An Ontology-Based Framework of Requirements Evolvement Management

Hongyue He
Institute of Command Automation, PLA University of Science and Technology, Nanjing, Jiangsu, China
Email: hehy2008@sina.com

Zhixue Wang, Ying Zhang and Weizhong Zhang
Institute of Command Automation, PLA University of Science and Technology, Nanjing, Jiangsu, China
Email: {wzxcx@163.com, zhywl66@163.com, willson_zwz@163.com}

***Abstract*—an ontology-based framework of Requirements Evolvement Management (REM) is proposed for controlling requirements evolvement and tracing requirements which are the important aspects of REM. The ontology is introduced to describe the information about requirements evolvement. Consistent analysis and influence analysis for requirements evolvement is researched in the process of REM. The inconsistency of requirements is verified through the manipulations of ontology, and different inconsistency is dealt in different ways. A layered dependence tree of requirements is built according to the dependent relationship between requirements to support the layered managements of requirements and an equation of cost analysis is defined to count the priority of requirements which are in the same layer.**

***Index Terms*—ontology, requirements evolvement, consistent analysis, influence analysis**

I. INTRODUCTION

As the requirement is the important aspect in software development and requirements elicitation is the beginning of the process of software development, the quality of requirements influences the process of software development and decides the quality of software. All the software developers want that requirements are well elicited and unchanged in the process of software development. But requirements are often evolved in the process of software development, from the design to maintenance, because of several kinds of reasons [1]. According to the investigation of Christof, 73% requirements are evolved in 15 projects of Alcatel, and more then 30% requirements are evolved in a two years project[2][3]. If requirements evolvement isn't managed correctly and normatively, the schedule and cost of software development will be influenced and the quality of software will not be as good as expectation. So the Requirement Evolvement Management (REM) has been an issue in the Requirements Engineering (RE) [4].

He Hongyue PhD. Tel: +86−25−80824566-8020.
E-mail: hehy2008@sina.com

To deal with requirements evolvement, several models for software development process, such as Agile Model, Spiral Model and CMMI, are developed. Because of the differentiations between the cognitions of requirements evolvement, different models focus on different aspects of requirement evolvement and they manage requirements evolvement in different ways. Some tools for requirements management, such as CaliberRM, RequisitePro, ClearQuest and DOORS, have been used in practical project. But they don't support to control requirements evolvement and trace requirement [5]. Especially, requirements evolvement will produce the inconsistency of requirements; however, the verification of these inconsistencies isn't supported by these tools.

Lu and Jin introduce the ontology into RE and propose an ontology-based method for domain modeling [6][7]. Their method introduces Knowledge Engineering into the process of RE and improve the quality of model of software requirement, practicability and automatization of the process of RE. As the ontology has a good capability of expressing and sharing knowledge, more and more researches use ontology in RE [8].

In this paper, an ontology-based framework of REM is proposed. According to the framework, the ontology describes the information about the requirement evolvement; the consistency and influence analysis of requirements evolvement is discussed by the manipulation of ontology; the requirements evolvement can be traced by the ontology. The requirements evolvement will be implemented orderly according to a layered dependence tree and priority. The paper is organized as follows. The concept model of the framework is built in the section Ⅱ. In section III, the information about requirements evolvement is transformed into OWL DL ontology. The consistency and influence analysis of REM is discussed in section Ⅳ. The related works will be presented in section Ⅴ.

