

Researches on Software Requirements Elicitation Approach of the Aviation Electronics Systems based on Multi-ontology

Xuan Hu¹, Jie Liu^{1*}, Yichen Wang²

¹ CEPREI, Guangzhou, China, 510610

² BUAA, Beijing, China, 100191

huxuan@ceprei.com, liujie@ceprei.com, wangyichen@buaa.edu.cn

**Corresponding author, liujie@ceprei.com*

Abstract—Because of the increasing complexity of software systems as well as the more various types of R&D personnel of the workers, software development is going to a significant trend of knowledge intensification. In the field of knowledge engineering, an ontology is an explicit specification of a conceptualization. In this paper, the ontology method is used to construct the requirement knowledge framework and the various of requirement knowledge is expressed as a clear, complete and consistent hierarchical ontology concept and association, which is more conducive to knowledge sharing and reuse as well as reflects multiple viewpoints of stakeholders. Furthermore, the relevant content of the decomposition of the generalization ontology is added in the framework, which is divided into the structure ontology and action ontology. It makes up for the deficiency of undifferentiated knowledge composition representation of the generalization ontology. Besides, the concept of software requirement error pattern is proposed and integrated into the multi-ontology framework of requirement knowledge in a consistent form. Based on the framework, the domain requirement model and the application requirement model can be built. It can be adopted as the basis of the requirement elicitation activities to avoid errors and improve the quality and reliability of software products.

Keywords—software requirements elicitation; software requirements error; pattern; ontology; aspect-oriented

I. INTRODUCTION

Software requirements elicitation as the basic work of requirements engineering not only has a direct impact on the requirements engineering activities itself, but also relates to the success or failure of the entire software project. It is increasingly valued as a key part of obtaining high-quality software. However, the current requirements elicitation activities still have the following problems, ambiguity, inconsistency and incompleteness. The increase in the size and complexity of software systems and the increasing complexity of types of R&D personnel have led to significant knowledge-intensive, diversified and complex software developments that have made it difficult to acquire complete knowledge. An ontology is a logical theory accounting for the intended meaning of a formal vocabulary, i.e., its ontological commitment to a particular conceptualization of the world. The intended models of a

logical language using such a vocabulary are constrained by its ontological commitment. An ontology indirectly reflects this commitment by approximating these intended models [1]. According to Gruber, an ontology is an explicit specification of a conceptualization [2, 3]. Therefore it is effective to introduce the ontology method to requirements elicitation to solve the above problems.

Nowadays the typical representatives of ontology-based requirements elicitation methods include, the requirement elicitation method based on multi-ontology framework [4], the automatic elicitation method based on enterprise ontology [5], the method of using domain ontology (DO) as domain knowledge for requirements elicitation [6], the use of semi-formal representation of semantic guidance consisting of domain concepts, associations, and axioms to assist requirements elicitation activities [7], the formal ontology representation of the original requirements is used to elicit and analyze the requirements based on the top-down refinement of the metamodel [8]. In this paper, the incompleteness problem in the requirements elicitation of AE system is fixed by integrating the software requirements error pattern (SREP) information into the requirements knowledge multi-ontology framework (RKMOF) to enhance the a priori information of the process. The effectiveness of the method is illustrated by evaluating the resulting RS. The example verification section uses the flight control (FC) system as an example to illustrate the above process.

II. THE AE SYSTEMS RKMOF

The knowledge systems modeling can be studied by means of knowledge aided design system (KADS) [4]. However, it has some deficiencies, e.g., lack of strong ties between different knowledge levels, incompleteness of knowledge, etc. Thus, it is necessary to integrate relatively independent knowledge layers to form a knowledge system. Ontology is an effective mechanism to achieve this. There are four types of ontologies, Top-level Ontologies, Domain and Task Ontologies, Application Ontologies [1]. On this basis, combined with existing research results [5, 9] and practical experiences, this paper obtains the quaternion of the AE systems RKMOF as follows,

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

The relationship between the entities in this definition is shown in Figure 1. AEGO can be mapped to the TO, instantiated to the DO and eventually the AO. AEGO can be divided into structure ontology and action ontology. Both DO and TO can be reused in the same domain, transformed into the domain requirements model (DRM) through domain analysis and the application requirements model (ARM) by reuse. In this framework, error data, domain knowledge and industry standards are all sources of DO. This framework is multi-viewpoint because domain experts, users and developers can all participate in the process of framework construction.

