International Journal of u- and e- Service, Science and Technology
Vol.8, No. 11 (2015), pp.1-10
http://dx.doi.org/10.14257/ijunesst.2015.8.11.01

Capturing Software Requirements Modeling from Ontology Tree

Yaqing Liu[1], Jinghuan Guo[1] and Yong Liu[2]
[1]*School of Information Science & Technology, Dalian Maritime University, Dalian 116026, China*
[2]*Artificial Intelligence Key Laboratory of Sichuan Province, Sichuan University of Science and Engineering, Zigong 643000, China*
liuyaqing@dlmu.edu.cn

Abstract

Requirements modeling is an important activity in software engineering and is more basic than other activities. But for scale domain, Requirements modeling usually suffers from limited knowledge of developers. In order to get a complete requirements model, software engineers have used domain ontology to capture requirements. However, previous approaches only can automatically capture requirements and the final software system is difficult to maintain. In this paper, we decompose old domain ontology into some sub ontologies and organize them into a tree model, which is called as Ontology Tree. Accordingly, we generate requirements model from Ontology Tree rather than old ontology. From real application, our approach shows a better development and maintainability of software.

Keywords*: Ontology; Ontology Tree; Requirements Modeling; Concept Lattice*

1. Introduction

In software engineering, the main phases of developing a software system usually include four parts: requirements modeling, system analyzing, system designing and system implementing [8]. Requirements modeling aims to get domain model, which is premise of system analysis. But for scale business domain, requirements modeling usually suffers from limited knowledge of developers so that final domain model is usually incomplete. Namely, domain model can't reveal the real activities well. Ontology is explicit specification of conceptualization of some domain and aims to make data sharable and reuse. So In order to get a complete domain model, software engineers have used domain ontology to capture requirements [1, 2]. These approaches based on ontology showed a good improvement for capturing complete requirements. But an unavoidable problem is that final software system is still hard to maintain. An obvious fact is representation model of domain ontology dominates quality of software [7]. So it becomes a key problem how to get a good representation model of domain ontology. In this paper, we represent domain ontology as a tree model, which is called as Ontology Tree. Accordingly, domain model is built based on Ontology Tree. From real application, our approach shows a better development and maintainability of software.

This paper is organized as follows. The whole framework for building a requirements model is given in Section 2. And algorithm of decomposing domain ontology is introduced in Section 3. We describe how a domain model obtained from ontology tree model in Section 4. A case study is given in Section 5. The related works are mentioned in Section 6 and conclusion and the next work are arranged in the last section.

2. Whole Framework

As you see from Figure 1, we obtain requirements model through two steps. The first step is responsible to decompose an initial domain ontology into an ontology tree, which consists of some sub ontologies. In second step, the requirements model is built from ontology tree. Next, we will elaborate the framework.

