Semantic-based Requirements Analysis and Verification

Haibo Hu, Lei Zhang
School of Software Engineering
Chongqing University
Chongqing, China
{hbhu, l.zhang}(at)cqu.edu.cn

Chunxiao Ye
College of Computer Science
Chongqing University
Chongqing, China
yecx(at)cqu.edu.cn

Abstract—**Requirements verification for complex software system has become more and more important in requirements engineering and it is one of the most helpful strategies for improving the quality of software system. Related work shows that requirement elicitation and analysis can be facilitated by semantic technologies and domain ontology. A new methodology for verifying software requirements with structural and formal semantics based on domain ontology is proposed in this paper. Requirement specification processed with natural language can be decomposed into a set of atomic requirement items which are represented with triplet of semantic elements in their domain ontology. Mapping semantic requirement items to concepts and their relationships in the domain ontology leads to a set of inference rules that are represented by description logics to verify the completeness, consistency and correctness of requirement specifications. The process of requirements verification is defined in order to evaluate the quality of requirement specifications, and its effectiveness is analyzed and evaluated with an experimental case study.**

Keywords- atomic requirement item; domain ontology; description logics; formal modeling; requirements verification

I. Introduction

With the increasing complexity of software systems and expansibility in its application areas, requirement engineering (RE) plays a more and more important role in the whole Software Development Life Cycle [1, 2]. RE did not become research focus until the end of the last century although software engineering has developed since forty years ago [3]. The American Standish Group's professional research shows: the factor related with software requirement accounts for about 50 percents in all factors leading to failures or problems in the software project [4], which indicates the importance of RE.

In the process of the requirements analysis, natural languages with powerful expressional ability are usually used to describe requirements. Besides, some semi-formal requirements analysis methods such as use-cases [5] and scenarios [6, 7] are also employed to support requirements elicitation and analysis. However, in practice, the stakeholders from different domains may participate in requirement process commonly, and then non-formal or semi-formal language's inherent ambiguity can lead developers and domain users to have different meanings about domain knowledge and requirements, and exert negative impact on requirements quality. The formal method based requirements analysis uses strict mathematics language to describe requirements and eliminate ambiguity, and effectively verify the completeness, consistency and correctness of the requirements [8]. Nevertheless, it is very difficult for domain users to understand formal methods owing to specialty. Besides, most software requirements analysts are lack of domain knowledge and end-users are ambiguous about system requirements, which often leads to an ambiguous and incomplete requirements specification and repetitive work after realizing the system. Therefore, it is very important that all participators have common understanding about shared domain concepts and knowledge, so as to ensure the completeness, consistency and correctness of the requirements elicitation.

With the development of knowledge engineering, the representation and reuse of domain knowledge are attracting more attention. A new requirement verification method based on domain knowledge and structured semantic description is proposed in this paper, by using domain ontology as objects association and knowledge representation method, decomposing the original requirements described in natural language into a number of well-structured atomic requirement item, formally describing requirements in ontology language easily understood by developers and domain users, matching atomic requirement item with the semantic knowledge expressed in domain ontology by using mapping rules. The Verification of the requirements completeness, consistency and correctness is also discussed, by making a set of inference rules which are represented by description logics. As the ontology language has formal semantics and allows inference to its semantic elements, "intrinsic experience" and "tacit experience"[9], that can be introduced to verify completeness, consistency and correctness of the requirements.

II. Semantic Technology Based on Ontology

In recent years, Ontologies with powerful ability of semantic expression and Ontological Engineering (OE) are attracting wide attention in Knowledge Engineering (KE) and related application areas. Ontology is derived from a philosophical concept, which can be used to describe the essence of all kinds of things.

In KE, the general definition of Ontology is: an ontology is a formal specification of a shared conceptualization [10]. In general, this definition contains four means [11]:

①conceptualization: the abstract model of phenomenon of objective world; ②explicit: concepts and relationships among them are defined explicitly; ③formal: strict mathematical description; ④share: the knowledge that ontology contains is recognized by its users.

Ontology commits itself to elicit the knowledge of related domains, provides common understanding of the knowledge and determines common terms recognized in this area; and give a clear definition about the interrelationship between these terms from different levels of formal model.

