ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)
DOI: 10.17559/TV-20150731092521

A COMPROMISE-NEGOTIATION FRAMEWORK BASED ON GAME THEORY FOR ELIMINATING REQUIREMENTS INCONSISTENCY

Tie Feng, Yi-rui Zhang, Ying Jin, Jing Zhang

Original scientific paper

For any proposed software development project, it is inevitable to confront requirements changes during the software development life cycle. Uncontrolled changes may cause bad requirements specification, which even further leads to project's failure. As a result, it is necessary to provide effective and flexible requirements change management. One of the kernel tasks of requirement change management is to eliminate requirement inconsistency caused by changes. In this paper, we consider negotiation process of the new and the old requirements specifications under Game theory. Both sides concession during the game process until the game achieves Nash equilibrium, i.e. both sides gain satisfied negotiation results. To be specific, firstly, the requirement set is represented in logical formula. Then a requirement conflict negotiation method based on mixed strategy and pure strategy Nash equilibrium is proposed separately, and a Compromise-Negotiation framework based on Game theory is presented as well. A case study will be given in the following part to verify our method's effectiveness. Finally, it comes to a comparison and conclusion.

Keywords: *compromise-negotiation; Game theory; Nash equilibrium; requirements inconsistency; software engineering*

Okvirno načelo za postizanje kompromisa u otklanjanju nedosljednosti u zahtjevima utemeljenim na Teoriji igara

Izvorni znanstveni članak

Za bilo koji predloženi projekt razvoja softvera, neizbježno je suočiti se s promjenama zahtjeva tijekom razvoja softvera. Nekontrolirane promjene mogu dovesti do loše specifikacije zahtjeva, što čak može rezultirati propadanjem projekta. Rezultat toga je potreba za osiguranjem učinkovitog i fleksibilnog upravljanja promjenama zahtjeva i jedan od osnovnih zadataka je uklanjanje nedosljednosti u zahtjevima izazvanim promjenama. U ovom radu razmatramo postupak pregovora o specifikaciji novih i starih zahtjeva primjenom Teorije igara. Obje strane čine ustupke tijekom igre dok se ne postigne Nashovaravnoteža, tj. dok obje strane ne budu zadovoljne rezultatom pregovora. Točnije, najprije se niz zahtjeva predstavlja logičnom formulom. Tada se odvojeno predlaže metoda pregovora oko zahtjeva, utemeljena na miješanoj strategiji i čistoj strategiji Nashoveravnoteže, a također i okvirno načelo za postizanje kompromisa na temelju Teorije igara. Daje se i analiza slučaja u svrhu provjere učinkovitosti naše metode. Na kraju se daju usporedbe i zaključak.

Ključne riječi: *Nashova ravnoteža; nedosljednost u zahtjevima; pregovori za postizanje kompromisa; razvoj softvera; Teorija igara*

1 Introduction

For any proposed software development project, its success depends largely on the quality of the requirements specification since it is the basis for the subsequent development activities [1]. Uncontrolled requirement changes may result in many serious problems during the development, especially when the software requirements specification is established [2]. These changes need a more flexible and effective handling model [3].

Zowghi et al. summarizes the problem of changing requirements as the belief revision [4, 5] process. Alchourron, Gardenfors, Makinson et al. raise a widely used AGM framework [6], in which the new changing request always has higher belief values, that is, when there is inconsistency between the old requirement and the new request, the system always fully accepts the new request. Garcez A. S. et al. propose to use the combination of the cycle reductive inference and the inductive learning to deduce the formation of requirements specifications [7, 8]. In contrast, Booth proposes a non-priority belief revision method [9]. Ke-Dian Mu et al. improve Booth's method by defining the specific negotiation function [10÷13], making the process of negotiation more objective. Yirui Zhang et al. propose a compromise-based negotiation framework to manage the requirements changes [14].

Game theory is a mathematical theory and method with a nature of struggle and competition which studies the phenomenon [15]. Decision Theory [16] is a branch of Game theory. In this paper, we apply Game theory to the problem of changing requirements. The inconsistency caused by the changes can be regarded as the interest conflict between the requester of changing requirements and the maker of the original requirements. Both of the two sides expect that 1) their requirements set can be put into the final requirements specification, and 2) that no inconsistencies are caused by the requirement changes. Thus, Game theory is applied here to make a negotiation both satisfy the requester and the maker, and eventually managing requirement changes.

This paper presents a Compromise-Negotiation framework based on Game theory to eliminate requirements inconsistency, aiming to solve the inconsistency problem of the requirements set caused by change. Section 2 defines the logic representation of the requirements specification with a brief introduction of Game Theory. Section 3 proposes the Compromise-Negotiation based on Game Theory. Section 4 solves a software engineering case. Section 5 gives a comparison to other relative works. Section 6 would give the conclusion.

2 Preliminaries

In software engineering, logic-based representation is widely used due to its powerful reasoning ability [3], in which requirements are described as facts of certain schemes. Let L_{Φ_0} be the logical language consisting of atoms Φ_0 and logical connectors $\{\vee,\wedge,\neg,\rightarrow\}$ etc., and operator $\vdash$ means inference relation in the classical logic.

In this paper, we take the definition of inconsistency as follows: logically speaking, if both the fact *a* and its

negation $\neg a$ exist in the requirements set S, it is safe to say that there is inconsistency in S. Namely, $a \wedge \neg a$ is an inconsistency, which we use $\perp$ to represent.

Below are several crucial definitions in this paper:

Definition 1.