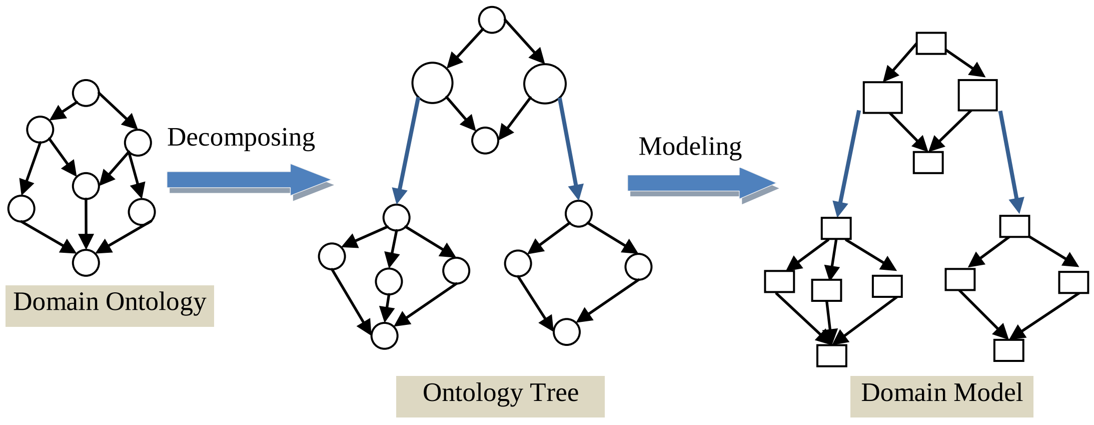

Figure 1. The Whole Framework for Building a Domain Model

3. Ontology Decomposition

3.1. Ontology and Concept Lattice

Definition 1. An **ontology** $\mathbb{O}$ is defined as a 5-tuple: $\mathbb{O}=\{C, P, R, A, I\}$, where:

- C is the set of classes, which represents various entities in some domain being modeled. We assume that classes are named by one or more natural language terms and are normally referenced within the ontology by a unique identifier.
- P is a set of properties. For example, for class "*country*", "*have population*" is a property.
- R is the set of relationships between classes. For instance, *synonym of*, *kind of*, *part of*, *instance of*, *property of*. But in this paper, only *kind of* relationship is considered.
- A is a set of axioms, usually formalized into some logic language. These axioms specify additional constraints on the ontology and can be used in ontology consistency checking and for inferring new knowledge from the ontology through some inference mechanism.
- I is set of instances.

Although it may get a more complete domain model to apply a domain ontology to requirements modeling, hugeness of domain ontology give rise to higher degree of coupling, which will make it difficult to develop and maintain software. In order to get a domain model which has a lower degree of coupling, domain ontology needs be decomposed.

3.2. Concept Lattice

As it is mentioned [17], ontology is high similar to formal concept lattice in structure. And formal concept analysis is often applied to ontology construction. So there is an inherent mapping between ontology and concept lattice. The mapping rule is as follow.

- Classes are mapped to formal concepts.
- Instances are mapped to objects.
- Properties are mapped to attributes.

- *Synonym of* relationship between classes is mapped to inheritance between formal concepts.

By the high similarity between ontology and concept lattice, we applied algorithm of evaluating sub-lattice from a concept lattice to decompose initial ontology.

Definition 2. A **formal context** FT is defined as a 3-tuple: $FT=(O, A, R)$, where

- O is a set of formal objects.
- A is a set of formal attributes.
- $R \subseteq O \times A$. For any $o \in O$ and $a \in A$, $(o, a) \in R$ holds iff object o has attribute a.

Definition 3. Given a formal context $FT=(O, A, R)$ and $X \subseteq B$, $Y \subseteq A$. $X'=\{a \in A | (b, a) \in R, \forall b \in X\}$ is said to be the **common attributes** of X. $Y'=\{b \in B | (b, a) \in R, \forall a \in Y\}$ is said to be the **common objects** of Y.

Definition 4. Given a formal context $FT=(O, A, R)$. A 2-tuple $fc=(X, Y)$ is said to be a **formal concept** from FT iff $Y=X'$ and $X=Y'$ hold.

Definition 5. Given two formal concepts (X_1, Y_1) and (X_2, Y_2) of a formal context (O, A, R), (X_1, Y_1) is said to be **superconcept** of (X_2, Y_2) and (X_2, Y_2) is said to be **subconcept** of (X_1, Y_1) if $X_2 \subseteq X_1$ or $Y_1 \subseteq Y_2$ holds. The relationship is represented as $(X_2, Y_2) \preccurlyeq (X_1, Y_1)$.

Definition 6. Given a formal context (O, A, R), consider the set of all formal concepts of this context, indicated as $FC(O, A, R)$. Then $\mathbb{L}=(FC(O, A, R), \preccurlyeq)$ is a complete lattice called **concept lattice**.

