



The definition of the AE systems RKMOF is, **Definition1**  $RKMOF = \langle AEGO, DO, TO, AO \rangle$ , where  $AEGO, DO, TO, AO$  represents the AE systems generalized ontology, domain ontology, task ontology and application ontology respectively.

The relationships between various ontologies are shown in Fig.1. The AEGO can be mapped onto the TO and instantiated to the DO. Moreover, the DO can be instantiated to the AO. Both DO and TO can be reused in the same area, translated into the domain requirements model (DRM) by domain analysis and the application requirements model (ARM) by reuse. Documents, domain knowledge, error data and industry standards are all sources of the DO. DO is divided into the action ontology (ACO), process ontology (PO) and structure ontology (SO). This framework is multi-viewpoint because stakeholders all participate in the framework construction.

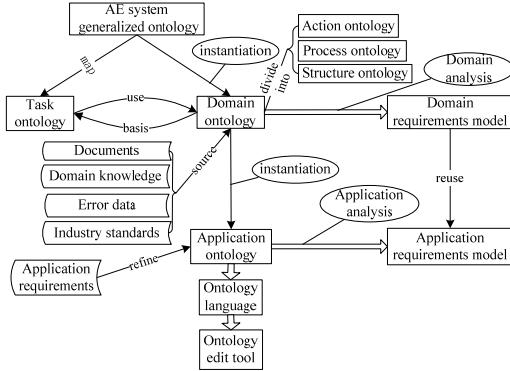


Fig. 1 The AE systems RKMOF

### 3 The structure of AE systems requirements knowledge ontology

#### 3.1 The construction elements of AE systems requirements knowledge ontology

The universal set of ontology construction elements is,  $Element\ Set = \{Concepts, Object-property, Data-property, P^R, P^C, Inherit-hierarchies, Relations, Instances, Map, Rules\}$ . *Concepts* refers to system component, function, behavior, etc. *Inherit-hierarchies* is regarded as a special case of *Relations*. *Object-property* which is a bridge for associating different instances and *Data-property* which is a bridge for associating an instance and a value are two types of property.  $P^R$  represents property restraints of property value type, range, etc.  $P^C$  represents property characteristics, e.g., property “interact” of “system” is symmetric. *Map* represents mapping between ontologies in different hierarchies. *Rules* includes axioms and user-defined rules.

#### 3.2 The AE systems requirements knowledge ontology hierarchic structure

Ontologies in various hierarchies can be constructed by using different construction elements introduced above. We will introduce the ontology structure in detail.

##### 3.2.1 The structure of AEGO

The definition of AEGO is,

**Definition2**  $AEGO = \langle Concepts, Object-property, P^C, Inherit-hierarchies, Relations \rangle$ .

It shows that the AEGO can be constructed by constructing its concept (class), class hierarchy, relation, property and property characteristic.

In Fig.2, the concept classes marked with “\*” are nonterminals, while the rest of them are terminals. There is an inheritance between child class and its father class. Fig.2 also shows that AEGO concepts not only include static elements, but include dynamic elements.

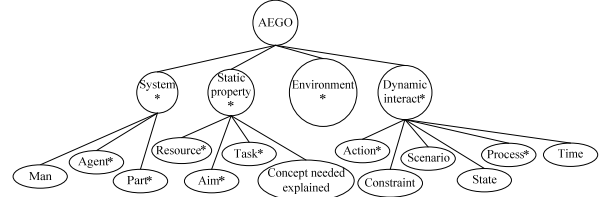


Fig. 2 A part of hierarchies of concept classes in AEGO

According to above AEGO structure, this paper acquires a part of the AEGO concept relation as Tab.1. The left of arrow is source concept node while the right is destination concept node.