If there is a formula set R, then the atom set $G(R)$ is: $G(R) = \{g \mid \exists u \in R\}$, which satisfy $g \subset u$, and the explanation set $A(R)$ is: $A(R) = \{a \mid a \models R\}$.

Definition 2.

If there is a formula set R, and Z is the explanation set, then the formula set that can satisfy the explanation set $R \| Z = \{u \in R \mid \exists a \in Z, a \models u\}$.

Definition 3.

$a(v)$ is $v's$ value of explanation a, Z is the explanations set, and B is the atoms set. Then the operation $a[B \to l] = \begin{cases} a(v) = l, v \in B \\ a(v), else \end{cases}$ is a replace operation of the explanation, and $Z[B \to l] = \{a[B \to l] \mid \forall a \in A\}$ is a replace operation of the explanation set.

In the above expressions, g and u are both formulas, a is an explanation, and $l \in \{0,1\}$.

3 A Compromise-Negotiation framework based on Game theory

Considering the similarity between the negotiation of requirements conflict and the Decision theory, this paper proposes a Compromise-Negotiation framework based on Game theory. Under this framework, requirements inconsistency caused by the requirement changes can be eliminated.

3.1 Requirements classification

Generally speaking, the requirements' priority is determined by their importance and urgency. For example, the most popular way is to divide according to three priorities order [17] or five priorities order [18]. In addition, the priorities order is defined by the expectation, the numerical estimate of the cost and risk of the requirements description, such as the Cost value classification method [19] and the Quality function scheduling division method [20]. In this paper, we adopt the three priorities order to divide requirements and the equivalence classes set.

Definition 4.

If L^m is a priority order, $L^m = \{l_1, ..., l_m\} (m \in N)$, and the set $S = \{a_1, ..., a_n\}$, for arbitrary element a_i and $a_j (i, j \in n, n \in N)$, $(a_i, a_j) \in R$. If they have the same priority l_h, then R is a priority equivalence relation.

According to this equivalence relation, the elements of S can be divided into different equivalence classes $\Delta^h (h \in m)$, then define the set $S_\Delta = \{\Delta^h \mid (h \in m)\}$ as the equivalence classes set. As a result, the requirements customer C_1 and C_2 divided the equivalence classes set according to the priority equivalence relation *R*.

3.2 Formalization of Game theory

Game theory is a branch of mathematics often used by economists, to work out how events will unfold as people and organizations act in what they perceive to be their best interests [16]. In the computer field, there are no common logic symbols and formulas. Therefore, we give the formal representation of the Game theory which is suitable for the requirements problem.

In the negotiation model of Game theory with two participants the participants need to use the quantitative values, namely the expectation pair, to denote the desire level of choosing one strategy. Define the formalization as follows:

Definition 5.

If $C_1's$ $S_\Delta = \{\Delta^1, ..., \Delta^n\}$, $C_2's$ $T_\nabla = \{\nabla^1, ..., \nabla^n\}$, we first compute the Cartesian product of $S_\Delta * T_\nabla$, which $S_\Delta * T_\nabla = \{(\Delta^i, \nabla^j) \mid \Delta^i \in S_\Delta, \nabla^j \in T_\nabla\}$. For an arbitrary order pair $(\Delta^i, \nabla^j) \in S_\Delta * T_\nabla$, we set an expectation pair (p_{ij}, q_{ij}) for it. The rules that are used for setting the expectation pair can be described as follows:

Under the equivalence class ∇^c of T_∇, we set the expectation p_{ic} of $S_\Delta's$ equivalence class Δ^i. Suppose that G is the collection of $S_\Delta's$ atoms set and $T_\nabla's$ atoms set, G_{∇^c} is the atom set of ∇^c in T_∇, $E = \{a \mid a \models \nabla^c\}$, $A = \{a \mid a \models \Delta^1 \cup ... \cup \Delta^{i-1}\}$, A_0 and A_1 are two atoms sets and $A_0 = \{v \mid \forall a \in A, a(v) = 0\}$ $A_1 = \{v \mid \forall a \in A, a(v) = 1\}$.

Then $B_0 = (G - G_{\nabla^c}) \cap A_0$, $B_1 = (G - G_{\nabla^c}) \cap A_1$, $A' = E \cap E[B_0 \to 0] \cap E[B_1 \to 1]$. So we see that: $p_{ic} = Max\{|x| \mid \exists a \in A', (a \models x) \wedge (x \subseteq \Delta^i)\}$.

(Symmetrically describe q_{cj})

Therefore, there is a mapping from the ordered pair (Δ^i, ∇^j) and the expectation pair (p_{ij}, q_{ij}), namely, $f(\Delta^i, \nabla^j) = (p_{ij}, q_{ij})$.

Suppose $f(\Delta^i, \nabla^c) = (p_{ic}, q_{ic}), f(\Delta^j, \nabla^c) = (p_{jc}, q_{jc})$, then $(p_{ic}, q_{ic}) < (p_{jc}, q_{jc})$ iff $p_{ic} < p_{jc}$. We have the rules that under the condition when C_2 is choosing the equivalence class ∇^c, the profit of the equivalence class Δ^i for the requirement customer C_1 is lower (equal or greater) than the profit of Δ^j. (Similarly express the relationship of ∇^i and ∇^j)

Definition 6.

The pure strategy is a form of choosing strategies for the negotiators. Under the pure strategy, the participant C_1 can choose any equivalence class in S_Δ, denoted as $PureS_{C_1}$. In which, $PureS_{C_1} = S_\Delta$. (Similarly, define the $PureS_{C_2} = T_\nabla$)

Definition 7.

