Proceedings of the 2012 2nd International Conference on Computer and Information Application (ICCIA 2012)

# Dynamic Evolution of Requirement Goal Deployed on Network Environment

Yang Liu
University of Electronic Science and Technology of China, Chengdu, Sichuan, 610041, China;
Chengdu Institute of Computer Application Chinese Academy of Sciences, Chengdu, 610041, China
ly1246@qq.com

Jinzhao Wu*
Guangxi University for Nationalities, Nanning, Guangxi, 530003, China;
Beijing Jiaotong University, Beijing, 100191, China
wjzcd2011@sina.com

***Abstract*—Under the network environment, it inevitably exists inconsistency and dynamic information in users' requirements because of the complicated and volatile network, update, restructuring and so on. This paper studies how to handle the inconsistent information and keep correctness of the requirement goals from the users. It also provides relevant rules of model evolution under dynamic environment. With maximal consistent set, the inconsistent information can be reserved to keep the indeterminacy in the early stage and expand the function of the software. It is helpful for the next step of the software development.**

*Keywords-requirement goal; inconsistent information; maximal consistent set; software development; dynamic information*

## I. INTRODUCTION

The requirements of software system are from knowledge expectations of users and designers to realistic social[1]. It is too hard to avoid inconsistent information in requirements from these knowledge expectations[2,3,4]. Because of the complexity of the network software and the fluidity of the needs from users, the key problem of system development is how to analyze and handle the possible inconsistent information and keep the correctness of the requirement goals under the dynamic environment.

Nowadays, the traditional target analysis methods as KAOS[5] and Tropos[6], can solve some problems, but they all can't deal with inconsistent information and the relationship describing is single. There are two methods to deal with inconsistent information from requirements analysis: (1)eliminate the inconsistent information to maintain the consistent information or discuss the inconsistent messages again with the users to reach an agreement gradually, (2)tolerate the inconsistent information in the greatest degree by paraconsistent[7] formal description to treat the contradictions in specification as a natural phenomenon. The first view can eliminate contradictions and maintenance the consistency of requirements analysis, but this method may be lost a lot of valuable information, and it is too difficult to reach an agreement for different users with their own interest and individual characters. In addition, when the users need a little dynamic change, it may bring a huge adjustment for the whole target structure.

In this study, motivated by the above requirements, we propose using maximal consistent set which is one of the paraconsistent ideas to describe users' target from requirements elicitation, and transform the conflict requirements demands according to the goal's attribute into some kind goals which can be chosen freely by the users. We also give the rules which can change with the dynamic changes from users.

The rest of the paper is organized as follows. Section Ⅱ discusses the background knowledge about maximal consistent set, and describes the inconsistent information with maximal consistent set. Modeling the Goal Model and the rules of dynamic evolution are discussed in section Ⅲ including adding, deleting and restructuring. Section Ⅳ concludes the paper and discussed future work.

## II. BACKGROUND

For drawing inferences from inconsistent information, it has produced some views form some consistent subsets[8,9]. Various approaches to the applications of paraconsistency to knowledge bases[10], and consistent subset is a branch of paraconsistent logic. In this section, we use it to describe the extraction of demand information before goal modeling. Then, with the guidance of RGPS-G, we give the GM representation.

### A. Extraction of requirements goals

During the requirements elicitation, we can clearly get the demands of users with SORL(Services-Oriennted Requirements Language)[11]. For any goals "g", it can be represents as:

$$\mathrm{Goal}(g) \wedge \mathrm{FunctionalGoal}(g,x,y,z) \wedge \mathrm{NonFunctionalGoal}(G,t,c,v,u,d)$$

"x, y, z" represents "Operation, Object, Manner" of functional goal. "t, c, v, u, d" represent "Type, Compare, Value, Unite, Degree" of non-functional goal, respectively.

In this paper, we focus on the relationships between goals[12,13]. We give a few related set definitions as follows.

**Definition 1** $\Delta$ is the set of describing extraction goals, then the problem set of $\Delta$ is defined as in (1).

$$PC(\Delta) = \bigcup MI(\Delta) \qquad (1)$$

Here,

$$MI(\Delta) = \left\{ \Gamma \in INC(\Delta) \middle| \forall I \in INC(\Delta), I \not\subset \Gamma \right\}$$

$$INC(\Delta) = \left\{ \Gamma \subseteq \Delta \middle| \Gamma \vdash \bot \right\}$$

**Definition 2** $\Delta$ is the set of describing extraction goals, then the free set of $\Delta$ is defined as follows:

$$FC(\Delta) = \bigcap MC(\Delta) \tag{2}$$

Here,

$$MC(\Delta) = \left\{ \Gamma \in CON(\Delta) \middle| \forall I \in CON(\Delta), \Gamma \not\subset I \right\}$$

$$CON(\Delta) = \left\{ \Gamma \subseteq \Delta \middle| \Gamma \mid\not\sim \bot \right\}$$

Specific can reference [8]. It is obvious that problem sets are those sets who has conflict information and which are also need to by analyze by software requirement analysis designers, and free sets are those sets which hasn't conflict information and there is no information need to discuss again.