Tab. 1 A part of concept relation definition

Type <sup>o</sup>	Relation <sup>o</sup>	Property <sup>o</sup>
Sub-aim <sup>o</sup>	aim <sup>o</sup> →aim <sup>o</sup>	partial ordering <sup>o</sup>
Need <sup>o</sup>	task <sup>o</sup> →resource <sup>o</sup>	support <sup>-1o</sup>
Support <sup>o</sup>	resource <sup>o</sup> →task <sup>o</sup>	need <sup>1o</sup>
Interact <sup>o</sup>	system <sup>o</sup> →environment <sup>o</sup>	symmetry <sup>o</sup>

Notes: “Need” and “need<sup>-1</sup>” are mutually inverse.

“Support” and “support<sup>-1</sup>” are mutually inverse.

Fig.3 shows a part of the AEGO concept space consisting of concepts and relations.

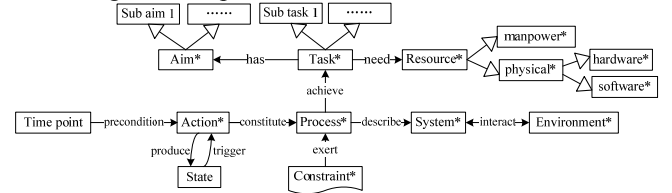


Fig. 3 A part of the AEGO concept space

##### 3.2.2 The structure of DO

The DO is a special ontology for describing designated domain knowledge which includes domain

concepts, relations, activities and criterions. It can be obtained by instantiation of the AEGO.

**Definition4**  $DO = \langle DomConcepts, Object-property, Data-property, P^R, P^C, DomInherit-hierarchies, DomRelations, Rules, Dommap \rangle$ .

$DomInherit-hierarchies$  is a domain concept inheritance relations set. The relation of inheritance is a dualistic relation involving two parameters which can be depicted by “cardinality” and “status”.  $DomInherit-hierarchies = \langle DomInherit-ID, Argument \langle Argument1 \langle cardinality, (status) \rangle, Argument2 \langle cardinality, (status) \rangle \rangle \rangle, (Argument1, Argument2 \in DomConcepts)$ .

$DomRelations$  is a set of domain concept relations except  $DomInherit-hierarchies$ .  $Dommap$  is a total function mapping from  $DomConcepts$  to  $Concepts$ . Thus, equivalent relation “ $\equiv_{domain}$ ” can be defined as,

**Definition5**  $a \equiv_{domain} b$  iff  $Dommap(a) = Dommap(b) = t, a, b \in DomConcepts, t \in Concepts$ .

This paper gives an instance of the structure of FC systems DO. A part of FC systems domain concept set is shown in Tab.2 and a part of FC systems concept relation set is shown in Tab.3.

Tab. 2 A part of domain concept set of FC systems DO

Domain concept name <sup>o</sup>	Corresponding “Concepts” <sup>o</sup>
FC system software <sup>o</sup>	system <sup>o</sup>
Flying control <sup>o</sup>	aim <sup>o</sup>
Attitude/direction control <sup>o</sup>	task <sup>o</sup>
Response time <sup>o</sup>	constraint <sup>o</sup>
Switchover of CPU redundancy <sup>o</sup>	action <sup>o</sup>

Tab. 3 A part of concept relation set of FC systems DO

Type <sup>o</sup>	Source node <sup>o</sup>	Aim node <sup>o</sup>
Has-aim <sup>o</sup>	attitude/direction control <sup>o</sup>	flying control <sup>o</sup>
Has-aim <sup>o</sup>	navigation computing <sup>o</sup>	flying management <sup>o</sup>
Direct-interact <sup>o</sup>	FC system <sup>o</sup>	power system <sup>o</sup>
Direct-interact <sup>o</sup>	FC system <sup>o</sup>	task equipment <sup>o</sup>
Indirect-interact <sup>o</sup>	FC system <sup>o</sup>	geography environment <sup>o</sup>

### 3.2.3 The structure of AO

This paper collects application requirements concepts and relations based on the AEGO and DO according to the purpose of AE systems requirements.

**Definition6**  $AO = \langle AppConcepts, Object-property, Data-property, P^R, P^C, AppInherit-hierarchies, AppRelations, Appmap, Rules \rangle$ .