The ontology's applications have expanded into areas of Software Engineering (SE)[12], Natural Language Processing [13], Information Retrieval [14], Database[15],etc. Particularly, it is applied to the whole process of SDLC in SE. As a kind of more precise, more canonical normative and formal language, ontology language can be used to describe requirement specification and express requirement knowledge formally and accurately [16, 17] in the stage of requirements analysis.

On the basis of domain knowledge sharing, ontology language can make full use of domain models that only contain indispensable terms of the domain knowledge to describe requirements formally by eliminating professional terms. Moreover, ontology is more advantageous to build up domain models and describe requirements than transitional knowledge representation methods because ontology language can be well understood by computers. The application of ontology has made an effective progress in the field of knowledge-sharing and requirements elicitation and analysis, but formal verification of the requirements is still a vital, urgent problem need to be addressed in the field of software engineering.

III. Ontologies Based Requirement Description

A. Modeling requirement analysis with domain ontology

At requirements analysis phase, requirements description can be simplified by breaking the original requirement into a number of related atomic requirement items, which helps to describe and verify the requirements accurately. Requirement decomposition is derived from the V-model for system engineering development and design[18]. An atomic requirement item can be mapped to a requirement description sentence that describes some atomic concepts and atomic relations in ontology system. Decomposition process should comply with the following principles:

① Decomposed requirements are semantically consistent with the original requirement; ② there is no redundant requirement item after decomposition.

The result of requirements decomposition can be directly applied. On the basis of KE and domain ontology, a new formal requirements analysis conversion model based on ontology is proposed in this paper as shown in Figure 1.

The above modeling framework can be described in the following steps:

Step 1: Breaking the original requirements into a set of atomic requirement items that can describe minimum problems. This is a self-check process, and will run until meeting the principles of requirements decomposition.

Step 2: Mapping atomic requirement items to semantic elements of the domain ontology. This is also a self-check process using inference rules. This means the requirements must be verified according to the requirement verifying model proposed in the following part of this paper. If requirements verification requirements can not be met, return to modify the original requirements, and skip to step 1.

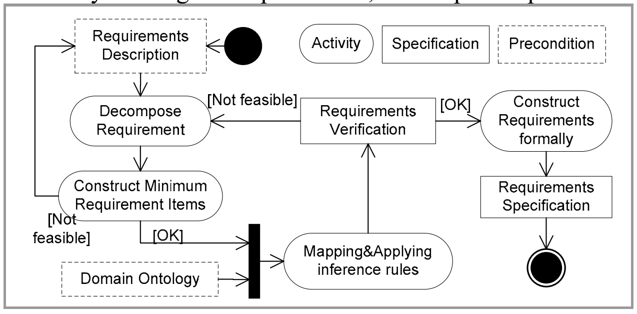
Requirements Description
Activity
Specification
Precondition
Decompose Requirement
[Not feasible]
Requirements Verification
[OK]
Construct Requirements formally
Construct Minimum Requirement Items
Requirements Specification
[Not feasible]
[OK]
Mapping&Applying inference rules
Domain Ontology

Figure 1. Framework of requirement analysis based-on ontology.

Step 3: Formally describing requirements gained from step 1 and 2 with the requirements representation method of software model.

Step 4: Composing formal requirement description into requirements specification.

This paper proposes a new formal method based on ontology to describe requirements, which can verify and check the completeness, consistency and correctness of the requirements through domain ontology model and inference rules.

B. Modeling Domain Ontology Concepts

A domain ontology system consists of a set of semantic elements, such as concepts, relationships between concepts and inference rules. The method of requirements elicitation based on domain ontology had been applied widely in the field of requirement engineering. Lu and Jin et al. have applied the domain ontology of enterprise organization structure to automatic requirements acquisition [16, 21]. Haruhiko et al. has proposed to apply the knowledge of domain ontology to the process of requirement elicitation [19]. On the basis of former research achievement, this paper bends itself to verify the requirements through describing requirements specification structurally and formally, and build up an ontology mapping and inference rules in section 3.3. The formal definition of domain ontology and atomic requirement item are as follows:

Definition 1: Domain ontology: D (C, R, A^R, X) represents domain ontology. It is a set of instances composed of related knowledge. These instances can be represented by a set of triplets, where,

C: Set of concepts within domain knowledge;

R: Set of relationships among concepts within domain knowledge;

A^R: Set of attributes of relationships;

X: Set of axioms. It is permanent assertion within domain knowledge.