For example, consider a formal context called *European Cities* where O={*Athens*, *Courmayeur*, *Innsbruck*, *London*, *Paris*, *Reykjavik*, *Rome*}, A={*Ska_Area*, *Beach*, *Lake*, *Euro*, *River*, *Skiing_Area*} and R is showed in Table 1 [17]. Concept lattice from the formal context shown by Table 1 is shown in Figure 2. *Ska_Area*, *Beach*, *Lake*, *Euro*, *River* and *Skiing_Area* are abbreviated to *Ska*, *Bea*, *Lak*, *Eur*, *Riv* and *Ski* respectively. Also, *Athens*, *Courmayeur*, *Innsbruck*, *London*, *Paris*, *Reykjavik* and *Rome* are abbreviated to *A*, *C*, *I*, *L*, *P*, *Re* and *Ro* respectively.

Table 1. The *European Cities* Context

	Skating_Area	*Beach*	*Lake*	*Euro*	*River*	*Skiing_Area*
Atehns	×	×	×	×		
Courmayeur				×		×
Innsbruck				×	×	×
London			×		×	
Paris			×	×	×	
Reykjavik			×			×
Rome	×	×	×	×	×	

3.3. Decomposition Algorithm

Definition 7. Given two formal context (O_1, A_1, R_1) and (O_2, A_2, R_2), $\mathbb{L}_1=(FC(O_1, A_1, R_1), \preccurlyeq)$ is said to be a sub concept lattice of $\mathbb{L}_2=(FC(O_2, A_2, R_2), \preccurlyeq)$ iff $A_1 \subseteq A_2$ and $R_1=\{(o,a)|(o,a) \in R_2$ and $a \in A_2 - A_1\}$ hold.

For example, let the formal context be (O_1, A_1, R_1) shown in Table 1 and let the formal context be (O_2, A_2, R_2) shown in Table 2. $\mathbb{L}_2=(FC(O_2, A_2, R_2), \preccurlyeq)$ is said to be a sub-concept of $\mathbb{L}_1=(FC(O_1, A_1, R_1), \preccurlyeq)$.

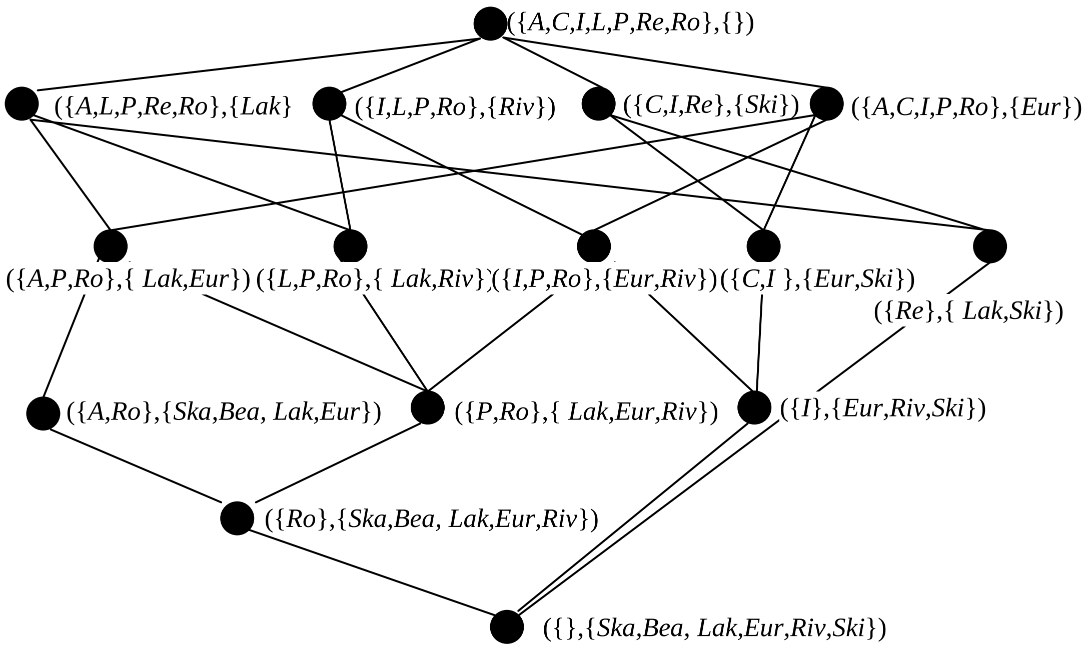

Figure 2. The Formal Concept Lattice from Formal Context Shown in Table 1

Table 2. The *European Cities* Context without Attributes of *Ska_Area* and *Beach*