**Definition7**  $a \equiv_{app} b$  iff  $Appmap(a) = Appmap(b) = t, a, b \in AppConcepts, t \in Concepts$ .

The explanations for the definition 6 and 7 are omitted because they are similar to the definition 4 and 5.

### 3.2.4 The structure of TO

The TO supplies primitives for context description and makes process of merging domain knowledge into

the context more convenient. The definition of requirements elicitation task ontology (RETO) is presented as follows.

**Definition8**  $RETO = \langle Requirement-Eliciting Task, Requirement-Eliciting Task-PSM, Requirement-Eliciting Taskmap \rangle$ , where  $Requirement-Eliciting Task = \langle Task ID, Circumstance \rangle$ ,  $Requirement-Eliciting Task-PSM = \langle Competence, Operational specification, Requirements \rangle$ ,  $Competence = \langle Input Behavior, Output Behavior, Goal \rangle$ ,  $Operational specification = \langle Inference Steps, Control Flow between the Inference Steps, Data Flow between the Inference Steps \rangle$ ,  $Requirements = \langle Concepts, Relations, Facts, Rules \rangle$ ,  $Requirement-Eliciting Taskmap: \{ Requirement-Eliciting Task Concepts \rightarrow Concepts \}$

Due to the limited space, we do not introduce the details of the RETO in this section. It will be introduced in the section 5 with a concrete instance.

## 4 The definition and representation of SREP

Requirements error is the summary of errors in software requirements engineering (SRE). The definition of SREP is introduced as follows referring to the definition of software error pattern [7].

**Definition9**  $SREP$  is the specific error which arises in all stages of SRE, occurs repeatedly in a certain direct scenario, propagates emanative in design and coding stages and may cause system (component) not to accomplish functions anticipated or affect system maintenance. This kind of error is general in the specific direct scenario and can be amended by a certain method.

It shows that the core elements of SREP are the direct scenario, error and solution.

The special property of the AE systems requirements knowledge ontology is its environment feature. Thus this paper merges the domain-related SREP [7] into the RKMOF. This paper uses programming in logic (Prolog) to represent the SREP because it not only expresses factual knowledge, but expresses rules.

Moreover, this paper constructs the RKMOF-based fact bank and transforms the instances represented in web ontology language (OWL) of the fact bank into the facts represented in Prolog grammar. The SREP needs to be brought into the rule bank. The rules in the rule bank are in Prolog grammar form, thus they do not need to be transformed. The concepts and predicates which these rules use should obey the definitions in the RKMOF to hold consistency with the concepts in the fact bank.

## 5 The RKMOF-based requirements elicitation approach with the SREP

In section 5, we present the procedure of the RKMOF-based requirements elicitation approach with the SREP which consists of the preparation stage and implementation stage.

**A.** In the preparation stage, the AEGO is constructed by domain experts and the domain SREP base is constructed by testers and domain experts.

The process of AEGO construction is,

- 1) Ascertaining the range and purpose of AEGO,
- 2) Eliciting the preliminary knowledge according to experts' options and stakeholders' requirements,
- 3) Obtaining the informal description of concepts, properties, class hierarchies and relations by data analysis,
- 4) Constructing the initial informal AEGO.

After the above steps, the knowledge structure of the AEGO can be obtained. This stage is basic because it supplies a knowledge treasure for the next stage.

**B.** The process of the implementation stage is below.

- 1) The TO is constructed by domain experts.
- 2) The DO is constructed by domain experts and requirements analysts. After this step, the knowledge structure of the DO in section 3 can be obtained.
- 3) The DRM can be constructed by requirements eliciting based on the TO and DO. This method is multi-view and guarantees the DRM to obey domain criterions.
- 4) Refinement of initial requirements can be implemented by detecting missing requirements items, adding the missing requirements items and striking out unrealistic functions and time expectations.
- 5) The AO can be constructed by eliciting and classifying of application domain concepts and relations. This process is user-oriented. UML class diagram can be adopted because the AO reflects the static knowledge of system.
- 6) The ARM can be constructed based on the DRM and AO. The ARM involves the dynamic knowledge of the application system. Therefore use cases should be developed by requirements analysts and domain users. The process of use case development is,
  - a) Determining the boundary of use case,
  - b) Identifying scenario, including the precondition, post condition and steps,
  - c) Identifying the primary and secondary scenarios,
  - d) Adopting UML use case diagram and activity diagram to represent use case.