**Example 1** In requirements extraction, we get the goals "Pay", "Book", and "CheckPassword". In addition, it needs "CheckPassword " in China when you buy somethind, but usually it doesn't need "CheckPassword" to overseas. So, the formulae $\Delta$ is as follows.

$$\Delta = \{Goal(pay), Goal(book), Goal(checkpassword), \neg Goal(checkpassword), Goal(pay) \to Goal(book), Goal(pay) \wedge Goal(checkpassword), Goal(pay) \wedge \neg Goal(checkpassword)\}$$

So,

$$MC(\Delta)_1 = \{Goal(pay), Goal(book), Goal(checkpassword), Goal(pay) \to Goal(book), Goal(pay) \wedge Goal(checkpassword)\}$$

$$MC(\Delta)_2 = \{Goal(pay), \neg Goal(checkpassword), Goal(pay) \to Goal(book), Goal(book), Goal(pay) \wedge \neg Goal(checkpassword)\}$$

$$FREE(\Delta) = MC(\Delta)_1 \cap MC(\Delta)_2 = \left\{Goal(pay), Goal(book), Goal(pay) \to Goal(book)\right\}$$

$$MI(\Delta)_1 = \left\{Goal(checkpassword), \neg Goal(checkpassword)\right\}$$

$$MI(\Delta)_2 = \left\{Goal(pay) \wedge Goal(checkpassword), Goal(pay) \wedge \neg Goal(checkpassword)\right\}$$

$$MI(\Delta)_3 = \left\{Goal(checkpassword), Goal(pay) \wedge \neg Goal(checkpassword)\right\}$$

$$MI(\Delta)_4 = \left\{\neg Goal(checkpassword), Goal(pay) \wedge Goal(checkpassword)\right\}$$

$$PC(\Delta) = MI(\Delta)_1 \cup MI(\Delta)_2 \cup MI(\Delta)_3 \cup MI(\Delta)_4 = \{Goal(checkpassword), \neg Goal(checkpassword), Goal(pay) \wedge Goal(checkpassword), Goal(pay) \wedge \neg Goal(checkpassword)\}$$

We can quickly get the undisputed goals and conflict goals in the example through the problems set and free set division. In this way, the location of contradiction is quickly improved for software requirement designers. Then, the confliction information can be solved under the next step with determination of the relationships between up and low goals.

## B. *Handling inconsistent information*

How to deal with the goals in problem sets is the very important problem which we need to care on. From references[11,12,13], there are six relationships in Goal model between goals. It enriches the single relationship(just as And and Or) to four relationships, and distinguishes core sub-nonfunctional goal "Mandatory" and extensible sub-nonfunctional goal "Optional", "Alternative" and "Or" and two constraint "Depend" and "exclude".

So, we can deal with the inconsistent information by the degree of their inconsistency. During the decomposition of top goals, their relations between goals can be defined as follows.

*a) HasMandatory($g_1$,$g_2$): represents the "Mandatory" relationship between up goal "$g_1$" and low goal "$g_2$".*

*b) HasOptional($g_1$,$g_2$): represents the "Optional" relationship between up goal "$g_1$" and low goal "$g_2$".*

*c) HasAlternative($g_1$,$g_2$): represents the "Alternative" relationship between up goal "$g_1$" and low goal "$g_2$".*

*d) HasOr($g_1$,$g_2$): represents the "Or" relationship between up goal "$g_1$" and low goal "$g_2$".*

*e) HasDepend($g_1$,$g_2$): represents goal "$g_1$" depends on goal "$g_2$".*

*f) HasExclude($g_1$,$g_2$): represents goal "$g_1$" excludes goal "$g_2$".*

Formal modeling just needs to depict the characterization which needs to be verified and simplify the original model by removed the unrelated property information towards verification. In order to simplify the model, we can use functions to filter information, and find the goals at last. The description of requirements goal from users reflects in all kinds of formulae by predicate "Goal" including atomic formulae, disjunctive normal form, conjunctive normal form, and so on.