	Lake	**Euro**	**River**	**Skiing_Area**
Atehns	×	×		
Courmayeur		×		×
Innsbruck		×	×	×
London	×		×	
Paris	×	×	×	
Reykjavik	×			×
Rome	×	×	×	

In order to decrease scale of concept lattice but not lose information, we propose a concept lattice decomposition algorithm based on semantic similarity of attributes.

From Table 1, it can be found easily that attribute *River* is close to attributes *Beach* and *Lake* in semantic. Attribute *Skating_Area* is close to attribute *Skiing_Area*. So it can be deduced that not of all attributes are same in semantic distance. By semantic distance, we may decompose formal context and fulfill concept lattice decomposition.

3.3.1. Attributes Clustering Based on Semantic Similarity

Formula 1 is used to evaluate similarity $as(a,b)$ between two words a, b in Wordnet[18]. From formula 1, it is noted that,

- The similarity between two words (a,b) is the function of their distance and the lowest common subsume $lso(a,b)$.
- If the $lso(a,b)$ is root, $depth(lso(a,b))=1, as(a,b)>0$; if the two words have the same sense, the word a, word b and $lso(a,b)$ are the same node. $len(a,b)=0$. $as(a,b)=1$; otherwise $0<depth(lso(a,b))<deep\_max$, $0<len(a,b)< 2*deep\_max$, $0<as(a,b)< 1$. Thus, the values of $as(a,b)$ are in (0, 1].

$$as(a,b) = \frac{2 * depth\left(lso(a,b)\right)}{len(a,b) + 2 * depth\left(lso(a,b)\right)} \quad \text{Formula (1)}$$

Algorithm 1 is used to cluster a group of attributes. {*Ska_Area*, *Beach*, *Lake*, *Euro*, *River*, *Skiing_Area*} can be clustered into three groups *All*={*WinterSports*, *Water*, *Eur*}, *WinnerSports*={*Skating_Area*, *Skiing_Area*}, *Wate*r={*Lake*, *River*, *Beach*}, where *All*, *WinnerSports* and *Water* are cluster name.

```
Algorithm 1:ACSS($A$)
Input: $A$, a group of attributes
Output: $\mathcal{Root}$
1.  FOR any two $a_1$ and $a_2$ in $A$
2.    IF $\forall b_1,b_2 \in A$, $as(a_1,a_2)>as(b_1,b_2)$ AND $\neg(a_1=b_2 \wedge a_2=b_1)$ AND $\neg(a_1=b_1 \wedge a_2=b_2)$
3.      delete $a_1$ and $a_2$ from $A$;
4.      $t \leftarrow lso(a,b)$;        Add $t$ to $A$;
5.      children($a_1$,$t$);      children($a_2$,$t$);//set $a_1$ and $a_2$ as children of $t$.
6.  IF $A$={$a$}   //There is only one element $a$ in $A$.
7.    $\mathcal{Root} \leftarrow a$
8.  RETURN $\mathcal{Root}$
```

3.3.2. Formal Context Decomposition

Algorithm 2 is used to decompose formal context. The decomposed formal context from Table 1 is shown in Table 3.1. In Figure 3.1, a concept lattice is generated from the formal context shown Table 3.1. Limited to page area, objects of every formal concept is replaced by the format "#*n*". *#1*, *#2*, *#3*, *#4*, *#5*, *#6*, #7 and *#8* are {*A*, *C*, *I*, *L*, *P*, *Re*, *Ro*}, {*A*, *C*, *I*, *P*, *Ro*}, {*A*, *C*, *I*, *Re*, *Ro*},{*A*, *I*, *L*, *P*, *Re*, *Ro*},{*A*, *C*, *I*, *Ro*},{*A*, *I*, *P*, *Ro*},{*A*, *I*, *Rey*, *Ro*} and {*A*, *I*, *Ro*}. Because formal concept ({*A*, *I*, *L*, *P*, *Re*, *Ro*},{*Wat*}) is top one including compound attribute *Wat*, a sub concept lattice from formal context shown in Table 3.2 is associated to attribute *Wat* is shown in Figure 3.2. Because formal concept ({*A*, *C*, *I*, *Re*, *Ro*},{*Win*}) is top one including compound attribute *Win*, a sub concept lattice from formal context shown in Table 3.3 is associated to attribute *Wat* is shown in Figure 3.3.