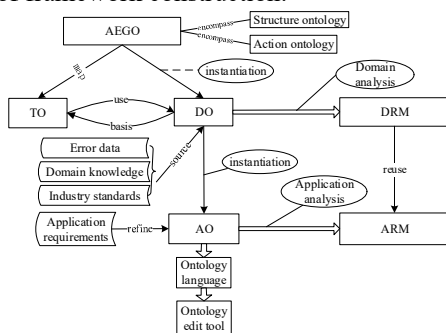


Figure 1. The AE systems RKMOF

III. THE STRUCTURE OF THE AE SYSTEMS REQUIREMENTS KNOWLEDGE ONTOLOGY

3.1 AE systems requirements knowledge ontology component

The complete set of ontology constituent elements is, $Element\ Set = \{Concepts, Object-property, Data-property, P^R, P^C, Inherit-hierarchies, Relations, Instances, Map, Rules\}$. "Inherit-hierarchies" can be regarded as a special case of "Relations". "Concepts", "Inherit-hierarchies", "Relations" and "Instances" constitute the basic skeleton of the ontology. Other elements besides these are the refinement of the basic skeleton.

3.2 AE systems requirements knowledge ontology hierarchy

3.2.1 AEGO construction

AEGO reflects the ontology components common to such systems. Its definition is,

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

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

1) AEGO concept class and class hierarchy construction

A part of hierarchies of concept classes in AEGO is shown as Figure 2, and the concept class with a "*" is a

non-terminal concept class, and the rest are all terminating concept classes.

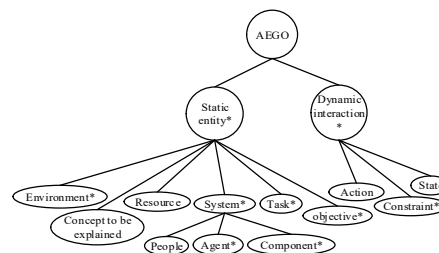


Figure 2. A part of hierarchies of concept classes in AEGO

If the inheritance relationship is defined as,

Definition3 The mechanism in which the subclass in the AEGO concept class hierarchy automatically shares the properties and structures of its parent class.

In that way, the subclass under the non-terminal concept class and the parent class form an inheritance relationship. It shows a new class can be implemented on the basis of an existing concept class, and the content of the existing class definition is taken as its own content, and new content is added. Figure 2 also shows that the AEGO concepts contain both static and dynamic elements of the ontology.

2) The decomposition of the AEGO

Based on the above researches, this paper further divides the AEGO into two categories, structure ontology ($O_{structure}$) and action ontology (O_{action}). They are used to describe the most basic structure and function of the system, and are represented by first-order predicate logic, $AEGO = O_{structure} \cup O_{action}$.

A. $O_{structure}$

A structure is a set of constraints on the action of a system, including a set of systems, people, agents, components, environments, and static properties and dynamic interactions that act on them.

B. O_{action}

An action describes the state transition of the system and environment. The state triggers the action and produces a new state by the action. The action in the scenario is expressed by an action and its associated trigger and effect states. An action is usually performed in its input-to-output conversion mode; furthermore, it can be extended to get multiple actions, e.g., conditional action, choice action, and concurrent issue, etc.

3) AEGO concept relationship and concept space construction

According to the above steps, the AEGO concept semantic association is obtained on the basis of the concept class hierarchy (the left arrow is the source concept node and the right is the destination concept node). And by integrating concepts (classes), concept class hierarchies, relationships, properties, and property characteristics, a part of ontology concept space of the AEGO is shown in Table 1. This is the informal representation of the initial AEGO. It can be proved that the inheritance relationship in the

concept association is a partial order relationship; the

“Interact” relationship is a symmetric relationship.

TABLE 1. A PART OF ONTOLOGY CONCEPT SPACE BASED ON STRUCTURE AND ACTION CONCEPT TYPES

concept type	concept relationship	relationship value type	P ^C	Description
Structure	objective	P Objective	partial order	Objective $\xrightarrow{\text{subObjective}}$ Sub-objective
	hasObjective	Task ⁺ →Objective	-	Task $\xrightarrow{\text{hasObjective}}$ Objective
	interact	System ⁺ →Environment	symmetry	System $\xrightarrow{\text{interact}}$ Environment
Action	produce	State	-	Action $\xrightarrow{\text{produce}}$ State
	operateOn	P(operator× resource)	-	Action $\xrightarrow{\text{operateOn}}$ (Operator× Resource)

Notes: “need” and “need⁻¹” are mutually inverse, “support” and “support⁻¹” are mutually inverse.

3.2.2 AE system DO construction

Definition4 A domain is a collection of systems with similar or identical functions that are used to address specific domain issues. This system is versatile when dealing with different application needs.

DO is a specialized ontology used to describe knowledge in a given domain. It presents a description of the domain concepts and relationships, domain activities, and the characteristics and laws of the field. These concepts and relationships are obtained by instantiating the corresponding content in the AEGO.