doi:10.4304/jsw.7.9.2018-2025

II. CONCEPT MODEL OF THE FRAMEWORK

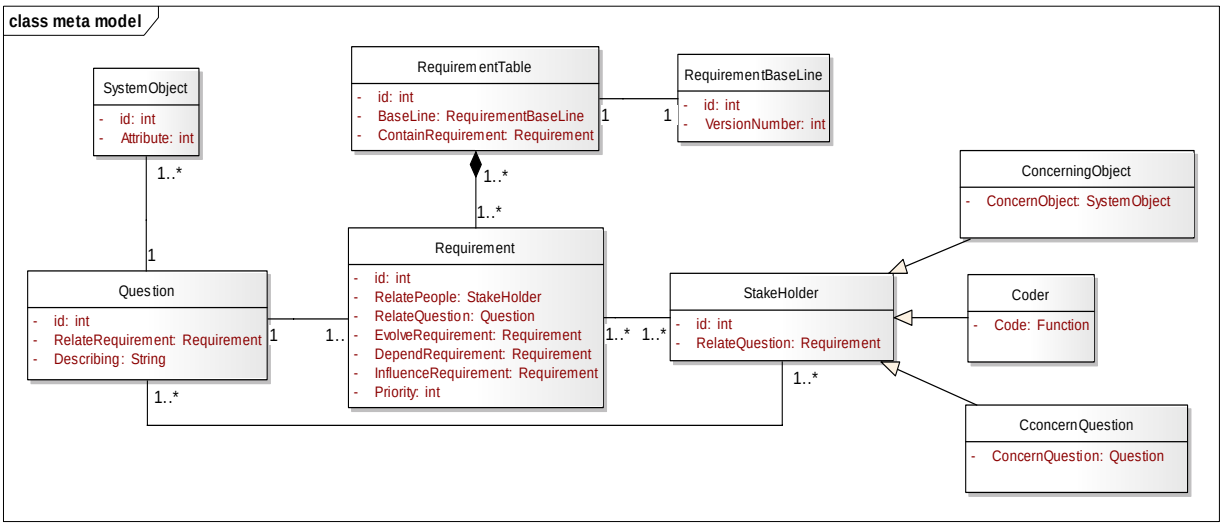

Figure 1. Concept model of REM

To well control the risk of requirements evolvement and implement the requirements evolvement, a good and effective process of REM is needed. Fig. 1 describes an ontology-based framework of REM which is referenced to the process of REM designed by Leffingwel [9]. The concept model focuses on *Requirement* and contains other concepts which are used in the process of REM such as *Question*, *StakeHolder*, *SystemObject*, *RequirementTable* and *RequirementBaseline*. These concepts are defined and interpreted as follow:

Definition 1: Requirement can be defined as a tuple (id, q, P, r, DR, IR, p) where

- *q is the question which this requirement focuses on*
- *P is the set of stakeholders who concern this requirement;*
- *r is a requirement which is evolved to this requirement;*
- *DR is the set of requirements which this requirement depends on;*
- *IR is the set of requirements which this requirement influences;*
- *p is the priority of this requirement;*

A requirement must focus on some question and several stakeholders, such as client, designer and coder, will concern it. There are three kinds of relationship between requirements, a requirement depends on other requirements; the evolvement of a requirement can influence other requirements; a requirement is evolved from other requirement. Because of these relationships, the priority of requirement is important in the process of dealing with several requirements evolvement.

Definition 2: A question can be defined as a tuple (id, R, d) where

- *R is the set of requirements which focus on this question;*
- *d is the string which describes this question;*

A question describes what the client concerns, and it can be related to some requirements. Besides client, a question is concerned by other stakeholders such as designer and coder.

Definition 3: A systemobject can be defined as a tuple (id, A) where

- *A is the set of attributes of this systemobject;*

A systemobject is an object which is described in some question. A question may describe several systemobjects.

Definition 4: A stakeholder can be defined as a tuple (id, R) where

- *R is the set of requirements which this stakeholder concerns;*

A stakeholder is people who directly or indirectly concern some requirements. A stakeholder may concern several requirements and there are three kinds of stakeholders, the client concerns the questions; the designer concerns the objects of the system; the coder concerns the code of the system.

Definition 5: A requirementtable can be defined as a tuple (id, R, b) where

- *R is the set of requirements which are contained in this requirementtable;*
- *b is the baseline which flags this requirementtable;*

Definition 6: A requirementbaseline can be defined as a tuple (id, v) where

- *v is the version number of the requirementtalbe flaged by this requirementbaseline;*

A requirementtable is a set of requirements which are in the same phase of software development and the requirementbaseline records the version number of this requirementtable.

Base on the above analysis, the information about requirements evolvement in one phase of software development process can be defined as follow:

Definition 7: The Information can be defined as a tuple (r, Q, S, O) where

- *r is the requirementtable in this phase;*
- *Q is the set of questions in this phase;*
- *S is the set of stakeholders in this phase;*
- *O is the set of systemobjects in this phase;*

III. FORMALIZATION OF CONCEPT MODEL

In this framework, ontology describes the information about requirements evolvement in the process of REM. The information belongs to one concept in the concept model in section Ⅱ and it can be understood as an instance of the concept. Base on the above analysis, an ontology-based method of formalization of concept model and information is proposed. The concept model is transformed into the T-Box which is a set of axioms of the ontology; the information about requirements evolvement is transformed into the A-Box containing the assertions of the ontology. To transform the concept model into T-Box, the classes and their relationships in concept model must be elicited firstly. TABLE I is the list of classes which are elicited from the concept model and TABLE Ⅱ is the list of relations which are also elicited from the concept model.