The mixed strategy is also a form of choosing strategies for the negotiators. In this form, the negotiators can choose the strategies according to a certain probability distribution on the basis of the known pure

strategy. C_1 can choose the equivalence class in S_Δ according to the probability distribution $(P_1,...,P_n)$ on the basis of the already known pure strategy. This form could be called the mixed strategy, denoted as: $MixS_{C_1}=(P_1,...,P_n)$.

(Similarly, define the $MixS_{C_2}=(Q_1,...,Q_n)$)

In the above expressions, c is a constant, a is the explanation of all the atoms in set S and T, $i,j,c\in n$, $n\in N$, $P_i>0$, $Q_j>0$, $P_1+...+P_n=1$, $Q_1+...+Q_n=1$.

3.3 The Nash equilibrium selection strategy

In the negotiation model of Game theory, there are two participants and each participant seeks its best strategy among three choices. The NE of pure strategy is a set of ordered pairs consisting of the best strategy of each participant's pure strategy and also is negotiation strategy that can make both sides achieve the best strategies. If there is an NE of the pure strategy, it could come out with the best result of the negotiation. To choose the NE, some useful definitions are as follows:

Definition 8.

Use $S_\Delta \bullet_N^{Pure} \nabla^c$ to denote the set of the ordered pair whose expectation pair has the highest value among the ordered pairs that come from the Δ^i in S_Δ and the ∇^c in T_∇. This set is called the pure strategy's best strategy to ∇^c in T_∇ for S_Δ.

Namely, $S_\Delta \bullet_N^{Pure} \nabla^c =$

$\{(\Delta^i,\nabla^c)\,|\,(\Delta^i\in S_\Delta)\wedge(f(\Delta^i,\nabla^c)=(\max_p,q_c))\}$.

In which, $(\max_p,q_c)=Max\{f(\Delta^i,\nabla^c)\,|\,\forall\Delta^i\in S_\Delta\}$. (Similarly, define $\Delta^c \bullet_N^{Pure} T_\nabla$)

And define $S_\Delta \bullet_N^{Pure} T_\nabla=\cup_{u=1}^n(S_\Delta \bullet_N^{Pure}\nabla^u)$ as the best strategy for S_Δ to all the pure strategies in T_∇. (Similarly, define $T_\nabla \bullet_N^{Pure} S_\Delta$)

Definition 9.

If $\Delta^c\in S_\Delta,\forall\nabla^i\in T_\nabla$, $(\Delta^c,\nabla^i)\notin(S_\Delta \bullet_N^{Pure}\nabla^i)$, then Δ^c is the dominated strategy for S_Δ to all the pure strategies in T_∇. And $S_\Delta \bullet_N^{WPure} T_\nabla=\{\Delta^c\,|\,\Delta^c$ is the dominated strategy for S_Δ to $T_\nabla\}$. (Similarly, define $T_\nabla \bullet_N^{WPure} S_\Delta$)

Definition 10.

Define $S_\Delta \bullet_{NE}^{Pure} T_\nabla=((S_\Delta \bullet_N^{Pure} T_\nabla)\cap(T_\nabla \bullet_N^{Pure} S_\Delta))$. Thus, base on the definition we can see that: to $\forall(\Delta,\nabla)\in(S_\Delta \bullet_{NE}^{Pure} T_\nabla)$, (Δ,∇) is a NE.

Definition 11.

Define the profit of Δ^c in C_1 as: $\sum_{v=1}^n p_{cv}*Q_v$, denoted as $PayoffS_{C_1\Delta^c}$. (Similarly, define $PayoffS_{C_2\nabla^c}$)

Definition 12.

Set $PayoffS_{C_2\nabla^j}$ of C_2 to be equal, then we can solve the mixed strategy of C_1, denoted as: $MixS_{C_1}=(P_1,...,P_n)$. Set $PayoffS_{C_1\Delta^i}$ of C_1 to be equal, then we can solve the mixed strategy of C_2, denoted as: $MixS_{C_2}=(Q_1,...,Q_n)$.

And define $S_\Delta \bullet_{NE}^{Mix} T_\nabla=(MixS_{C_1},MixS_{C_2})$ as an NE of the mixed strategy.

According to the Game Theory, the NE of the mixed strategy is a probability distribution pair. But if the strategy's NE becomes 1, it is a pure strategy, namely, the NE of the mixed strategy does not give the specific method about employing which strategy for both sides. Therefore, in this section, define the expectation sum based on the NE of the mixed strategy and the method needs to choose strategies based on the NE of the mixed strategy.

Definition 13.

Under an NE of mixed strategy $S_\Delta \bullet_{NE}^{Mix} T_\nabla=(MixS_{C_1},MixS_{C_2})$, define $C_1's$ expectation sum as $\sum_{u=1}^n p_{uc}*P_u$ under the strategy of ∇^c, denoted as $Sum_{C_1\nabla^c}$. And define $C_2's$ expectation sum as $\sum_{v=1}^n q_{cv}*Q_v$ under the strategy Δ^c of C_1, denoted as $Sum_{C_2\Delta^c}$.

Definition 14.

Under a NE of mixed strategy $S_\Delta \bullet_{NE}^{Mix} T_\nabla=(MixS_{C_1},MixS_{C_2})$, define the strategy choosing method of C_1 and C_2 as:

$ChooseMix_{C_1}=\{(\Delta^i,\nabla^j)\,|\,P_i=Max\{P_1,...,P_n\}\}$,

$Sum_{C_1\nabla^j}=Max\{Sum_{C_1\nabla^1},...,Sum_{C_1\nabla^n}\}$.