Use case diagram reflects system functions observed by users. Therefore it is relatively stable. While activity diagram can be used to model work flow among different components, illustrate conditions of use case and system state. Therefore it is fluctuant. An instance of activity diagram is shown in Fig.4.

As shown in Fig.4, the alternative path is expressed by a rhombus. Therefore the diagram has parallel

expression so that it can represent the same actions and control flow as in the RETO.

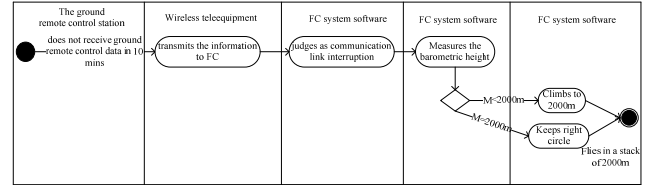


Fig. 4 An activity diagram of UAV FC system software

The RETO describes the aim, inference steps and control flow of the requirements elicitation. Therefore activity diagram should be transformed into the RETO. The RETO corresponding to Fig.4 is shown in Fig.5.

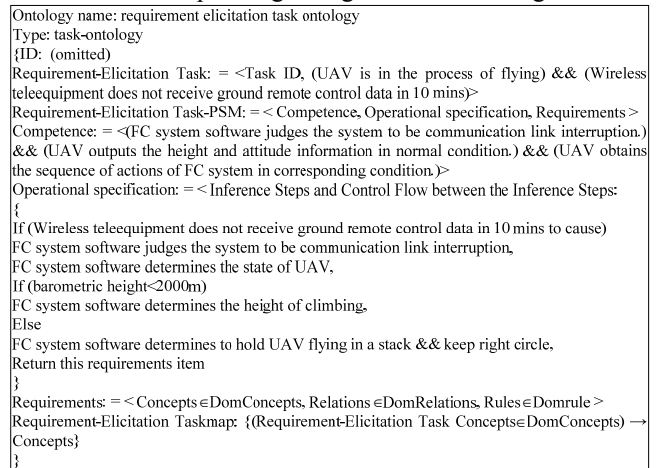


Fig. 5 The UAV FC system software communication link interruption treating RETO

e) Considering abnormal scenario. This step includes the following sub-steps.

i) The SREP should be mapped from the AEGO hierarchy to the DO hierarchy.

The SREP introduced in the section 4 is located in the AEGO hierarchy which is too abstract to apply. Therefore it is necessary to map it onto the DO hierarchy. The concrete process is based on the element *Dommap* in the DO definition. However the mapping is the inverse mapping of *Dommp*.

ii) The SREP should be weaved into the RETO by AOM.

The RETO in the Fig.5 considers the normal conditions. However it does not consider the abnormal conditions and may neglect some important errors which may cause system to malfunction. Therefore the SREP should be weaved into the RETO to enrich the knowledge. Moreover, we find that some actions of system or environment always exist in more than one requirement items. We call this kind of action "crosscutting concerns". Fig.6 shows an instance of crosscutting concerns. In Fig.6, "Pre-flight switch instruction has been executed" cross cuts two safety

requirements items. Therefore it is a “crosscutting concerns”.

AOM is useful in the security requirements elicitation [8]. We also consider this method to be useful in ontology-based requirements elicitation approach for weaving of the SREP.