TABLE I.
THE LIST OF CLASSES IN T-BOX

description of class	superclass	definition of class	constraint
requirement	N/A	Requirement	Requirement ? •
evolved requirement	requirement	RequirementNew	RequirementNew ? Requirement
requirement table	N/A	RequirementTtable	RequirementTtable ? •
requirement baseline	N/A	RequirementBase	RequirementBase ? •
question	N/A	Question	Question ? •
system object	N/A	SystemObject	SystemObject ? •
stakeholder	N/A	StakeHolder	StakeHolder ? •
client	stakeholder	Client	Client ? StakeHolder
designer	stakeholder	Designer	Designer ? StakeHolder
coder	stakeholder	Coder	Coder ? StakeHolder
code	N/A	Code	Code ? •
function	N/A	Function	Function ? •

TABLE II.
THE LIST OF RELATIONS IN T-BOX

description of relation	definition of relation	domain	constraint in domain	range	constraint in range
A requirement table contains some requirements	Contain	RequirementTalbe	1	Requirement	1..*
A requirement is evolved from other requirement	Evolve	Requirement	1	Requirement	1
A requirement depends on other requirement	Depend	Requirement	1	Requirement	1..*
A requirement influences other requirement	Influence	Requirement	1..*	Requirement	1..*
A stakeholder concerns some requirements	Concern	StakeHolder	1..*	Requirement	1..*
A question describes some system objects	Associate	Question	1	SystemObject	1..*
A baseline flags some requirement table	Flag	RequirementBase	1	RequirementTable	1
A requirement focuses some question	Settle	Requirement	1..*	Question	1
A stakeholder concerns some questions	Consider	StakeHolder	1..*	Question	1..*
A designer codes some codes	Coding	Designer	1	Code	1..*
A segment of code comprises some functions	Comprise	Code	1..*	Function	1..*

To transform the information about requirements evolvement into OWL DL ontology, an algorithm – Build-Ontology is designed as follow:

Algorithm: Build-Ontology
Input: TABLE I, TABLE Ⅱ, Information
Output: OWL DL ontology
begin
for all *c* in column of "definition of class" in TABLE I, **do**
Add the expression of *c* and its constraint into the OWL DL ontology;
end for;
for all $r = (c_1, c_2)$ in TABLE Ⅱ, the relation $r^{\sim}$ is the inverse relation of *r*, **do**
Add the expression of $c_1 \hat{o} \ \forall r.c_2$ and $c_2 \hat{o} \ \forall r^{\sim}.c_1$ into the OWL DL ontology;
end for;
for all the multiplicity constraint in range of every relation $r = (c_1, c_2)$ in TABLE Ⅱ, **do**
if multiplicity constraint is "1" **then**
Add the expression of $c_1 \hat{o} \leq 1r.c_2$ and $c_1 \hat{o} \geq 1r.c_2$ into the OWL DL ontology;
else if multiplicity constraint is "1..*" **then**

Add the expression of $c_1 \hat{o} \geq 1r.c_2$ into the OWL DL ontology;
end if;
end for;
for all the multiplicity constraint in domain of every relation $r = (c_1, c_2)$ in TABLE Ⅱ, the relation $r^{\sim}$ is the inverse relation of r, **do**
if multiplicity constraint is "1" **then**
Add the expression of $c_2 \hat{o} \leq 1r^{\sim}.c_1$ and $c_2 \hat{o} \geq 1r^{\sim}.c_1$ into the OWL DL ontology;
else if multiplicity constraint is "1..*" **then**
Add the expression of $c_2 \hat{o} \geq 1r^{\sim}.c_1$ into the OWL DL ontology;
end if;
end for;
for all elements in Information, **do**
Add the expression of the assertion describing that the element belongs to one class into the OWL DL ontology;
end for;
for all couples of elements in Information, **do**
Add the expression of the assertion describing that the couple of elements belongs to one relation into the OWL DL ontology;
end for;
return OWL DL ontology;
end