We can compare denseness of the concept lattice shown in Figure 3 to the old one shown in Figure 2 by evaluating the number of dependence relation. We regard a pair of super-sub classes as a dependence relation. The number of dependence relation shown Figure 2 is 26 and the number of dependence relation shown Figure 3 is 23 because the numbers of dependence relation shown in Figures 3.1, 3.2 and 3.3 are 12, 7 and 4, respectively. The denseness of new concept is smaller than the old one.

```
Algorithm 2: FCD($FC$, $r$)
Input:  $FC$, formal context $FC$=($O$, $A$, $R$)
        $r$, ACSS($A$)
Output: $\mathcal{FCs}$, a set of formal contexts.
1.  $A^* \leftarrow$set($r$);  //evaluate set of attributes.
2.  $R^*=\{(o,a)|(o,a)\in R$ and $a \in A^*\}$
3.  $\mathcal{FCs} \leftarrow \mathcal{FCs} \cup \{(O, A^*, R^*)\}$;
4.  IF $r$ has children
5.    For every child $c$ of $r$
6.      FCD($FC$, $c$);
7.  RETURN $\mathcal{Root}$
```

4. Ontology Decomposition and Domain Modeling

By algorithm 2 and the mapping rule between ontology and concept lattice, a domain ontology can be decomposed in a ontology tree. And according to ontology tree, a domain model is generated by followed mapping rule.

- Classes in ontology are mapped to classes in domain model.
- Instances are mapped to entities.
- Properties in ontology are mapped to properties in domain model.
- *synonym of* relationship between classes is mapped to inheritance between classes.

Table 3.1. Top Formal Context of *European Cities*

	WinnerSports	*Water*	*Euro*
Atehns	×	×	×
Courmayeur	×		×
Innsbruck	×	×	×
London		×	
Paris		×	×
Reykjavik	×	×	
Rome	×	×	×

Table 3.2. Sub-formal Context of *European Cities*

	Lake	*River*	*Beach*
Atehns	×		×
Innsbruck		×	
London	×	×	
Paris	×	×	
Reykjavik	×		
Rome	×	×	×

Table 3.3. Sub-formal Context of European Cities

	Skatting_Area	*Skiing_Area*
Atehns	×	
Courmayeur		×
Innsbruck		×
Reykjavik		×
Rome	×	