$ChooseMix_{C_2}=\{(\Delta^x,\nabla^y)\,|\,Q_y=Max\{Q_1,...,Q_n\}\}$,

$Sum_{C_2\Delta^x}=Max\{Sum_{C_2\Delta^1},...,Sum_{C_2\Delta^n}\}$.

Then the method of choosing strategies is:

$ChooseMix=ChooseMix_{C_1}\cup ChooseMix_{C_2}$.

In which, c is a constant, and $c,i,j,x,y\in n,n\in N$.

3.4 The choose algorithm

Definition 15.

If $ChoosePure\cap ChooseMix\neq\varnothing$, we choose the ordered pairs set to form the system, otherwise we choose the ordered pairs set $ChoosePure\cup ChooseMix$.

So the choose algorithm can be described as Fig.1. In which, CA will be introduced in next section.

3.5 The compromise algorithm

The negotiation framework proposed in this paper makes both negotiators to get satisfied and reasonable outcome, in which at least one side makes concessions in the negotiation process. Thus the compromise negotiation framework eliminates the inconsistencies existing in the system.

Problem domain refers to the scope of the questions, internal relations of the problems and the logical possibility space. In this paper, the interpretations set of the system set is the problem domain set.

Definition 16.

If the system set Sys is established, its interpretations set is the problem domain set, denoted as $ESys$. Then set the initial value of the system set Sys as all the atoms that appear in the negotiation, the initial value of the problem domain set $ESys$ as the interpretations set of the system set Sys.

Definition 17.

If given the formulas set R, the relationship of interpretation $=_\Omega$ and $>_\Omega$ are:

For $\forall a,b, a =_\Omega b$, if $\left|R \| \{a\}\right| = \left|R \| \{b\}\right|$. And $a >_\Omega b$ if $\left|R \| \{a\}\right| > \left|R \| \{b\}\right|$. In which, a,b is an inter pretation separately.

Definition 18.

If given the formulas set R and the interpretations set E, $R's$ divide-operation $\circ$ to E is: $R \circ E = \{k^1,...,k^n\}$ .In which, $k^1,...,k^n$ is a division of E, and $\forall a,b \in k^i, a =_\Omega b, (1 \le i \le n)$, $\forall a \in k^i, \forall b \in k^j$, $a >_\Omega b, (1 \le i < j \le n)$.

Definition 19.

If given the equivalence class set $\Delta^m (m \in 1,2,3)$, $\nabla^n (n \in 1,2,3)$ and the problem domain set $ESys$, then the result by using the division operation $\Delta^m \circ ESys$ is the solution set under Δ^m ,denoted as $S^m[i], (i \in N)$. And the result by using the division operation $\nabla^n \circ ESys$ is the solution set under ∇^n, denoted as $T^n[j], (j \in N)$.

Based on compromise and concession, given system set Sys and problem domain set $ESys$, this paper selected the solution set $S^m[i]$ under the equivalence class set Δ^m and $T^n[j]$ under the equivalence class set ∇^n to negotiate, and finally eliminated possible inconsistencies of the two negotiation sides. The flow chart of the compromise algorithm can be described as Fig.2.