Ⅳ. PROCESS OF REM

In the process of REM, there are mainly four steps to manage the requirements evolvement [5], as follow:

1 Analyze the influence of requirements evolvement;
2 Discuss and confirm the requirements evolvement;
3 Implement the requirements evolvement;
4 Modify related productions;

But these steps don't deal with the consistency analysis which is the important aspect of requirements analysis in system of systems (SOS) [10]. Especially the requirements evolvement may produce inconsistency. Fig.2 show a process of REM focusing on the consistency analysis and influence analysis, which is referenced to the related work [5][9]. The whole process is divided into four steps as follow:

1 Request. A requirement evolvement is requested by one stakeholder.
2 Consistency analysis. Verify the inconsistency between requirements and then deal with this inconsistency in a suitable way.
3 Influence analysis. Analyze the dependent relationship between requirement and the cost for implementing requirements evolvement to decide the sequence of implementation of requirements evolvement.
4 Implement. Implement the requirements evolvement.

In this paper, the Consistency analysis and Influence analysis are mainly discussed because of the limitation of space. When the request of requirements evolvement is produced, the consistency analysis will be done. If the requirement evolvement produces the inconsistency which will make a fatal mistake in the software, the request should be denied; otherwise the request will be accepted. According to the result of consistency analysis, if the request is denied, the whole process is over; if the request is accepted, influence analysis will be done, and then the requirements evolvement will be implemented.

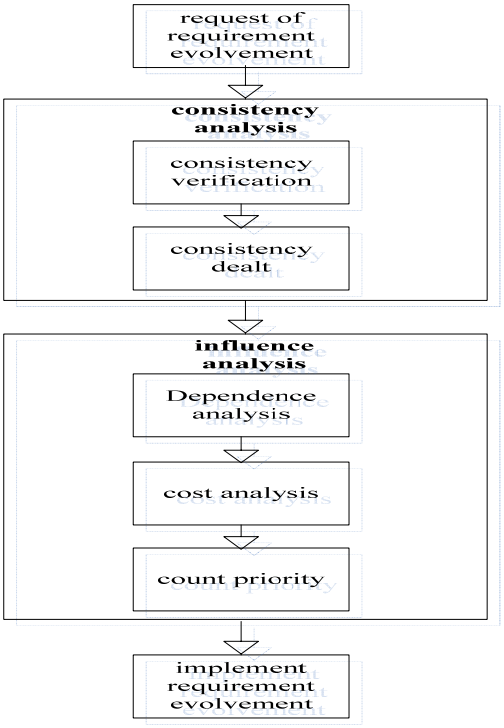

Figure 2. Process of REM

A. Consistency Analysis

In the realm of requirements consistency analysis, the representative approach is rules-based requirements consistency management which uses a set of consistent rules to verify the inconsistency of requirements [11]. Several kinds of consistent rules are defined, and they can be used from requirement elicitation to software coding [12]. In the process of REM, two kinds of consistent rules are defined as follow:

1 Identity rule: in the same requirement baseline, the attribute of object which is described by two requirements must be identical.
2 Authorization rule: in the different requirement baseline, the stakeholders of requirement must be same.

The identity rule verifies the inconsistency that the attribute of object described by two requirements are different. For example, in the development of individual online bank system, the business man wants to increase the sum money of one deal to 10000 to attract more clients; but to ensure the safety of client's money, the safety man wants to limit the sum money of one deal to 5000, that can reduce the loss of client when the information of client is stole. Then, the sum money of one deal is in a inconsistent state. The authorization rule verifies the inconsistency that one stakeholder modifies the requirement which isn't authorized to him. When the initial requirements are elicited, these requirements are

authorized to different stakeholders. In the process of REM, one stakeholder can modify these requirements authorized to him and can't modify those which are not authorized to him.