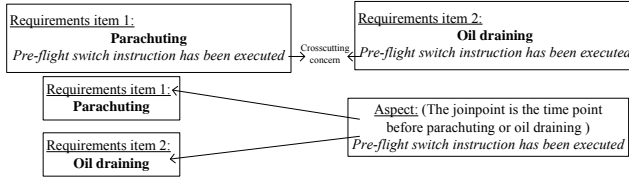


Fig. 6 An instance of crosscutting concerns

Activity diagram can model work flow for use case diagram in detail. Meanwhile, occurrence of activity, achievement of function and work flow in activity diagram emerge in certain scenario which is the same with the scenario of the SREP. Therefore this paper uses AOM to weave the SREP into the RETO, i.e., the corresponding event sequence of activity diagram, according to scenario occurrence place and condition by adopting SLAF [9].

The primary scenario expresses the normal actions of agent and environment. For instance, “UAV parachutes”, the verb “parachutes” is analyzed as the data flow concept. As shown in Fig.5, the alternative path which is expressed by “if...else...” statement reflects the control flow as in programming languages.

The scenario fragment represents the secondary scenario expressing the abnormal actions of agent and environment. It derives from the SREP. The description template of a scenario fragment is shown below.

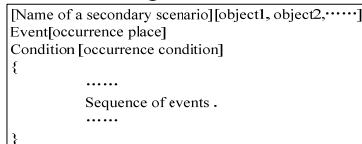


Fig. 7 The description template of a scenario fragment

In Fig.7, the scenario fragment corresponds to aspect, and the “event”, “condition” and “sequence of events” are for “jointpoint”, “pointcut” and “advice” in the aspect-oriented technique. The “event” and “condition” determine the place where the scenario fragment is woven.

Fig.8 shows an instance of weaving scenario. The first part of the figure represents the primary scenario of UAV instruction parachuting which is normal. The second and the third parts give the secondary scenarios of the SREP1 and SREP2. In this approach, we adopts “if” and “while” directives for scenario fragment weaving. Therefore the form of weaved scenario is the same with the RETO and can be used as the result of requirements elicitation. In the process of weaving, we weave the scenario fragment into the place where the action may occur. The event statements that agree with the specific

verbs or nouns in the AO knowledge base are the joinpoints. Therefore the effectiveness of this method depends on the quality of the AO knowledge base greatly.

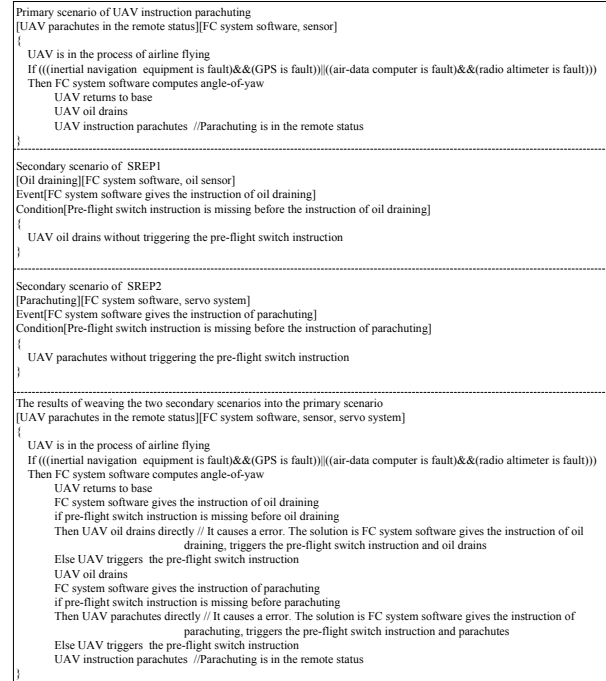


Fig. 8 An instance of weaving scenario

## 6 Case studies

This paper constructs ontology of a subsystem of UAV FC system including several continuous versions of a certain type of UAV FC systems software. Moreover, we adopt the comparison experiment to inspect the RSs to validate the effectiveness of the proposed approach. The comparative method [2] is used for version 3.3.x and the proposed method is used for version 3.3.(x+1).