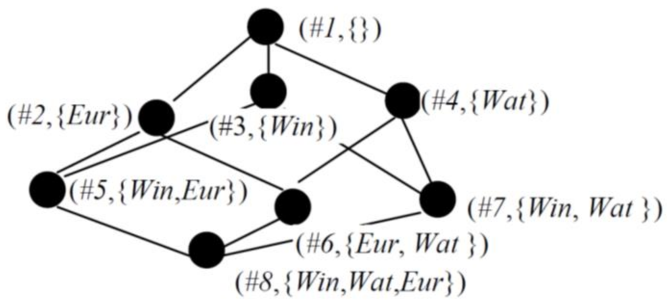

Figure 3.1. The Formal Concept Lattice from Table 3.1

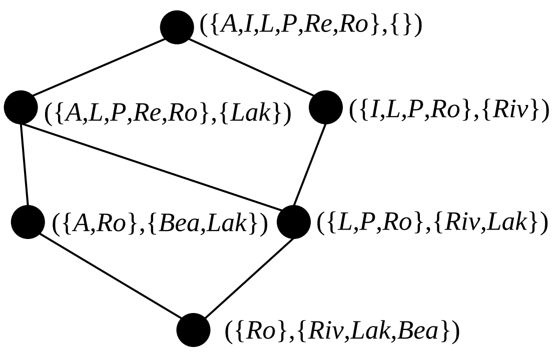

Figure 3.2. The Formal Concept Lattice from Table 3.2

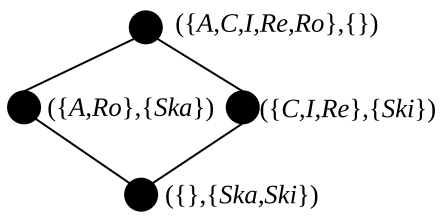

Figure 3.3. The Formal Concept Lattice from Table 3.3

5. A Study Case

We applied our approach to a restaurant management information system for a hotel. We state our approach by domain modeling and model maintaining.

An initial hotel ontology is provided by staff and managers. The ontology contained 78 classes and there are average 6 properties for a class. There are 128 dependence relations. After we apply our approach to the ontology, the ontology is simplified. 21 sub ontologies and 110 dependence relations are obtained.

More important, the top domain ontology contains only 15 classes, which is effective to avoid being drowned by trivial details for top managers and is easier to interactive between top managers and requirements model staff. Also, bottom ontologies are more detailed, which is more suitable to hotel staff. For maintenance of requirement model, part to be changed is easier to be located accurately and the influence scope is easier to be evaluated according to the mapping between requirement model and ontologies.

6. Related Work

Requirements modeling is an important and basic activity in software engineering. Three kinds of approaches are paid more attentions. They are Goal-Oriented approach [3], Intention-Oritented approach [4] and approach based on problem framework [5].

Goal-Oriented approach is actuated by final goal. Requirements engineers decompose, refine and abstract final goal step by step. After the course is done iteratively, an and-or tree model is obtained. Related concepts can be extracted from an and-or tree. KAOS is a famous implemention of the approach[6].

Intention-Oritented approach underline contribution of every attendance. Every attendance's activity is thinked as independent requirements modeling. And then requirements information from attendances, including all kinds of dependences related to goals, tasks, resources and soft goals, are reconciled and intergrated and reach to a complete requirements model. A standard implement is i^* modeling framework[9].

Approach based on problem frame is a promising one in recent year. Interaction between reality and software is emphasized. The analysis for structure of interaction is a main activity in requirements modeling. Some effective problem frameworks such as Context Effect frame, Small Problem Frames, User Interaction Frame, *etc.*, are proposed [10]. Also, two methods for solving problems are given. The first is based on problem decomposion. Some decomposion stragies such as structure-oritented decomposion[11], goal-oritented decomposion[12] and decomposion based on ontology[13], are put forward. Approach based on problem frame has been applied to some fields successfully. Problem frame can provide requriements specification in eXtreme Program. Problem frame is used to describe Web service[14]. It is used to model web software development[15] and distributed system[16] in such fields as e-commerce, insruance, *etc*.

7. Conclusion

In this paper, we propose an effective approach to requirements modeling. We decompose domain ontology into some sub ontologies and organize these sub-ontologies in manner of tree. By ontology tree, requirement model can be built well. From real application, the tree model representation for ontology can support development and maintenance of software better.

Acknowledgements

We thank all reviews for their useful comments to improve the paper. This work was supported by the Fundamental Research Funds for the Central Universities(3132015043), National Natural Science Foundation of China(61203283), Liaoning Provincial Natural Science Foundation of China(2014025004), The Open Project Program of Artificial Intelligence Key Laboratory of Sichuan Province (2013RZJ02, 2012RYJ02).