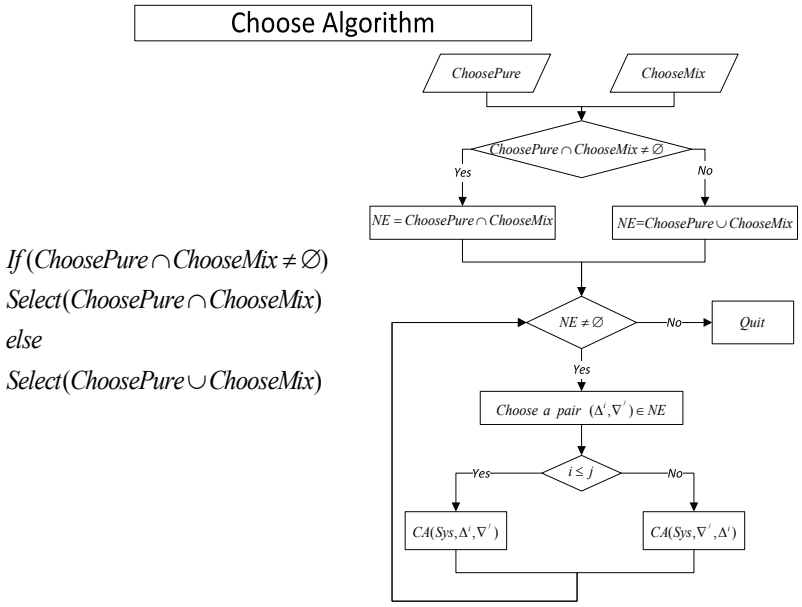

Figure 1 The choose algorithm

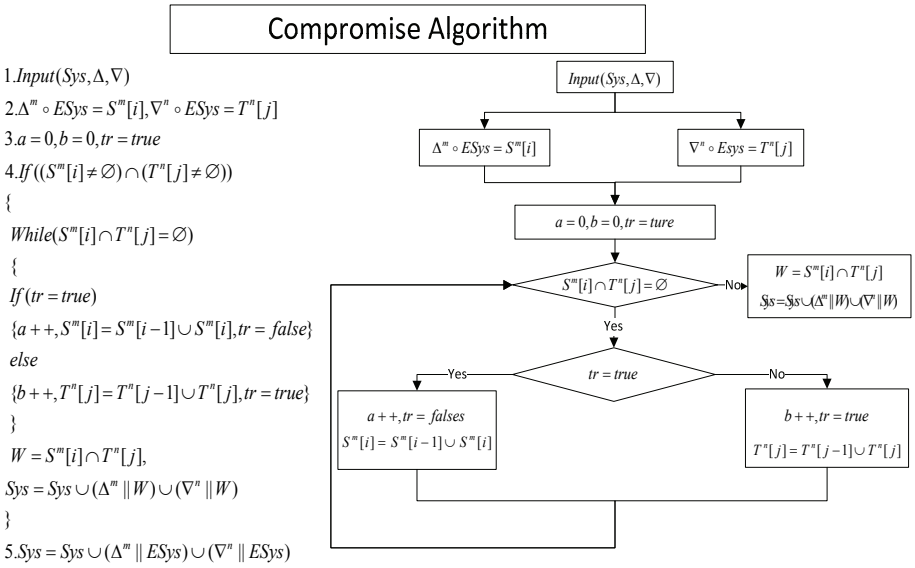

Figure 2 The compromise algorithm

3.6 The Compromise-Negotiation framework

Based on the above work, we give the Compromise-Negotiation framework as Fig 3.

The whole process is composed of two phases: the matrix game and the Compromise. In the matrix game, no matter whether there is NE of the pure strategy or not, there must be NE of a mixed strategy, and the compromise algorithm is finite. So the above framework also meets the finite nature.

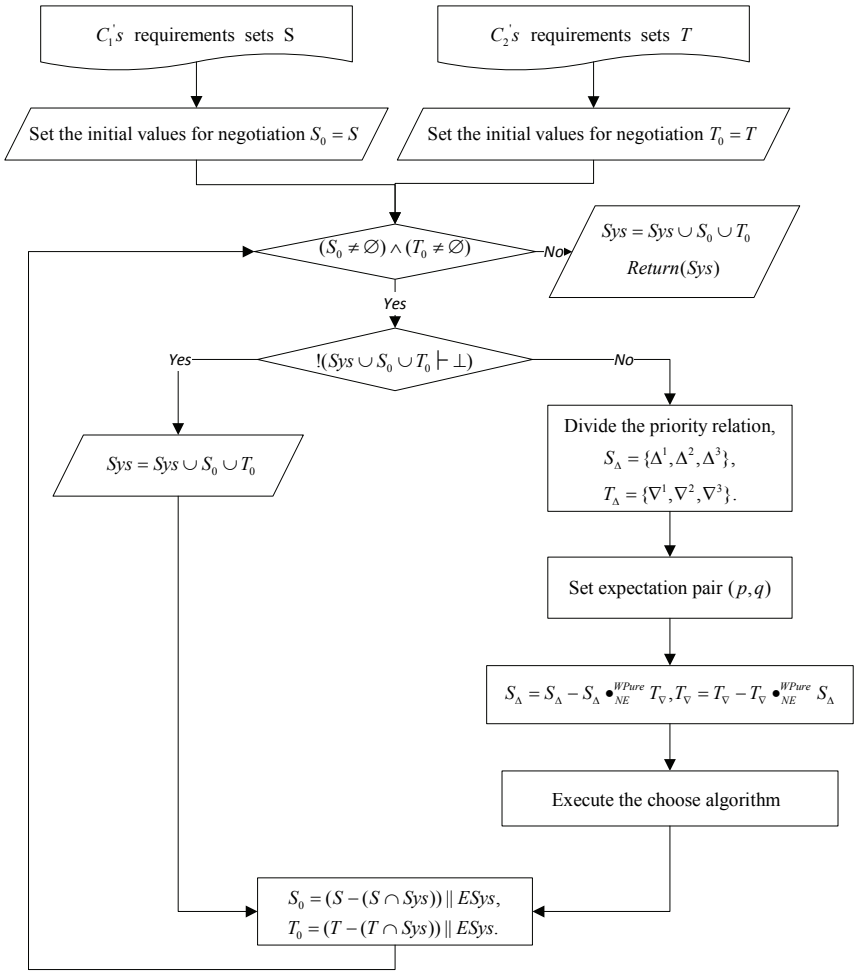

Figure 3 Comprise-Negotiation framework based on Game theory

4 A software engineering case study

In this section, we apply our Compromise-Negotiation framework in an actual case.

4.1 Requirements specification of the Access Control System

Currently, most parking lots of the residential areas are installed with the Access Control System [12]. The Access Control System is controlled by the computer. It is a management system that manages the passageway.

In the Access Control System of the residential area, Manages the Parking requirements has three priority requirements which are included as (1-4) in Tab.1, Manages the Fire engines requirements has three priority requirements which are included as (5-7) in Tab. 1.

To facilitate the description, we use $Aut(x)$ to denote x is awarded a special license, use $Ent(x)$ to denote x can enter the residential area, use $Ala(x)$ to denote if x tries to enter the residential area the alarm will be triggered, use $Push(x, y)$ to denote x pushes the button y, use $Eme(x)$ to denote x is the emergency engine, use $entr$ to denote the button for entering the residential area, use $fire\_e$ to denote the fire engines.

Table 1 The requirements of manages the parking

(1) The car is not allowed to enter the residential area without a specific permission.
(2) The car is allowed to enter the residential area with a specific permission.
(3) The situation when a car tries to enter the residential area without a specific permission will trigger the alarm.
(4) If the alarm is triggered, the owner of the car will not be able to press the button for entering the residential area again.
(5) The fire engines are regarded as emergency vehicles.
(6) The emergency engine is allowed to enter the residential area without a specific permission.
(7) In addition to the emergency engines, the others are not allowed to enter the residential area without a specific permission.