**Definition 3** $\Delta^{U}$ is the set which has been filtered by functions and only has predicate "Goal", "HasMandatory", "HasOptional", "HasAlternative", "HasOr", "HasDepend", and "HasExclude" from set of formulae $\Delta$. Then, the *Ground State Requirements Set* $\Lambda$ defined in (3).

$$\Lambda = \left\{\Lambda_1, ..., \Lambda_n\right\}, \Lambda_i \in MC\left(\Delta^U\right) \tag{3}$$

## III. GOAL MODELING IN DYNAMIC EVOLUTION

## A. *Goal modeling with GM*

Based on the characteristics of Goal model, formal modeling is mainly considered the relationships between goals and their influence to the achievement of the top goal. So, we use GM to describe it.

**Definition 4** A GM is a tuple $<g_0, O, G, M, A, R, P, D, E, \Sigma>$, where

- $g_0$: is represents the highest goal in Goal Layer;
- $O$: is represents the set of all atomic goals in Goal Layer;
- $G$: is represents the set of all goals in Goal Layer including atomic goals and composition goals;
- $M$: is represents the set of Mandatory relationship chain between up goals and low goals in the whole Goal Layer;
- $A$: is represents the set of Alternative relationship chain between up goals and low goals in the whole Goal Layer;
- $R$: is represents the set of Or relationship chain between up goals and low goals in the whole Goal Layer;
- $P$: is represents the set of Optional relationship chain between up goals and low goals in the whole Goal Layer;
- $D$: is represents the set of Depend relationship chain

between goals in the whole Goal Layer;

- $E$: is represents the set of Exclude relationship chain between goals in the whole Goal Layer;
- $\Sigma$: is represents the mapping from the *Ground State Requirements Set* $\Lambda$ .

The Algorithm of solving $\Sigma$ is as follows. The inputs are the *Ground State Requirements Set* of Goal model, and the output is the element $\Sigma$ of GM. So, we can get all kinds of demands from different users.

**Algorithm 1**(Get$\Sigma$)

```
Input: a Ground State Requirements Set $\Lambda$.
Output: the element $\Sigma$ of GM.
1.  B=$\varnothing$; C=$\varnothing$.
2.  for each $\Lambda_i \in \Lambda$ do
        for each g∈G, do
            if $\Lambda_i \models Goal(g)$ then B=B ∪ {g}
3.  C=C ∪ {B}
    If C=$\varnothing$ ,then return Wrong.
        Else
          Return C
```

A GM model is correctness, iff: $\forall \Sigma_i, \Sigma_i \in \Sigma$

- $g_0 \in \Sigma_i$ ;
- $\forall x \forall y \bullet x \in \Sigma_i \wedge (x,y) \in M \rightarrow y \in \Sigma_i$ ;
- $\forall x \exists y \bullet x \in \Sigma_i \wedge (x,y) \in R \rightarrow y \in \Sigma_i$ ;
- $\forall x \exists y \forall z \bullet x \in \Sigma_i \wedge (x,y) \in A \wedge (x,z) \in A \rightarrow (y \in \Sigma_i \wedge z \notin \Sigma_i)$;
- $\forall x \forall y \bullet x \in \Sigma_i \wedge (x,y) \in D \rightarrow y \in \Sigma_i$ ;
- $\forall x \forall y \bullet x \in \Sigma_i \wedge (x,y) \in E \rightarrow y \notin \Sigma_i$ .

Those rules reflect the properties of the relationship chain between goals in Goal model.

When we verify the correctness of the model, we check whether all the $\Sigma_i$ , $\Sigma_i \in \Sigma$ from the *Ground State Requirements Set* $\Lambda$ satisfy the rules or not actually.

## *B. Dynamic evolution*

One of the remarkable features of networked software is the dynamic change of users' demands. It is very easy to appear dynamic phenomenon for update and restructuring of software. All of the changes of Goal model can be attributed to two kind situations: adding a goal and deleting a goal.

*1) Adding a goal:* The basic idea of adding a goal is adding corresponding new goal and new relationships into elements in GM, and reproducing the new *Ground State Requirements Set* $\Lambda'$ which is brought by the new formal specification. The Algorithm of adding a goal is as follows in algorithm 2.

When we add a new goal, the correctness and completeness of the Goal model need to be verified again.