Tab. 4 The SREP manifestation and severity level distribution of a certain type of UAV FC system software

Manifestation <sup>o</sup>	Error number of each significance degree <sup>o</sup>							
	/ Total number of errors <sup>o</sup>							
	Set I <sup>o</sup>				Set II <sup>o</sup>			
	C <sup>o</sup>	I <sup>o</sup>	O <sup>o</sup>	T <sup>o</sup>	C <sup>o</sup>	I <sup>o</sup>	O <sup>o</sup>	T <sup>o</sup>
Function <sup>o</sup>	0 <sup>o</sup>	1 <sup>o</sup>	1 <sup>o</sup>	2 <sup>o</sup>	0 <sup>o</sup>	0 <sup>o</sup>	2 <sup>o</sup>	2 <sup>o</sup>
Performance <sup>o</sup>	0 <sup>o</sup>	1 <sup>o</sup>	0 <sup>o</sup>	1 <sup>o</sup>	0 <sup>o</sup>	1 <sup>o</sup>	0 <sup>o</sup>	1 <sup>o</sup>
Interface <sup>o</sup>	0 <sup>o</sup>	2 <sup>o</sup>	3 <sup>o</sup>	5 <sup>o</sup>	0 <sup>o</sup>	1 <sup>o</sup>	3 <sup>o</sup>	4 <sup>o</sup>
Safety <sup>o</sup>	1 <sup>o</sup>	1 <sup>o</sup>	0 <sup>o</sup>	2 <sup>o</sup>	1 <sup>o</sup>	0 <sup>o</sup>	0 <sup>o</sup>	1 <sup>o</sup>
Environment <sup>o</sup>	1 <sup>o</sup>	2 <sup>o</sup>	0 <sup>o</sup>	3 <sup>o</sup>	0 <sup>o</sup>	0 <sup>o</sup>	0 <sup>o</sup>	0 <sup>o</sup>
Maintenance <sup>o</sup>	0 <sup>o</sup>	1 <sup>o</sup>	1 <sup>o</sup>	2 <sup>o</sup>	0 <sup>o</sup>	0 <sup>o</sup>	2 <sup>o</sup>	2 <sup>o</sup>
Document <sup>o</sup>	0 <sup>o</sup>	1 <sup>o</sup>	3 <sup>o</sup>	4 <sup>o</sup>	0 <sup>o</sup>	2 <sup>o</sup>	2 <sup>o</sup>	4 <sup>o</sup>
Total <sup>o</sup>	19 <sup>o</sup>				14 <sup>o</sup>			

Note: C, I and O stand for severity level “critical”, “important” and “ordinary”. T stands for “total”.

Tab.4 records the results of the RS inspection. The set I adopts the comparative method and the set II adopts ontology-based method introduced in this paper.

It shows that the total number of errors of the set I is larger than the set II. The number and severity level distribution of performance errors in the set I is the same with the set II. The number of function errors of the set I is the same with the set II; however, the severity level distributions of them are different. The results of maintenance error and document error are the same with the function error. The number of the rest kinds of errors in the set I is larger than the set II respective with the set I having more severe errors. Moreover the number of environment errors in the set II is zero.

This paper adopts metrics [1] to estimate the RS quality quantitatively. Note that  $|Con|=36$  and  $|Rel|=45$ .  $|ReqItem|$  of RS 1 is 11 while  $|ReqItem|$  of RS 2 is 13. Tab.5 gives the quality measure results of RS 1 and RS 2.