4.2 Logical representation of the Access Control System

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

$$S=\begin{cases}\neg Aut(fire\_e)\rightarrow\neg Ent(fire\_e),\\ Aut(fire\_e)\rightarrow Ent(fire\_e),\\ \neg Aut(fire\_e)\rightarrow\neg Ala(fire\_e),\\ Ala(fire\_e)\rightarrow\neg Push(fire\_e,entr).\end{cases}, \text{ and}$$

$$T=\begin{cases}Eme(fire\_e),\\ Eme(fire\_e)\rightarrow Ent(fire\_e)\\ \wedge\neg Aut(fire\_e),\\ \neg Aut(fire\_e)\wedge\neg Eme(fire\_e)\\ \rightarrow\neg Ent(fire\_e).\end{cases}.$$

The presentation of the logic requirements is completed. Nonetheless, we find that $Aut(fire\_e)$ and $\neg Aut(fire\_e)$ are included in the requirements set $S\cup T$ at the same time. Thus we know that: $S\cup T\ \perp$. Evidently, the final goal of this case is eliminating the inconsistency in the requirements set.

4.3 Apply the Compromise-Negotiation framework for eliminating inconsistency

To be simple, we use letters to replace the requirements in the requirements set.

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

Then $S=\{a\vee\neg b,\neg a\vee b,a\vee\neg c,\neg c\vee\neg d\}$,

$T=\{e,\neg e\vee(\neg a\wedge b),a\vee e\vee\neg b\}$.

Execute the Framework:

The initial value of the framework: Set the initial value: $S_0=S,T_0=T,Merge=\varnothing$. We can see that $S_0\neq\varnothing$ and $T_0\neq\varnothing$, $S\cup T\vdash\perp$, $\perp$ is $a\wedge\neg a$.

Execute the First Cycle Negotiation:

C_1 and C_2 divide equivalence classes for S_0 and T_0, and get the equivalence classes: $S_\Delta=\{\Delta^1,\Delta^2,\Delta^3\}$, $T_\Delta=\{\nabla^1,\nabla^2,\nabla^3\}$.

In which: $\Delta^1=\{a\vee\neg b\}$, $\Delta^2=\{\neg a\vee b,a\vee\neg c\}$, $\Delta^3=\{\neg c\vee\neg d\}$, $\nabla^1=\{e\}$, $\nabla^2=\{\neg e\vee(\neg a\wedge b)\}$, $\nabla^3=\{a\vee e\vee\neg b\}$.

Then we use a third-order matrix to denote the expectation pair (p,q), set as Tab. 2.

Table 2 Negotiation matrix

C_1 \ C_2		T_∇		
		∇^1	∇^2	∇^3
S_Δ	Δ^1	(1,1)	(1,0)	(1,1)
	Δ^2	(2,1)	(2,1)	(2,1)
	Δ^3	(1,1)	(1,1)	(1,1)

So we know: $S_\Delta \bullet_{NE}^{WPure} T_\nabla=\{\Delta^1,\nabla^3\}$. At this time:

$ChoosePure=\{(\Delta^2,\nabla^1),(\Delta^2,\nabla^2),(\Delta^2,\nabla^3)\}$,

$ChooseMix=\{\varnothing\}$.

And we get: NE=$ChoosePure\cup ChooseMix$

$=\{(\Delta^2,\nabla^1),(\Delta^2,\nabla^2),(\Delta^2,\nabla^3)\}$.

Execute The Step:

First-Step:

Now, NE=$\{(\Delta^2,\nabla^1),(\Delta^2,\nabla^2),(\Delta^2,\nabla^3)\}\neq\varnothing$, first we choose (Δ^2,∇^1), then execute $CA(Sys,\Delta^2,\nabla^1)$.

Because $Sys=\varnothing$, so $ESys=\{00000,...,11111\}$.

Use the operation $\Delta^2\circ ESys$ can get: $S^1[0]=\{11xxx,0x0xx\}$, and use the operation $\nabla^1\circ ESys$ can get: $T^1[0]=\{xxxx1\}$. In which, $x=\{0,1\}$. (we don't give $S^1[1]$ and $T^1[1]$ any more)

Because $W=S^1[0]\cap T^1[0]=\{11xx1,0x0x1\}\neq\varnothing$, so both of S and T are compromised directly.

Then

$Sys=Sys\cup(\Delta^2\,||\,W)\cup(\nabla^1\,||\,W)=\{\neg a\vee b,a\vee\neg c,e\}$.

Second-Step:

At this time, NE=$\{(\Delta^2,\nabla^2),(\Delta^2,\nabla^3)\}\neq\varnothing$, second we choose (Δ^2,∇^2), then execute $CA(Sys,\Delta^2,\nabla^2)$.

Now $Sys=\{\neg a\vee b,a\vee\neg c,e\}$, $ESys=\{11xx1,0x0x1\}$.

So use the operation $\Delta^2\circ ESys$ and $\nabla^2\circ ESys$ again, we can get:

$S^1[0]=\{11xx1,0x0x1\}$ and $T^1[0]=\{01xx1,xxxx0\}$.

Because $W=S^1[0]\cap T^1[0]=\{010x1\}\neq\varnothing$, so both of S and T are compromised again.