Definition 2: Atomic requirement item: **r** represents a assured atomic requirement description in the set of requirements. We can use lots of triples to describe domain ontology after its concepts have been mapped to requirement terms. That's *r* (*s, p, o*) or *p*(*s, o*).

s$\in$***C***, Subject concept of atomic requirement item;

o$\in$***C***, Object concept of atomic requirement item;

p$\in$*R*, Binary relationship of subject concept and object concept of atomic requirement item.

We divide A^R into two parts: set *V* and *A*. So, *p* (*s, o*) can be represented as *v (s, o)* and *a (s, o)* respectively.

Set *V* can be represented as a set: {is-a, has-a, synonymy, contradict}, Where is-a represents a kind of relationship of inheritance. For example, is-a (Member, Anyone) represents that the concept Member is sub-concept of the concept Anyone; has-a represents a kind of relationship of property, for example, has-a (BorrowedBook, ReturnDate) represents the concept BorrowedBook has a property named ReturnDate, synonymy represents a kind of relationship of "synonymy", that is, associated concepts represent the same semantic mean, for example, synonymy(Member, Leaguer), contradict represents a kind of relationship of "contradict", that is, associated concepts have inconsistent semantic meanings, such as, contradict (AvailabledBook, BorrowedBook).

Set *A* is also a set of associated relationships, which represents interactive operation of concepts, for example, borrow (Member, Book) denotes that Member has a function of borrowing Book. Therefore, decomposed original requirement description can be mapped to ontology elements. That is:

Requirements ≡ {R_1, R_2, …, R_n},

$\forall R_i = \{r_1, r_2, \ldots, r_n\} = \{p_1, p_2, \ldots, p_n\}$.

The process of mapping atomic requirement items to ontology model is shown in Figure 2.

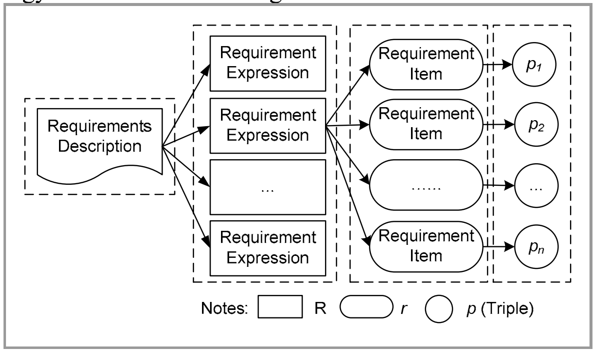

Figure 2. Mapping requirement description to domain ontology

It is intuitionistic to express semantic means of ontology using triples. The triples emphasize phraseological criterion and restriction criterion of concepts representation, as well as associated criterion among concepts. However, the semantic means of concepts themselves are handled by domain users. The following part of this paper explains formal representation of ontology system by an example named Borrowing Books as shown in Figure 3.

C. Mapping and Inference Rules

The mapping relation from atomic requirement item *r* to semantic elements of ontology system can be represented as $r \rightarrow \{C, R\}$.

It is represented as $F_m(r)$ in this paper and is named as mapping function. For example, *r*–"Anyone can read book" is mapped to {{Anyone,Book},read} in Figure 4, which can be as read(Anyone,Book) in the form of triple.

We introduce symbol *D(c)*/*D(p)* to describe the inference rules more conveniently, which represents that instance concept *c* or binary relation *p* is used to describe requirement *r* in requirement specification *D*.

