Per(fire\_e)\wedge -Emer(fire\_e)\rightarrow -Ent(fire\_e)\}$

And we use a bit vector consisting of truth values of $(Per(fire\_e), Ent(fire\_e), Tri(fire\_e), Emer(fire\_e), Push(fire\_e, entr))$ to denote each possible world:

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

The stratification of W based on S:

$W_1^S=\{w_2, w_4, w_5, w_6, w_7, w_8, w_{26}, w_{28}\}, W_2^S=\{w_{10}, w_{12}, w_{13}, w_{14}, w_{15}, w_{16}\},$

$W_3^S=\{w_{18}, w_{20}, w_{29}, w_{30}, w_{31}, w_{32}\},$

$W_4^S=\{w_1, w_3, w_9, w_{11}, w_{17}, w_{19}, w_{21}, w_{22}, w_{23}, w_{24}, w_{25}, w_{27}\}, T=\{w_{17}, w_{18}, w_{21}, w_{22}\}$, then we start the negotiation based on model (1):

$S_0=W_1^S=\{w_2, w_4, w_5, w_6, w_7, w_8, w_{26}, w_{28}\}, T_0=T=\{w_{17}, w_{18}, w_{21}, w_{22}\}$, obviously, $S_0\cap T_0=\varnothing$, $S_1=S_0\cup W_2^S=\{w_2, w_4, w_5, w_6, w_7, w_8, w_{10}, w_{12}, w_{13}, w_{14}, w_{15}, w_{16}, w_{26}, w_{28}\}$, $T_1=T_0$, $S_1\cap T_1=\varnothing$,then go on the negotiation, $T_2=T_1=\{w_{17}, w_{18}, w_{21}, w_{22}\}$, $S_2=S_1\cup W_3^S=\{w_2, w_4, w_5, w_6, w_7, w_8, w_{10}, w_{12}, w_{13}, w_{14}, w_{15}, w_{16}, w_{18}, w_{20}, w_{26}, w_{28}, w_{29}, w_{30}, w_{31}, w_{32}\}$.

$S_2\cap T_2=\varnothing=\{w_{18}\}$, the negotiation finishes, then $Merge(S,T)=\{w_{18}\}$. Then the modified requirements specification is:

$$S_{High}^2=\left\{\begin{array}{l}(Per(fire\_e)\rightarrow Ent(fire\_e)), (\neg Per(fire\_e)\rightarrow Tri(fire\_e)), Emer(fire\_e), \\ Emer(fire\_e)\rightarrow Ent(fire\_e)\wedge\neg Per(fire\_e), \neg Per(fire\_e)\wedge\neg Emer(fire\_e)\rightarrow\neg Ent(fire\_e)\end{array}\right\}$$

$$S_{Medium}^2=\{(Tri(fire\_e)\rightarrow\neg Push(fire\_e, entr))\}$$

$$S_{Low}^2=\varnothing$$

In actual circumstances, the entrance of the fire engine should not trigger the warning alarm, and the requirement $(\neg Per(fire\_e)\wedge\neg Emer(fire\_e)\rightarrow Tri(fire\_e))$ is suggested to replace the original requirement $(\neg Per(fire\_e)\rightarrow Tri(fire\_e))$, after verification, the replace operation can keep the consistency well, so the request is fully accepted, and the final requirements can be described in natural language as follows:

(1) The requirements with the priority of "$High$":

The vehicles with permission can be allowed to enter this area;

The fire engine should be seen as the emergency vehicle.

The emergency vehicles without permission can be allowed to enter this area;

Except for the emergency vehicles, the vehicles without permission cannot be allowed to enter this area;

Except for the emergency vehicles, the condition that a vehicle without permission tries to enter the area will trigger the warning alarm.

(2) The requirements with the priority of "*Medium*":

If the warning alarm is triggered, the vehicle cannot push the entrance button again.

5. Comparison and 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 [5] thinks that the new information is always more reliable than the old, and the new information should be fully accepted. In Mu's framework [10], 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.