Then, $Sys=Sys\cup(\Delta^2\,||\,W)\cup(\nabla^2\,||\,W)$

$=\{\neg a\vee b,a\vee\neg c,e,\neg e\vee(\neg a\wedge b)\}$.

Third-Step:

NE=$\{(\Delta^2,\nabla^3)\}\neq\varnothing$, finally we choose (Δ^2,∇^3), then execute $CA(Sys,\Delta^2,\nabla^3)$.

Now $Sys=\{\neg a\vee b,a\vee\neg c,e,\neg e\vee(\neg a\wedge b)\}$,

$ESys=\{010x1\}$.

So use the operation $\Delta^2\circ ESys$ and $\nabla^3\circ ESys$ again, we can get: $S^1[0]=\{010x1\}$, $T^1[0]=\{01xx1\}$.

Because $W=S^1[0]\cap T^1[0]=\{010x1\}\neq\varnothing$, so both of S and T are compromised in final.

And we find NE=$\varnothing$ now, so it's not needed to execute the CA.

The Step is Completed.

Update

$S_0=(S-(S\cap Result))\,||\,A(Result)=\{\neg c\vee\neg d\}$ and

$T_0=(T-(T\cap Result))\,||\,A(Result)=\{\varnothing\}$.

Now $S_0\neq\varnothing$ but $T_0=\varnothing$, so it's not needed to execute the Cycle Negotiations.

The first Cycle Negotiation is completed.

Then update

$Sys=\{\neg a\vee b,a\vee\neg c,e,\neg e\vee(\neg a\wedge b),a\vee e\vee\neg b,\neg c\vee\neg d\}$

and return it.

The framework is completed.

Thus, we can get the final negotiation result: $\{\neg a \vee b, a \vee \neg c, e, \neg e \vee (\neg a \wedge b), a \vee e \vee \neg b, \neg c \vee \neg d\}$ .It is consistent and the eliminated inconsistency is: $\{a \vee \neg b\}$.

Then the final requirements of the system can be described in Natural language in Tab. 3.

Table 3 The consistent system requirements

(1) The car is allowed to enter the residential area with a specific permission.
(2) The situation when a car tries to enter the residential area without a specific permission will trigger the alarm.
(3) The fire engines are regarded as emergency vehicles.
(4) The emergency engine is allowed to enter the residential area without a specific permission.
(5) In addition to the emergency engines, the others are not allowed to enter the residential area without a specific permission.
(6) If the alarm is triggered, the owner of the car will not be able to press the button for entering the residential area again.

5 Conclusion

In order to certify our method has much efficiency and flexibility, we designed an experiment to compare with the reference [6] and [12]. 1000 requirements sets were selected as input, and we used the architecture in [6] (represented by the red line), [12] (represented by the yellow line) and the one proposed in this paper (represented by the blue line) respectively to handle with the existing inconsistencies.

Experiment 1 samples the number of data to handle with and the running time of the architecture, **Experiment 2** samples the number of data to handle with and the number of inconsistencies that the architecture handles with. Both of them have the sampling interval, set to 100. When the architecture runs normally, after handling with 100 data, namely a time interval, the architecture outputs the running time and the number of inconsistencies to the PC, as shown in Fig. 4.

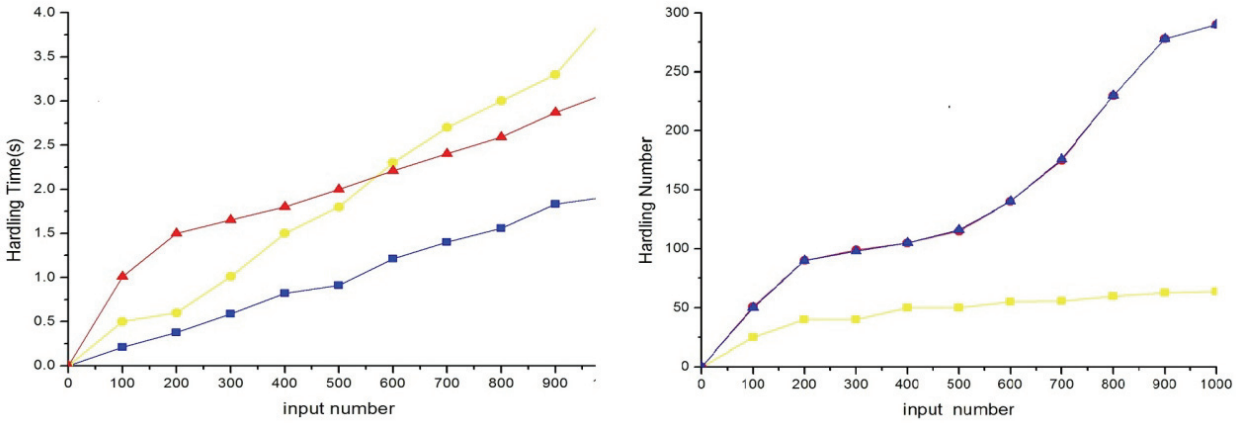

Figure 4 The data comparison in Experiment 1 and Experiment 2

The results show that: 1) The handling time of the architecture we proposed is less than the architecture [6] and the architecture [12], which proved the architecture. 2) The architecture proposed in this paper can handle with more inconsistencies than the architecture [12] and equal to the architecture [6], which proved the flexibility of the architecture.

6 Conclusion and future works

The cycle Compromise-Negotiations based on the idea of Game theory proposed in this paper can handle most requirements inconsistency. We discussed the reasons that caused the requirements inconsistency. Moreover, under the guidance of the idea of the Compromise-Negotiations, we eliminated the inconsistency among the requirements successfully by transforming the Game Theory's idea to the symbols and formulas used in our framework.