As the information about requirements evolvement is transformed into the OWL DL ontology in section III, the verification of inconsistencies based on the identity rule and authorization rule can be implemented through the reasoning and querying of the ontology. The inconsistency based on the identity rule is described as the contradiction between these axioms in ontology, and it can be implemented through the reasoning of ontology. The inconsistency based on authorization rule is described as the integrality of assertions in ontology, and it can be implemented through the querying of ontology. The querying clauses in SPARQL are as follow:

```
SELECT ?a ?b ?c ?d
WHERE
    {?a xmlns:Evolve ?b.
    ?c xmlns: Concern ?a.
    ?d xmlns: Concern ?b.
    ?c owl: differentfrom ?d.}
```

Because of the different reasons for these inconsistencies, the influences of them on software development are different, so they will be dealt in different ways. There are three ways to deal with these inconsistencies, as follow:

Resolve

This kind of way focuses on the inconsistency based on identity rule. When the descriptions of object concerned by different stakeholders are different, if these descriptions are conflictive, that means there isn't an object satisfying these descriptions. This inconsistency produced by these descriptions is not tolerant and must be resolved because it can make the software disabled. The related request of requirements evolvement must be rejected and the influence of them is cancelled.

Tolerate

This kind of way also focuses on the inconsistency based on identity rule. When the descriptions of object by different stakeholders are different, if these descriptions are not conflictive, that means there is some object satisfying these descriptions. This inconsistency produced by these descriptions is tolerant and a warning is sent to these stakeholders. The related request of requirements evolvement is accepted and the influence of them will be implemented.

Negotiate

This kind of way focuses on the inconsistency based on the authorization rule. There are two reasons for stakeholder modifying the requirements which are not authorized to him, one is accidental; the other is that the primary stakeholder left this project and the related requirements are authorized to the new stakeholder. To find out the reason for this inconsistency, we must negotiate with stakeholders. If the reason for this inconsistency is former, the request of this requirements evolvement is rejected. Otherwise, the request of this requirements evolvement is accepted.

B. Influence Analysis

As the requirements evolvement will indirectly influence the cost of software development and the functionality of software, they should be dealt carefully in the process of REM. The influence should be analyzed carefully and then requirements evolvement is managed according to the result of influence analysis. Referenced to the layered management of requirements [9], the low layered requirements should be dealt earlier than the high one because the low one influences the high one, the requirements are layered by the dependence between them and the low layered requirements will be dealt early. According to the result of cost analysis which focuses on the cost of implementing the requirements evolvement, the priority of requirements evolvement in the same layer is counted.

Dependence Analysis

The requirements are not independent with each other, and there is a dependent relationship between them which describes that the evolvement of one requirement will influence the other requirement. The dependence between requirements is a part of the tracing information of requirements. There are three kinds of popular method for managing the tracing information [13], i.e. tracing table, tracing list and automated tracing link. Mohammad et al.[14] use ontology to build a tracing information repository which records the tracing information of requirements. In section III, the *Depend* and *Influence* relationships of requirements have been defined, and these information have also been transformed into the OWL DL ontology which is a repository containing the tracing information of requirements. The following manipulations describe that how can draw and analyze the tracing information from OWL DL ontology.

TABLE III.
TRACING TABLE

	R_1	R_2	R_3	R_4	R_5	R_6
R_1			*	*		
R_2						*
R_3		*			*	
R_4		*			*	
R_5						*
R_6						

TABLE III describes the tracing information between six requirements, the sign * expresses that the requirement in this line depends on the requirement in this column. This tracing information is transformed into OWL DL ontology using *Depend* and *Influence*. This information in OWL DL ontology is described as the set of assertions, as follow:

Depend (R_1, R_3), Depend (R_1, R_4),
Depend (R_2, R_6), Depend (R_3, R_2),
Depend (R_3, R_5), Depend (R_4, R_2),
Depend (R_4, R_5), Depend (R_5, R_6),
Influence (R_3, R_1), Influence (R_4, R_1),
Influence (R_6, R_2), Influence (R_2 R_3),
Influence (R_5, R_3), Influence (R_2, R_4),
Influence (R_5, R_4), Influence (R_6, R_5),

The tracing list of six requirements can be drawn through the querying of ontology, the querying clause in SPARQL is as follow:

```
SELECT ?a ?b
WHERE
    {?a xmlns:Depend ?b.
    ?a rdf: type xmlns:RequirementNew.}
```

TABLE Ⅳ describes the tracing list which is drawn according to the result of above query .

TABLE IV.
TRACING LIST

Requirement	Dependence	Requirement	Dependence
R_1	R_3, R_4	R_4	R_2, R_5
R_2	R_6	R_5	R_6
R_3	R_2, R_5		

To implement the layered management, a layered dependence tree which describes the *Depend* relationship between requirements is needed. An algorithm – Coustrut-layeredtree is designed to construct this tree.