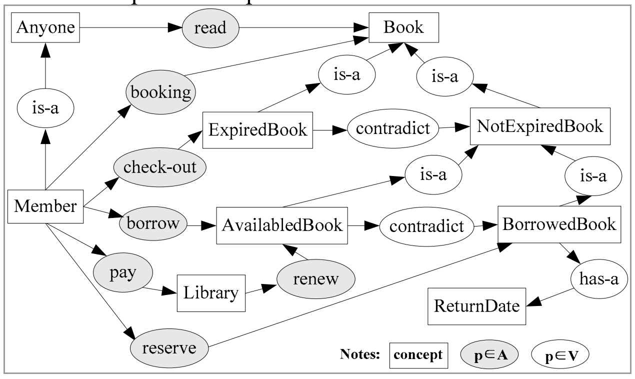

Figure 3. The domain ontology system of "Borrowing Book" module

We use decidable Description Logics as inference mechanism. Firstly, we give the definition of *p*(*s, o*):

Definition 3: *Functional, F*(*p*): Represents *o* is single-valued function of *s*, i.e. borrow(Member,AvailabledBook).

Definition 4: *Transitive, T*(*p*): $p(x,y) \sqcap p(y,z) \Rightarrow p(x,z)$.

Definition 5: *Symmetric,* S(*p*): $p(x, y) \Rightarrow p(y, x)$,

Notes: $x \in C, y \in C, z \in C, p \in$ R.

In the field of DL**s**, it exists relationship of *is-a* among relations besides concepts, for example, *is-a (p1,p2)* represents that p_1 is sub-relation of p_2. Therefore, some inference rules can be defined as follows:

Rule 1: is-a(x, y) $\sqcap$ *p(y, z)* $\Rightarrow$ *p(x, z)*. Note: $x \in C, y \in C, z \in C, p \in$ R. For example,

is-a(Member, Anyone) ⊓ read(Anyone, Book)
⇒ read(Member, Book)

Rule 2:is-a$(p_1,p_2) \sqcap p_2(x,y) \Rightarrow p_1(x,y)$. Note:$x \in C, y \in C, p_1 \in$ R, $p_2 \in$ R.

Rule 3: $\exists p \exists x \exists y \exists z |(\mathrm{F}(p) \sqcap p(x,y) \sqcap p(x, z)) \equiv$ *contradict(y, z)*. Note: $x \in C, y \in C, z \in C, p \in$ R.

It can be inferred in Rule 3 that *y* and *z* is inconsistent if there is *x*, *y*, *z* and relations between *x* and *y*, *x and z* under the precondition that *p* is *Functional*; and vice versa. The marking of *contradict* is given directly in Figure 3.

We can analyze requirement description from the semantic point of view, and then consummate it step by step through using $F_m(r)$ and a set of inference rules.

D. Ontology Based Requirement Verification

The method for verifying the requirements proposed in this paper is used to check the completeness, consistency and correctness of requirements specification through inference rules. Because relation instances have clear semantic means in set V, all triples proposed in this section belong to $a(s, o)$. For simplicity, we represent it as the form of $p(s, o)$.

Completeness Verification: The basic idea of verifying completeness is: checking all triples $p(s, o)$ of the ontology system. If there is no mapping relation between certain triple and requirement item, requirements specification is incomplete. It can be represented formally as follows:

Formula 1: $\forall p \exists s \exists o \mid (D(s) \sqcap p(s, o)) \Rightarrow (D(p) \sqcap D(o))$

Formula.1 denotes: For any $\boldsymbol{p}$, if there are concept $\boldsymbol{s}$ and $\boldsymbol{o}$ and relation $\boldsymbol{p}$ between $\boldsymbol{s}$ and $\boldsymbol{o}$, and $\boldsymbol{s}$ is described in requirement specification, then p and o also should be introduced in requirement specification, otherwise, requirement specification is incomplete.

Consistency Verification: The basic idea of verifying consistency is: tracking all binary relations p mapped from requirement to ontology system. If there are the same two or more relation p and subject s, for example, $p(s, o_1)$, $p(s, o_2)$, $p(s, o_3)$…, and two or more objects exist relation of *contradict* among o_1, o_2, o_3…, for example, contradict (o_1, o_2), then it represents requirement specification is not consistent [17, 22]. It can be represented formally as follows:

Formula 2: $\exists p \exists x \exists y \exists z \mid (p(x,y) \sqcap p(x,z) \sqcap contradict(y,z) \sqcap D(x) \sqcap D(y) \sqcap D(z)) = \text{true}$

Formula.2 denotes: If there are relation $\boldsymbol{p}$ and concept $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$, and $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$ are described in requirement specification. And there is relation $\boldsymbol{p}$ both between $\boldsymbol{x}$ and $\boldsymbol{y}$, $\boldsymbol{x}$ and $\boldsymbol{z}$ respectively, and $\boldsymbol{y}$ is *contradict* with $\boldsymbol{z}$, then it can be deduced that requirement specification is not consistent.