Based on Mu's work, this paper improved and complemented the work of dividing stratifications for W, redefined the forms of the current requirements specification and the request of requirements changes, S and T are seen as completely equal negotiating parties, the priorities of the requirements in S and T are ignored, and we defined "4 equivalence classes" division method, whose division result is the foundation to the negotiation. Our division method is very efficient when facing the situation that the two negotiators having uncertain definitions for the requirements priorities or when the conflicting requirements have the same priority level. As the quantity of the equivalence classes is finite, the negotiation round is finite, it accelerates the negotiation and improves the efficiency of negotiation, it perfects the original framework to some extent. Finally, a case of managing the requirements changes for an access control system is used to illustrate the correctness and efficiency of the method proposed in this paper.

Acknowledgements

Project supported by the Program for New Century Excellent Talents in University (No.NECT-10-0436).

References

[1] E. Knauss and S. Meyer, "Experience-based requirements engineering tools", managing requirements knowledge, Springer Berlin Heidelberg, (**2013)**, pp. 333-351.
[2] Z. Jin, L. Liu and Y. Jin, "Software Requirements Engineering: Principles and Methods", Beijing: Science Press, (**2008)**.
[3] I. Sommerville and G. Kotonya, "Requirements engineering: processes and techniques", John Wiley & Sons, Inc., (**1998)**.
[4] K. E. Wiegers, "Software Requirements", 2nd Ed. Portland: Microsoft Press, USA, (**2003)**.
[5] D. Lehmann, "Belief revision, revised", Proceedings of the 14th international joint conference on Artificial intelligence, vol. 2, Morgan Kaufmann Publishers Inc., (**1995)**.

[6] D. Zowghi, "A requirements engineering process model based on defaults and revisions", In Proc. the 11th International Workshop on Database and Expert Systems Applications (DEXA2000), Greenwich, UK, (**2000**) September 6-8, pp. 966-970.
[7] R. Booth, "A negotiation-style framework for non-prioritised revision", In Proc. the 8th Conference on Theoretical Aspects of Rationality and Knowledge (TARK2001), Siena, Italy, **(2001)** July 8-10, pp. 137-150.
[8] S. Hansson, "A survey of non-prioritized belief revision", Erkenntnis, vol. 50, no. 2-3, (**1999)**, pp. 413-427.
[9] R. Booth, "On the logic of iterated non-prioritised revision", Conditionals, Information, and Inference, Springer Berlin Heidelberg, **(2005)**, pp. 86-107.
[10] K. D. Mu, W. Liu, Z. Jin, *et al.*, "Managing software requirements changes based on negotiation-style revision", Journal of Computer Science and Technology, vol. 26, no. 5, (**2011**), pp. 890-907, pp. 890-907.
[11] K. Wiegers, "First things first: Prioritizing requirements", Software Development, vol. 7, no. 9, (**1999**), pp. 48-53.
[12] V. Gervasi and D. Zowghi, "Reasoning about inconsistencies in natural language requirements", ACM Transaction on Software Engineering and Methodologies, vol. 14, no. 3, (**2005)**, pp. 277-330.
[13] M. Aiello, P. Bulanov and H. Groefsema, "Requirements and tools for variability management", Computer Software and Applications Conference Workshops (COMPSACW), 2010 IEEE 34th Annual, IEEE, (**2010)**.

Authors

Jianxiu Bai

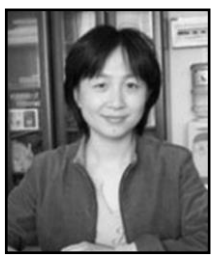
She was born in 1989, received her B.Sc. in Computer Science and Technology (2012) from JiLin University. Her main research interests include formal methods and software requirements engineering.

Ying Jin

She was born in 1971, professor, Ph.D. supervisor. Her main research interests include software requirements engineering, formal methods, programming language and its implementation.

Yirui Zhang

He received his B.Sc. in Computer Science and Technology (2011) from JiLin University. His main research interests include software requirements engineering and formal methods.



Jing Zhang

She was born in 1975, lecturer. Her main research interests include software requirements engineering, formal methods, programming language and its implementation.