Algorithm：Construct-layeredtree

Input：TABLE Ⅳ

Output：Layered dependence tree

Step1：construct a layer for the tree and put the requirement which is in the column of "Requirement" and not in the column of "Dependence" in the layer, then delete the line which the requirement is in.

Step2：

Step2.1 if the table is not empty, construct a new layer in the bottom of tree and put the requirement which is in the column of "Requirement" and not in the column of "Dependence" in the new layer, then delete the line which the requirement is in, go to **Step2**.

Step2.2 if the table is empty, return the tree.

Fig. 3 is a tree which is constructed by the Coustrut-layeredtree.

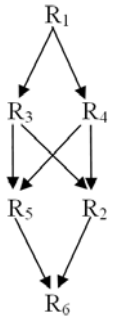

Figure 3. Layered dependence tree

According to the layered dependence tree, the low layer requirement will be implemented earlier than the high layer requirement. If the requirements are in the same layer, the cost analysis will be implemented to count the priority of them.

Cost Analysis

The implementation of requirements evolvement will need stakeholders to do related works, such as redesigning the system object and modifying the code of software. These all kinds of resources expended by these work are the cost of requirements evolvement. Because of the differentiation in capabilities of stakeholders and conditions of software development, it is difficult to develop a general method to count the cost. Provided that the capabilities of stakeholders are same and the conditions of software development are familiar, an equation for cost analysis of requirements evolvement is defined as follow:

$$W(r) = \alpha \cdot P(r) + \beta \cdot O(r) + \chi \cdot C(r)$$

- *r is the requirement which is analyzed;*
- *Function P(r) counts the number of stakeholders of r;*
- *Function O(r) counts the number of system objects which are redesigned;*
- *Function C(r) counts the number of code which will be modified;*
- α, β *and* χ *are the weight of these functions;*

A requirement has three kinds of related stakeholders, but the function *P(r)* counts the number of designers and coders which are related to *r*. The function *P(r)* can be implemented through the querying of ontology. The querying clause in SPARQL is as follow:

```
SELECT ?a
WHERE
    {{?a xmlns:Concern r.
     ?a rdf: type xmlns:Designer.}
     UNION
    {?a xmlns:Concern r.
     ?a rdf: type xmlns:Coder.}}
```

The result of above query is the set of designers and coders related to *r*, and the cardinality of the set is the result of function *P(r)*.

A requirement focuses on one question which may describe several system objects. The function *O(r)* counts the number of these objects, and it can be also implemented through the querying of ontology. The querying clause in SPARQL is as follow:

```
SELECT ?a
WHERE
    {r xmlns:Settle ?b.
     ?b xmlns:Associate ?a.}
```

The result of above query is the set of system objects related to *r*, and the cardinality of the set is the result of function *O(r)*.

As the code of software comprises some functions in programming language, the function in the code is used as a unit to count the code. The ontology contains the information about the functions of code, so *C(r)* can be also implemented through the querying of ontology. The querying clause in SPARQL is as follow:

```
SELECT ?a
WHERE
    {?b xmlns:Concern r.
     ?b xmlns:Coding ?c.
     ?c xmlns:Comprise ?a.}
```

The result of above query is the set of functions which are in the code related to *r*, and the cardinality of the set is the result of function *C(r)*.

As different software development process models focus on the different aspects which are involved in the

equation, the value of α, β and χ can be set according to the relevant software development process model. The priority of requirements is set according to the result of the equation. If the result is high, it means that the implementation of this requirement evolvement needs more works, and then the requirement will be dealt earlier.

According to the following rules, the sequence of implementation of requirements evolvement in the process of REM can be set.

1 In the layered dependence tree, the low layered requirement will be implemented earlier than the high one;

2 In the same layer, the requirement which has a high result of cost analysis will be implemented earlier than the one with a low result.

Ⅴ. Related Work

As requirements change and the information between them must be tracked throughout the process of software development[15], The requirements management should be treat as a process. There are some researches on the requirements engineering[16], and several commercial tools are available.