Correctness Verification: The basic idea of verifying correctness is: tracking all requirement items in requirement specification. If some requirement item(s) can not be mapped to one or more triple $\boldsymbol{p(s, o)}$, then it represents that requirement specification is not correct. It can be represented formally as follows:

Formula 3: $\exists p \exists s \exists o \mid (p(s,o) \sqcap D(p) \sqcap D(s) \sqcap D(o)) = \text{false}$

Formula 3 denotes: If some requirement item(s) can not be mapped to one or more triple $p(s, o)$ in the domain ontology system, then it can be deduced that the requirement item is redundant, that is, the requirement specification is not correct.

According to all rules defined in this paper, we can verify completeness, consistency and correctness of requirement specification through matching atomic requirement item with the semantic elements. The whole process can be described as follows:

p_o: A binary relation instance of Set A in domain ontology system;
p_r: A binary relation instance after requirement item is mapped;
$p_o(s_o, o_o)$: A triple instance of Set A in domain ontology system;
$p_r(s_r, o_r)$: A triple instance after requirement item is mapped;
contradict(s_o, o_o): A triple instance that s_o and o is *contradict* in domain ontology system;
$P_o(S_o, O_o)$: A set of triples that their relation instances p_o are the same.

```
1: procedure Verify Completeness, Consistency, Correctness
2:          for each $p_o$ in $A$ do
3:        if $\exists p_r = p_o, p_r \in$ A then
4:          for each $p_o(s_o, o_o)$ in $P_o(S_o, O_o)$ do
5:              if $\exists p_r(s_r, o_r) = p_o(s_o, o_o)$ then
6:                    $p_o(s_o, o_o) \rightarrow$ Complete
7:              else
8:                    $p_o(s_o, o_o) \rightarrow \neg$Complete
9:              end if
10:         end for
11:         for each $contradict(s_o, o_o)$, $contradict \in$ V do
12:             if $p_o \in$ F($p$) and $\exists p_r(s_r, s_o) \wedge \exists p_r(s_r, o_o)$ = true, then
13:                   $p_r(s_r, s_o) \wedge p_r(s_r, o_o) \rightarrow$ Inconsistent
14:             else
15:                   $p_r(s_r, s_o) \wedge p_r(s_r, o_o) \rightarrow$ Consistent
16:             end if
17:         end for
18:     else
19:        $p_o(s_o, o_o) \rightarrow \neg$Complete
20:     end if
21:       end for
22:       for each $p_r(s_r, o_r)$ do
23:     if $\exists p_o(s_o, o_o) = p_r(s_r, o_r)$ then
24:        $p_r(s_r, o_r) \rightarrow$ Correct
25:     else
26:        $p_r(s_r, o_r) \rightarrow \neg$Correct
27:     end if
28:       end for
29: end procedure
```

After above verifying process, we can use following formulas [19] to evaluate the quality of requirement specification: $|R_{item}|$

Formula 4:*Completeness* $= |F_m| / |O|$

Formula 5:*Consistency* $= |\neg\text{contradict}(x,y) \wedge F_p| / |F_p|$

Formula 6:*Correctness* $= |F_{r\_item}| / |R_{item}|$

Where $|F_m|$ is the number of triple O(x,y) in ontology system that is mapped to requirement items; $|O|$ is the number of triples O(x,y) in ontology system; $|\neg\text{contradict}(x,y) \wedge Fp|$ is the number of atomic requirement items which can be mapped to ontology system and are not *contradict*; $|F_p|$ is the number of atomic requirement items which can be mapped to binary relation O in ontology system; $|F_{r\_item}|$ is the number of atomic requirement items which have been mapped to triples O(x, y) in ontology system; $|R_{item}|$ is the number of all atomic requirement items after requirement specification decomposition.