Tab. 5 Quality measure results of RS 1 and 2

Metrics <sup>o</sup>	RS1 <sup>o</sup>	RS2 <sup>o</sup>
Correctness <sup>o</sup>	9/11=81.8% <sup>o</sup>	13/13=100% <sup>o</sup>
Completeness <sup>o</sup>	68/81=84% <sup>o</sup>	77/81=95% <sup>o</sup>
Consistency <sup>o</sup>	14/23=60.9% <sup>o</sup>	15/23=65.2% <sup>o</sup>

There is an evident difference of “correctness” and “completeness” between the RS 1 and 2. For “correctness”, ideally, all requirements items should be mapped onto ontology because ontology is the semantic basis of problem domain. Therefore, “correctness” can reflect the ratio of mapping elements. The higher ratio is the better quality RS possesses. The result shows that some requirements items of the RS 1 do not correspond to the ontology knowledge base. For “completeness”, ideally, all ontology elements should be referred in RS. Therefore, “completeness” can reflect the ratio of mapping elements. The higher ratio is the better quality RS possesses. The result shows that the knowledge which the RS 1 involves is incomplete because many ontology elements have not emerged in the RS 1. That is because the requirements elicitation of RS 1 is only based on the multi-ontology framework which does not involve the SREP. The experiment results also show that the completeness of knowledge in requirements knowledge ontology will affect the requirements development greatly.

The results also show that “consistency” of two RSs is low. It explains the cause of the relatively large number of interface errors in Tab.4. Therefore the ontology should be complemented by adding the interface knowledge.

The deficiency of this experiment is that it analyzes the quality of RS only by its static property. Thus, how to estimate the quality of RS by the dynamic property directly needs to be further studied.

## 7 Conclusions

We propose a RKMOF-based requirements elicitation approach with the SREP for AE system software. Firstly, we construct the AE systems RKMOF. Secondly, we present the construction elements of AE systems requirements knowledge ontology. Thirdly, we adopt the AEGO, DO, TO and AO to express knowledge in different hierarchies. Details are also given. Furthermore, the definition and representation of SREP are presented. Finally, the detailed processes of RKMOF-based requirements elicitation approach with the SREP are given. The experiment results validate the effectiveness of the proposed approach preliminarily. However, our studies still have some deficiencies. Thus, the following issues will be further discussed, 1) The SREP bank should be enriched to involve more SREP to play a role of error-prevention in requirements elicitation; 2) How to estimate the quality of RS by the dynamic property directly needs to be further studied; 3) Results of the experiment show that the number of interface errors of the set II is not small. It means that the knowledge of interface is relatively incomplete. Therefore, the interface property should be further described.

## References

- [1] H.Kaiya and M.Saeki, Using Domain Ontology as Domain Knowledge for Requirements Elicitation, in Proceedings of the 14<sup>th</sup> IEEE International Requirements Engineering Conference, 2006: 186-195.
- [2] Z.Y.Li, Z.X.Wang, Y.Y.Yang, Y.Wu and Y.Liu, Towards a Multiple Ontology Framework for Requirements Elicitation and Reuse. The 31<sup>st</sup> Annual International Computer Software and Applications Conference, IEEE, 2007:189-195.
- [3] Cf. T. R. Gruber, A Translation Approach to Portable Ontologies. Knowledge Acquisition, 1993, 5(2):199-220.
- [4] Z.Jin, Ontology-based Requirements Elicitation, Chinese J. Computers, 2000, 23(5): 486-492.
- [5] Z.Jin, D.Bell and F.G.Wilkie, Automatically Acquiring the Requirements of Business Information Systems by Using Business Ontology, in Proceedings of ECA I'98 Workshop on applications of Ontology and Problem Solving Methods, 1998: 78-87.
- [6] Y. Lee and W. Zhao, An ontology-based Approach for Domain Requirements Elicitation and Analysis, in Proceedings of the First International Multi-Symposiums on Computer and Computational Sciences, 2006: 364-371.
- [7] X.Hu, etc., Application of Software Requirement Error Pattern Based on Ontology, Journal of Beijing University of Aeronautics and Astronautics, 2009, 35(6): 723-727.
- [8] H. Itoga and A. Ohnishi, Security Requirements Elicitation via Weaving Scenarios based on Security Evaluation Criteria, QSIC, 2007: 70-79.
- [9] H. H. Zhang and A. Ohnishi, Transformation between scenarios from different viewpoints, IEICE transactions on information and systems, vol. E87-D, No.4, 2004:801-810.