Some recommendations on requirements management for software system are proposed in Guide to the Software Engineering Body of Knowledge[17]. The basic concepts and general guidelines for requirements management are defined, and a detailed description of the phases of iterative requirements management process is proposed. Youngki et al.[18] focus the traceability of requirements evolvement, they propose a approach for evolving traceability links from requirements to system components.

In some researches, the technologies of meta-model and UML profile are used to manage requirements. Kabanda et al.[19] define a requirements meta-model framework for software system to enhance product adoption, which focuses the social features: users, policies and culture. Ramesh et al [20] interview 26 software development organizations and then present models for requirements management focusing on requirements traceability. Zhu et al[21] propose UML profiles for modeling design decisions and non-functional requirements in a general way. The traceability between design decisions and architectural elements is maintained, and the inconsistency over this traceability is checked. Pardillo et al[22] present a meta-model connecting goals, requirements and measures. To provide the meta-model with a familiar notation, a UML profile based on the i* framework is proposed, which facilitates this approach in the context of UML-based software engineering process. Arpinen et al[23] define general concepts and abstract interfaces for managing requirements between requirements management and system development, and present a general meta-model for requirements management focusing on software and embedded system domains. The UML profile is used to facilitate the meta-model to UML tools in practice.

As the ontology can represent and share the knowledge, it is also used to manage requirements in software process. Roy et al[24] present a ontology-based framework for requirements management. The requirements are constructed by ontology to provide a shared conceptualization of knowledge which is needed for the specification of a product. Mohammad et al[14] use ontology to build a requirement traceability repository in order to achieve inbuilt requirements management and to conduct requirement analysis at the Computational Independent Model level.

There are some tools for requirements management in practice[25]. Borland's CaliberRM supports mainly aspect of requirement management except products traceability and communions. The IBM rational develops two requirements management tools: RequisitePro and ClearQuest. RequisitePro is a general requirements management tool. ClearQuest is a requirements evolvement tool, though it is not specifically designed for requirement evolvement. The two tools are often working together. But these tools don't support the consistency analysis and influence analysis in the process of REM.

Ⅵ. Conclusion

An ontology-based framework of REM which is independent of software development process model is proposed, and it focuses on the consistency analysis and influence analysis of requirements evolvement. An ontology contains the information about requirements evolvement is built and the REM can be implemented through the manipulation of ontology. First, a concept model of the framework is built to describe the concepts and their relationships in the framework. Then, applying the concept model, the information about the requirements evolvement is transformed into the OWL DL ontology. The inconsistencies can be verified through the reasoning and querying of ontology, and they are dealt in different ways according to their taxonomy. To implement the layered management of requirements, the algorithm – Construct-layered tree is designed to construct a layered dependence tree of requirements. Finally, an equation of cost analysis is defined to count the priority of requirements in the same layer.

Acknowledgment

This work was supported by the National Defense Advanced Research Program of China under the project number 51306010202.

References

[1] Norman F Schneidewind. "Investigation of the risk to software reliability and maintainability of requirements changes", *17th IEEE International Conference on Software Maintenance*, Florence. November, 2001.pp:127-136

[2] Christof Ebert. "Understanding the product life cycle: Four key requirements engineering techniques", *IEEE Software*, 2006, 23(3).pp:19-25.

[3] Jozef De Man, "Christof Ebert. Requirements uncertainty: influencing factors and concrete improvements", *27th*