**Algorithm 2**(AddGoal)

```
Input: a new goal and its corresponding relationships and new formal specifications brought by the Ground State Requirements Set $\Lambda$.
Output: a new GM.
1.  if $\exists x, x \in G$ and $(x,g) \in M' \cup R' \cup A' \cup P'$ then
        for each x do $O=O \setminus \{x\}$.
2.  $G=G \cup \{g\}$; $O=O \cup \{g\}$; $M=M \cup M'$; $A=A \cup A'$;
    $R=R \cup R'$; $P=P \cup P'$; $D=D \cup D'$; $E=E \cup E'$.
3.  $\Sigma$=Get$\Sigma$ ( $\Lambda', G$ )
4.  return $\Omega = (g_0, O, G, M, A, R, P, D, E, \Sigma)$
```

*2) Deleting a goal:* The basic ideas of deleting a goal existed in GM is much easier than adding a new goal. It just needs to remove the goal and its corresponding relationships in GM. The *Ground State Requirements Set* needn't change. The Algorithm of deleting a is divided into two parts: one is sub part of the main algorithm as follows in algorithm 3; the other is main part in algorithm 4. Because composite goals are made up by atomic goals. When we delete a compostie goal, its sub-goals will be influenced.

**Algorithm 3**(DeleteAtomicGoal)

```
Input: a atomic goal and its corresponding relationships.
Output: a new GM.
1.  $G=G \setminus \{g\}$; $O=O \setminus \{g\}$.
2.  for each $\Sigma_i \in \Sigma$ do
        if $\exists x, x \in G$ then $\Sigma_i = \Sigma_i \setminus \{x\}$.
3.  $\Sigma = \bigcup \Sigma_i$ .
4.  if $\exists x, x \in dom\, M \rhd \{g\}$ then $M=M \setminus \{(x,g)\}$.
      Else
        If $\exists x, x \in dom\, A \rhd \{g\}$ then $A=A \setminus \{(x,g)\}$.
          Else
            If $\exists x, x \in dom\, R \rhd \{g\}$ then $R=R \setminus \{(x,g)\}$.
              Else
                $P=P \setminus \{(x,g)\}$.
5.  if $\exists x, x \in dom\, D \rhd \{g\}$ then $D=D \setminus \{(x,g)\}$.
6.  if $\exists x, x \in dom\, E \rhd \{g\}$ then $E=E \setminus \{(x,g)\}$.
7.  if $\exists x, x \in dom\, D \lhd \{g\}$ then $D=D \setminus \{(g,x)\}$.
8.  if $\exists x, x \in dom\, E \lhd \{g\}$ then $E=E \setminus \{(g,x)\}$
9.  if $\exists x, dom\, M \rhd \{g\} \cup dom\, A \rhd \{g\} \cup dom\, R \rhd \{g\}$
        $\cup dom\, P \rhd \{g\} = \{x\}$ then $O=O \cup \{x\}$
10. return $\Omega = (g_0, O, G, M, A, R, P, D, E, \Sigma)$
```

The main Algorithm of deleting any goal needs recursive call Algorithm 3 as follows in algorithm 4.

**Corollary 1** Deleting a goal, the correctness and completeness of the Goal model can be kept.

**Proof.** if a Goal model $\Omega$ is correctness and completeness, then, that is mean that $\forall \Sigma_i, \Sigma_i \in \Sigma$ satisfy the rules. When we delete some $\Sigma_j, \Sigma_j \in \Sigma$, the left $\Sigma_k, \Sigma_k \in \Sigma$ still satisfies the rules.

**Algorithm 4**(DeleteGoal)

**Input**: a goal and its corresponding relationships.
**Output**: a new GM.

```
1.  if g∈$O$ then return $\Omega$ =DeleteAtomicGoal(g).
      else
        C={g}
        for each x, x∈ran M◁C∪ran A◁C∪ran R◁C∪
              ran P◁C do C=C∪{x}
            for each x, x∈C and x∈$O$ do
                $\Omega$ =DeleteAtomicGoal(g)}
2.  return $\Omega$
```

*C. Goal Restructing*

In fact, goal restructuring is a mixed use of adding and deleting goal. The only one place need to pay attention to is that, if the restructuring models share a highest target, then, we need to define a new highest goal to be "$g_0$".