References

[1] X. Chen, B. Yin and Z. Jin, "Ontology-Guided Requirements Modeling Based on Problem Frames Approach", Journal of Software, vol. 22, no. 2, **(2011)**, pp. 177-194.
[2] Z. Jin and L. Liu "Towards Automatic Problem Decomposition: An Ontology-based Approach", Proceedings of the 2nd International Workshop on Advances and Applications of Problem Frames, **(2006)**, pp. 41−48.
[3] A. V. Lamsweerde, "Goal-Oriented Requirements Engineering: A Guided Tour", Proceedings of the 5th IEEE International Symposium on Requirements Engineering, **(2001)**, pp. 249−263.
[4] E. Yu, "Towards modeling and reasoning support for early-phase requirements engineering", Proceedings of the 3rd IEEE ,International Symposium On Requirements Engineering, **(1997)**, pp. 226−235.
[5] M. Jackson, "Why Software Writing is Difficult and will Remain So", Information Processing Letters, vol. 88, no. 1-2, **(2003)**, pp. 13−15.
[6] A. Dardenne, A. V. Lamsweerde and S. Fickas, "Goal-Directed Requirements Acquisition", Science of Computer Programming, vol. 20, no.1-2, **(1993)**, pp. 3−50.
[7] M. Fowler, K. Beck, J. Brant, W. Opdyke and D. Roberts, "Refactoring: Improving the Design of Existing Code", **(2002)**.
[8] L. Tokuda and D. Batory, "Automating Three Modes of Evolution for Object-oriented Software Architecture", Proceedings of the 5th conference on object oriented technologies and Systems, **(2009)**, pp. 189-202.
[9] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia and M. J. Tropos, "An Agent-oriented Software Development Methodology", Autonomous Agents and Multi-Agent Systems, vol. 8, no. 3, **(2004)**, pp. 203−236.
[10] J. G. Hall and L. Rapanotti, "Problem Frames for Socio-technical Systems", Proceedings of Human Computer Interaction: Concepts, Methodologies, Tools, and Applications. Hershey: Information Science Reference, **(2009)**, pp. 713−731.
[11] L. Rapanotti, J. G. Hall, M. Jackson and B. Nuseibeh, "Architecture Driven Problem Decomposition", Proceedings of the 12th IEEE, International Requirements Engineering Conference, **(2004)**, pp. 73−82.
[12] S. Bleistein, K. Cox and J. Verner, "Validating Strategic Alignment of Organizational IT Requirements Using Goal Modeling and Problem Diagrams", The Journal of Systems and Software, vol. 79, no. 2, **(2006)**, pp. 362−378.
[13] Z. Jin and L. Liu, "Towards Automatic Problem Decomposition: An Ontology-based Approach", Proceedings of the 2nd International Workshop on Advances and Applications of Problem Frames, **(2006)**, pp. 41−48.
[14] G. J. Cai, Z. Jin and Z. J. Feng, "A method for Web Service Description by Using Problem Frames Approach", Proceedings of the 3rd International Workshop on Applications and Advances of Problem Frames, **(2008)**, pp. 23−28.
[15] S. Jeary and K. Phalp, "On the Applicability of Problem Frames to Web-based business Applications", Proceedings of the 1st International Workshop on Applications and Advances of Problem Frames, **(2004)**, pp. 35−38.
[16] C. Haley, "Using Problem Frames with Distributed Architectures: A Case for Cardinality on Interfaces", Proceedings of the 2nd International Workshop from Software Requirements to Architectures, **(2003)**, pp. 130−133.

[17] A. Formica, "Ontology-based Concept Similarity in Formal Concept Analysis", Information Sciences, vol. 176, no. 18, **(2006)**, pp. 2624-2641.
[18] L. Meng, R. Huang and J .A. Gu, "Review of Semantic Similarity Measures in WordNet", International Journal of Hybrid Information Technology, vol. 6, no. 1, **(2013)**, pp. 1-10.

Authors

Yaqing Liu, he is an associate professor, he received the Ph D degree from Dalian Maritime University in 2011. The main research directions: Semantic Web, Ontology.

Jinghuan Guo, she is an associate professor, she received the Ph D degree from Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences in 2002. The main research directions: Semantic Web, Ontology.

Yong Liu, he is a lecturer, he received the bachelor degree from in Sichuan University of Science & Engineering in 2003. The main research directions: Semantic Web, Ontology, Artificial Intelligence.