Besides, designing solutions with the idea of fuzzy mathematics [21] and Combination Vote [22] will be the target of this paper in future. We will continue to propose more perfect and comprehensive frameworks to manage the changes in requirements.

Acknowledgements

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

7 References

[1] Davis, A. M. Software requirements: objects, functions, and states. Prentice-Hall, Inc.1993.

[2] Perini, A.; Susi, A.; Avesani, P. A machine learning approach to software requirements prioritization. // IEEE Transactions on Software Engineering. 39, 4(2013), pp. 445-461. DOI: 10.1109/TSE.2012.52

[3] Pohl, K. Requirements engineering: fundamentals, principles, and techniques. Springer Publishing Company, Incorporated, 2010. DOI: 10.1007/978-3-642-12578-2

[4] Zowghi, D.; Jin, Z. A framework for the elicitation and analysis of information technology service requirements and their alignment with enterprise business goals. // In Computer Software and Applications Conference

Workshops (COMPSACW), 2010 IEEE 34th Annual. IEEE, (2010, July), pp. 269-272.
[5] Bano, M.; Zowghi, D. User involvement in software development and system success: a systematic literature review. // In Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering. ACM, (2013, April), pp. 125-130. DOI: 10.1145/2460999.2461017
[6] Alchourron, C. E.; Grdenfors, P.; Makinson, D. On the logic of theory change: Partial meet contraction and revision functions. // J. Symb. Log. 50, 2(1985). pp. 510-530.DOI: 10.2307/2274239
[7] Garcez, A. D. A.; Russo, A.; Nuseibeh, B.; Kramer, J. Combining abductive reasoning and inductive learning to evolve requirements specifications. // IEE Proceedings-Software, 150, 1(2003), pp. 25-38.DOI: 10.1049/ip-sen:20030207
[8] Borges, R. V.; d'Avila Garcez, A.; Lamb, L. C.;Nuseibeh, B. Learning to adapt requirements specifications of evolving systems (NIER track). // In Proceedings of the 33rd International Conference on Software Engineering. ACM, (2011, May), pp. 856-859.
[9] Booth, R. 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, July), pp. 137-150.
[10] Mu, K.; Jin, Z.; Lu, R.; Peng, Y. Handling non-canonical software requirements based on annotated predicate calculus. // Knowledge and Information Systems. 11, 1(2007), pp. 85-104.DOI: 10.1007/s10115-006-0021-y
[11] Mu, K.; Liu, W.; Jin, Z. An approach to generating proposals for handling inconsistent software requirements. // In Knowledge Science, Engineering and Management. Springer Berlin Heidelberg, (2011), pp. 32-43. DOI: 10.1007/978-3-642-25975-3_4
[12] Mu, K. D.; Liu, W.; Jin, Z.; Hong, J.; Bell, D. Managing software requirements changes based on negotiation-style revision. // Journal of Computer Science and Technology. 26, 5(2011), pp. 890-907.DOI: 10.1007/s11390-011-0187-y
[13] Mu, K.; Liu, W.; Jin, Z. Measuring the blame of each formula for inconsistent prioritized knowledge bases. // Journal of Logic and Computation. 22, 3(2012), pp. 481-516.DOI: 10.1093/jigpal/exr002
[14] Yirui Zhang, Ying Jin, Jianxiu Bai, Jing Zhang. A Negotiation Framework for Managing the Requirements Changes. // Cybernetics and Information Technologies. Volume 13, Issue Special Issue, Pages 75–87, ISSN (Print) 1314-4081, DOI: 10.2478/cait-2013-0039, December 2013
[15] Dixit, A. K. Thinking strategically: The competitive edge in business, politics, and everyday life. WW Norton & Company, 1991.
[16] Myerson, R. B. Game theory. Harvard university press, 2013.
[17] Wiegers, K. First things first: prioritizing requirements. // Software Development. 7, 9(1999), pp. 48-53.
[18] Davis, A. Just enough requirements management: where software development meets marketing. Addison-Wesley, 2013.
[19] Karlsson, J.; Ryan, K. A cost-value approach for prioritizing requirements. // Software, IEEE. 14, 5(1997), pp. 67-74.DOI: 10.1109/52.605933
[20] Pardee, W. J. To Satisfy & Delight Your Customer. Dorset House Publishing Company, Incorporated, 1996.
[21] Ni, M.; Chen, W. Analysis and Evaluation for the College Financial Risk Based on Fuzzy Mathematics. // In Computer and Management (CAMAN), 2011 International Conference on IEEE, 2011, pp. 1-4.
[22] Ahn, D. S.; Oliveros, S. Combinatorial voting. // Econometrica. 80, 1(2012), pp. 89-141. DOI: 10.3982/ECTA9294

Authors' addresses

Tie Feng
Jilin University, College of Computer Science and Technology,
No 2699 Qianjin Street, Changchun, Jilin Province, China
E-mail: fengtie@jlu.edu.cn

Yi-rui Zhang
Jilin University, College of Computer Science and Technology,
130012, Changchun, China
E-mail: SoNbility@163.com

Ying Jin
Jilin University, College of Computer Science and Technology,
130012, Changchun, China
E-mail: jinying@jlu.edu.cn

Jing Zhang, *corresponding author*
Jilin University, College of Computer Science and Technology,
130012, Changchun, China
E-mail: zhangjing99@jlu.edu.cn