Our major target is to verify completeness, consistency and correctness. However, improving requirement specification is beyond the scope of the paper.

IV. CASE STUDY

The paper takes Book borrowing module of the library management system as a case study for analyzing the process and result of requirement verification.

In the case study, decomposed requirement items in requirement specification are mapped to semantic elements in domain ontology system. Then we can analyze requirement item semantically. The ontology system of case study is represented as Figure 4. The ontology can be represented with triples formally as follows:

```
is-a(Member, Anyone)
is-a(ExpiredBook, Book)
…
has-a(BorrowedBook, ReturnDate)
has-a(BorrowedBook, RenewDate)
contradict(ExpiredBook, NotExpiredBook)
contradict(AvailabledBook, BorrowedBook)
read(Anyone, Book)
pay(Member, Library)
…
```

The requirement items can be mapped to triples after original requirement has been decomposed, which is showed in Table I.

TABLE I. REQUIREMENT ITEMS AND SEMANTICS WITH TRIPLE TUPLE

#	Requirement Description	Present in Triple Tuple
1	Anyone can read book	read(Anyone, Book)
2	Only member can borrow book	borrow(Member, Book)
2.1	Member can borrow available book	borrow(Member, AvailabledBook)
2.1	Member can borrow borrowed book	borrow(Member, BorrowedBook)
3	Library can reserve available book	reserve(Library, AvailabledBook)
4	Member must check-out expired book	check-out(Member, ExpiredBook)
5	Member can booking book	booking(Member, Book)

The requirement items that have been mapped to domain ontology system are shown as Figure 4.

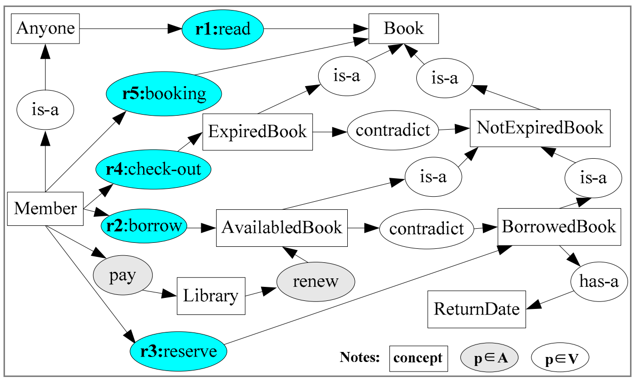

Figure 4. Mapping the semantic requirement items to domain ontology

The following conclusions can be drawn according to former rules and formulas. There is no mapping relation between requirement items in the requirement list and triple pay(Member, Library), renew(Member,BorrowedBook) in ontology system, so the requirement list is not complete.

There are binary relation *borrow* $\in F(p)$ and contradict(AvailabledBook,BorrowedBook). Therefore, the triple borrow(Member,BorrowedBook) and the triple borrow(Member, AvailabledBook) are not consistent. That is, *r2.1* and *r2.2* are not consistent.

There is no triple borrow(Member,BorrowedBook) in ontology system, which can be mapped by *r2.2*, therefore, the requirement list is not correct.

Requirement can be verified with the formulas presented in section 3.4. Note that *r2.1* and *r2.2* sub-items belonging to *r2* can be mapped to binary relation *borrow*, therefore, $|F_p|=6$, $|F_m|=5$, $|O|=7$, $|\neg contradict(x,y) \wedge F_p|=5$, $|F_{ritem}|=5$, $|R_{item}|=6$. Thus it can be calculated that,

1) Completeness = 5/7 = 71.4%;
2) Consistency = 5/6 = 83.3%;
3) Correctness = 5/6 = 83.3%.

ACKNOWLEDGMENT

The work is supported by the NSF Foundation of China under Grant No.60803027, and the Natural Science Foundation in Chongqing City of China under Grant No. CSTC 2008BB2312.