International Conference on Software Engineering, Missouri, May, 2005.pp:553-560.
[4] Anthony Finkelstein, Jeff Kramer. "Software engineering: a roadmap", *In Proc. Of The Future of Software Engineering,* April, 2000.pp:3-22.
[5] Qin Zhongsen, Li Juan. "Precess of requirements change management and analysis of requirements managements tools", *Computer Engineering and Design,* 2009,30(11).pp:601-1605.
[6] Jin Zhi. "Ontology-Based Requirements Elicitation", *Chinese Journal of Computers,* 2000,23(5).pp:486-492.
[7] Lu Ruqian, Jin Zhi. "Beyond the Knowledge Engineering ", *Journal of Computer Science and Technology,* 2006, 21(5).pp:790-799.
[8] Henderson-Sellers, B. "Bridging metamodels and ontologies in software engineering", *The Journal of Systems and Software,* 2011.pp:301-313.
[9] Leffingwel D, Widrig D. "Managing Software Requirements: A Use Case Approach", Addison Wesley, 2003.
[10] Zhu Xuefeng, Jin Zhi. "Managing the Inconsistency of Software Requirements ", *Chinese Journal of Software,* 2005, 16(7).pp:1221-1231.
[11] Nuseibeh B, Easterbrook S, Russo A. "Leveraging inconsistency in software development ", *IEEE Computer,* 2000, 33(4).pp:24-29.
[12] Spanoudakis G, Zisman A. "Inconsistency management in software engineering: Survey and open research issues", In: Chang SK, ed. *Handbook of Software Engineering and Knowledge Engineering. Singapore*: World Scientific Publishing Co., 2001. pp:329-380.
[13] Jin Zhi, Liu Lin and Jin Ying. "*Software Requirement Engineering: Theory and Method*", Beijing: Science Publication, 2008.
[14] Mohammad Y., Venera Z. and Moteb Al B. . "Requirements Analysis and Traceability at CIM Level", *Journal Software Engineering and Applications*,2010,pp:845-851.
[15] Halbleib H. ."Requirements management", *Information System Management* 2004,21.pp:8-14.
[16] Stefan W. and Jens von P.. "A survey of traceability in requirements engineering and model-driven development", *Software and Systems Modeling*,2010,9(4),pp:529-565.
[17] Abran A., Moore W. Bourque P. et al. "Guide to the Software Engineering Body of Knowledge", Los Alamitos, CA: IEEE Computer Society, 2004.
[18] Youngki H. Minho K. and Sang-Woong L. "Requirements Management Tool with Evolving Traceability for Heterogeneous Artifacts in the Entire Life Cycle", *2010 Eighth ACIS International Conference on Software Engineering Research Management and Applications*, Canada, May 2010, pp:248-255.
[19] Kabanda S.K., Adigun M., and Chani T. "A requirements metamodel framework for enhancing product adoption", *In Proceedings of the South African Telecommunications Networks and Applications Conference*, Mauritius, September 2007.
[20] Ramesh B. and Jarke M. ."Toward reference models for requirements traceability", *IEEE Transaction on Software Engineering,* 2001,27.pp:59-93.
[21] Zhu L. and Gorton I.. "UML profiles for design decisions and non-functional requirements", *In Proceedings of the Workshop on SHAring and Reusing Architectural Knowledge Architecture, Rationale, and Design Intent*, Washington, 2007.pp:8-15.
[22] Pardillo J., Molina F., Cachero C. et al. "A UML profile for modeling measurable requirements", *In Proceedings of the Advances in Conceptual Modeling Challenges and Opportunities, Lecture Notes in Computer Science*,2008.pp:123-132.
[23] Arpinen T., Timo D.H. and Marko H.. "Meta-Model and UML Profile for Requirements Management of Software and Embedded Systems", EURASIP *Journal on Embedded Systems*.vol.2011, 14 pages.2011
[24] Roy R., Kerr C. and Corbett J.. "Design requirements management using an ontological framework", *Cirp Annals-Manufacturing Technology.* 2005;54.pp:109-112
[25] Zainol A. and Mansoor S.. "An Investigation of a Requirements Management Tool elements",*2011 IEEE Conference on Open Systems(ICOS2011)*,Langkawi, September,2011.pp:53-58

Hongyue He was born in 1985. He is a PhD student of Institute of Command and Automation, PLA University of Science and Technology University, where he received B.S. and M.S. degrees. His research interests are requirements engineering, focusing on specification and management.

Zhixue Wang was born in 1961.He is a professor of Institute of Command and Automation, PLA University of Science and Technology. His research interests are software engineering, requirements engineering, theory and technology of command automation, currently focusing on domain-specific modeling and formal verification. He received B.S. degree from Hefei Polytechnic University, M.S. degree from National University of Defense and Technology, and used to be a visiting researcher in Faculty of Information Technology, University of Brighton, England.

Ying Zhang was born in 1982. She is a PhD student of Institute of Command and Automation, PLA University of Science and Technology University, where he received M.S. degrees. His research interests are requirements engineering, focusing on specification and formal verification.

Weizhong Zhang was born in 1983. He is a PhD student of Institute of Command and Automation, PLA University of Science and Technology University, where he received B.S. and M.S. degrees. His research interests are requirements engineering, focusing on behavioral-modeling and formal verification.