**Definition 5** let

$$\Omega_1 = (g_{01}, O_1, G_1, M_1, A_1, R_1, P_1, D_1, E_1, \Sigma_1)$$

$$\Omega_2 = (g_{02}, O_2, G_2, M_2, A_2, R_2, P_2, D_2, E_2, \Sigma_2)$$

then, the model "$\Omega = (g_0, O, G, M, A, R, P, D, E, \Sigma)$" is a restructuring goal, iff:

- $g_0=g_{01}\Diamond g_{02}$: $g_0$ is the shared top goal of $g_{01}$ and $g_{01}$.
- $O=O_1\cup O_2$;
- $G=G_1\cup G_2$;
- $M=M_1\cup M_2$;
- $A=A_1\cup A_2$;
- $R=R_1\cup R_2$;
- $P=P_1\cup P_2$;
- $D=D_1\cup D_2$;
- $E=E_1\cup E_2$;
- $\Sigma = \bigcup \Sigma_k, \Sigma_k = \Sigma_i \cup \Sigma_j, \Sigma_i \in \Sigma_1, \Sigma_j \in \Sigma_2$.

**Corollary 2** If all the goal models which will compose a larger restructuring goal model are sub-Mandatory goals to the new highest target, the restructuring model can still keep the correctness and completeness; if not, the restructuring model only can keep the correctness.

## IV. CONCLUDE AND FUTURE WORK

Compared with KAOS and Tropos, this method proposed in this paper can express more relations and tolerate some conflicts and accommodate the dynamic evolution. The model of GM likes a tree for the highest target as the root. It not only can describe the relation between low layer goal, but also the relation between up layer goals. Although there are some methods to get the maximal consistent set of formulae, the way how to get the maximal consistent set automaticly and effectively, is still our future work.

## ACKNOWLEDGMENT

This work is supported by the Guangxi Key Laboratory of Hybrid Computational and IC Design Analysis Open Fund under Grant No.HCIC201101, the National Natural Science Foundation of China under Grant 60973147, the Natural Science Foundation of Guangxi under Grant No.2011GXNSFA018154, the Science and Technology Foundation of Guangxi under Grant No. 10169-1, and Guangxi Scientific Research Project No.201012MS274.

## REFERENCES

[1] B. Nuseibeh, S. Easterbrook, "Requirements Engineering: A Roadmap", Proc. of the 22th Int'1 Conf. on Software Engineering, Future of Software Engineering Track. Limerick: IEEE Computer Press, 2000, pp: 35-46.

[2] R. Balzer, "Tolerating Inconsistency", Proc. of the 23th Int'1 Conf. on Software Engineering. Toronto: IEEE Computer Press, 2001, pp: 665-667.

[3] C. Ghezzi, B. Nuseibeh. "Guest Editorial: Introductio to the Special Section", IEEE Trans. on Software Engineering, 25(6), pp: 782-785, 1999.

[4] S. Easterbrook, M. Chechik, "Int'1 Workshop on Living with Inconsistency", Proc. of the 23th Int'1 Conf. on Software Engineering. Toronto: IEEE Computer Press, 2001, pp: 749-750.

[5] K. Boness, A. Finkelstein, R. Harrison, "A Lightweight Technique for Assessing Risks in Requirements Analysis", IET Software, 2(1), pp: 46-47, 2008.

[6] P. Brescinani, P. Giorgini, F. Giunchiglia, et al, "Tropos: An Agent-Oriented Software Development Methodology", Joural of Autonomous Agents and Multi-agent Systems, 8(3), pp: 203-236, 2004.

[7] A. Hunter, "Reasoning with Contradictory Information Using Quasi-Classical Logic", Journal of Logic and Computation, 10(5), pp: 677-703, 2000.

[8] J. Grant, A. Hunter, "Measuring Inconsistency in Knowledgebases", Journal of Intelligent Information Systems, 27(2), pp: 159-184, 2006.

[9] A. Hunter, "Paraconsistent Logics", In Hadbook of Defeasible Reasoning ad Uncertainty Managemet , Kluwer, 1998.

[10] J. Grant and V. S. Subrahmanian, "Applications of Paraconsistency in Data and Knowledge Bases", Synthese, 2000, 125: 121-132.

[11] K. Q. He, R. Peng, J. Wang, et al, "Network Software", Science Press, China, 2008.

[12] Y. Liu, J. Z. Wu, J. Zhao, et al, "Decomposition Model Building and Otology Reasoning toward G-Layer of RGPS Requirement Meta-Model", 2010 3th Iternational Conference on Advance Computer Theory and Engineering, Institute of Electrical and Electronics Engineers, Chengdu, China, 2010, Vol.2, pp: 50-54.

[13] Y. Liu, J. Z. Wu, "Formal Verfication of RGPS-G", the International Conference on Computer Science and Serivce System CSSS2011, Nanjing, China, 2011, Vol.4, pp: 3248-3251..