REFERENCES

[1] Lu Mei, Li Mingshu. Review of Methods and Tools of Software Requirements Engineering. Journal of Computer Research and Development, 1999, 36(11): 1289-1300(in Chinese)

[2] Zhang Jiazhong, Xu Jiafu. Advances in Requirements Engineering. Journal of Computer Research and Development, 1998, 35(1): 1-5(in Chinese)

[3] Shu Fengdi, Zhao Yuzhu, Wang Jizhe, et al. User-Driven Requirements Elicitation Method with the Support of Personalized Domain Knowledge. Journal of Computer Research and Development, 2007, 44(6): 1044-1052(in Chinese)

[4] Jin Zhi, Liu Lin, Jin Ying. Software Requirements Engineering: Principles and Methodology. Beijing: Science Press. 2008:4-5.

[5] Regnell B, Kimbler K, Wesslén A. Improving the use case driven approach to requirements engineering. Proceedings of the 2nd IEEE International Symposium on RE'95. New York: IEEE, 1995: 40-47.

[6] Damas C, Lambeau B, Dupont P, van Lamsweerde A. Generating Annotated Behavior Models from End-User Scenarios. IEEE Transaction on software Engineering, 2005, 31(12): 1056-1073.

[7] Zhang Yan, Hu Jun, Yu Xiaofeng, et al. Scenario-Driven Component Behavior Derivation. Journal of Software, 2007, 18(1): 50-61(in Chinese)

[8] Ta Weina, He Jifeng. Requirement Analysis Based on Formalized Method. Computer Engineering, 2003, 29(18): 107-108, 191(in Chinese)

[9] Ian K. Bray. An Introduction To Requirements Engineering. Translated by Shu Zhongmei, et al. Beijing: Posts & Telecom Press, 2003.

[10]Studer R, Benjamins V. R, Fensel D. Knowledge Engineering, Principles and Methods. Data and Knowledge Engineering, 1998, 25(1-2): 161-197.

[11]Deng Zhihong, Tang Shiwei, Zang Ming, et al. Overview of Ontology. Acta Scientiarum Naturalium Universitatis Pekinensis, 2002, 38(5): 730-738(in Chinese)

[12]Li Shanping, Yin Qiwei, Hu Yujie, et al. Overview of Researches on Ontology. Journal of Computer Research and Development, 2004, 41(7): 1041-1052 (in Chinese)

[13]Dominique E, Chris N, Andrew Z. Towards Ontology-based Natural Language Processing. Proceedings of the 4th Workshop on NLP and XML. Barcelona, 2004:59-66.

[14]Guarino N ,Masolo C, Vetere G. OntoSeek: Content-based Access to the Web. IEEE Intelligent Systems, 1999, 14(3):70-80.

[15]Lei Jiang, Topaloglou T, Borgida A, et al. Goal-Oriented Conceptual Database Design.Proceedings of the 15th International Conference on Requirements Engineering (RE'07). New York: IEEE, 2007:195-204.

[16]Lu Ruqian, Jin Zhi, Chen Gang. Ontology-Oriented Requirements Analysis. Journal of Software, 2000, 11(8): 1009-1017(in Chinese)

[17] Nuseibeh B, Easterbrook S, Russo A. Making Inconsistency Respectable in Software Development. Journal of Systems and Software, 2001, 58(2): 171-180.

[18]Mark S. Fox, Fil Salustri. A Model for One-Off Systems Engineering Proceedings of the 12th Conf on AI and Systems Engineering Workshop (AAAI'94), Seattle, USA. 1994.

[19]Haruhiko Kaiya, Motoshi Saeki. Ontology Based Requirements Analysis: Lightweight Semantic Processing Approach. Proceedings of the 5th

International Conference on Quality Software (QSIC'05). Melbourne: IEEE, 2005: 223-230.

[20]Nuseibeh B, Easterbrook S. Requirements Engineering: A Road- map. International Conference on Software Engineering (ICSE'2000). New York: ACM, 2000: 35-46.

[21]Jin Zhi. Ontology-Based Requirements Elicitation. Chinese Journal of Computers, 2000, 23(5): 486-492(in Chinese)

[22]Hunter A, Nuseibeh B. Managing Inconsistent Specifications: Reasoning, Analysis and Action. ACM Trans on Software Engineering and Methodology, 1998, 7(4): 335-367.