1) DO definition

Definition5 $DO = \langle DomConcepts, Object-property, Data-property, P^R, P^C, DomInherit-hierarchies, DomRelations, Rules, Dommap \rangle$, where $DomConcepts = \langle DomConcept-name, DomConcept-ID \rangle$; $DomInherit-hierarchies = \langle DomInherit-ID, Argument \langle Argument1 \langle cardinality, (status) \rangle, Argument2 \langle cardinality, (status) \rangle \rangle \rangle$, $(Argument1, Argument2 \in DomConcepts)$; $DomRelations = \langle DomRelation-name, DomRelation-ID, Argument \langle Argument1 \langle cardinality, (status) \rangle, Argument2 \langle cardinality, (status) \rangle \rangle \rangle$, $(Argument1, Argument2 \in DomConcepts)$.

Definition6 $a \equiv_{domain} b$ iff $Dommap(a) = Dommap(b) = t$, $a, b \in DomConcepts$, $t \in Concepts$.

This equivalence relationship can be recorded as $[t] \equiv_{domain}$.

2) DO concept set and concept relationship set

In this paper, the research field is limited to FC system. The specific content of the FC system part concept set and concept relationship set can be found in the reference [9].

3.2.3 TO construction

TO is a collection of words that describes the problem-solving structure of the existing task. Since this paper studies the ontology-based requirements elicitation method, the definition of the requirements elicitation TO (RETO) is given as follows [9].

Definition7 $RETO = \langle Requirement-Eliciting Task, Requirement-Eliciting Task-PSM, Requirement-Eliciting Taskmap \rangle$,

Specifically expressed,

Ontology name: RETO

Type: TO

{ID: // identifier

Requirement-Eliciting Task: = $\langle Task ID, Circumstance \rangle$,

Requirement-Eliciting Task-PSM: = $\langle Competence,$

Operational specification, Requirements $\rangle$, Competence: =

$\langle Input Action, Output Action, Objective \rangle$, Operational

specification: = $\langle Inference Steps, Control Flow between$

the Inference Steps, Data Flow between the Inference

Steps $\rangle$, Requirements: = $\langle Requirement-Eliciting Task$

Concepts, Requirement-Eliciting Task Relations,

Requirement-Eliciting Task Facts, Rules $\rangle$, Requirement-

Eliciting Taskmap: $\{ Requirement-Eliciting Task Concepts$

$\rightarrow Concepts \}$

3.2.4 AO construction

The definition of the AO is given below.

Definition8 $AO = \langle AppConcepts, Object-property, Data-property, P^R, P^C, AppInherit-hierarchies, AppRelations, Appmap, Rules \rangle$.

Definition9 $a \equiv_{app} b$ iff $Appmap(a) = Appmap(b) = t$, $a, b \in AppConcepts$, $t \in Concepts$.

This equivalence relationship can be recorded as $[t] \equiv_{app}$.

The explanations for the definition 8 and 9 are omitted because they are similar to the definition 5 and 6.

IV. SREP DEFINITION AND ONTOLOGY REPRESENTATION

The definition of SREP can be given with reference to the software error pattern definition [10].

Definition10 SREP refers to the error produced in the software requirements development stage, which occurs repeatedly in a certain scenario of error lifetime, and spreads in subsequent design and implementation, and may cause the system (component) to fail to perform the expected function or affect the system maintainability. Such errors are general and common in a particular scenario and can be corrected by some means.

The definition shows that the core of the SREP is the “scenario”, “error-manifestation” and “solution”.

In this paper, the ontology representation of the domain-related SREP proposed in [10] is improved, and the new one is obtained and integrated into the multi-ontology requirement knowledge framework. The SREP is summarized as experiences, and its role is reflected in the regulation of new requirements elicitation activities in the form of rules. Because programming in logic (Prolog) can not only be used to represent the factual knowledge of states, attributes, concepts of things, but also to express the

causal relationship of things, this paper uses Prolog to represent the SREP and incorporate it into the rule base.

V. RKMOF-BASED REQUIREMENTS ELICITATION APPROACH

In addition to the basic knowledge required for requirements elicitation, the framework also adds knowledge of SREP, which greatly enriches the prior information of requirements elicitation, therefore it is relatively complete. The following are the specific steps of requirements elicitation method based on the RKMOF.

1) Preliminary phase

The RKMOF-based requirements elicitation method pays great attention to the accumulation of knowledge in the early stage. That is, before the formal requirements elicitation begins, the testers and domain experts must jointly construct the SREP library, as shown in Figure 3.

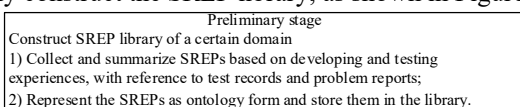


Figure 3. The work steps of RKMOF-based software requirements elicitation method preliminary phase

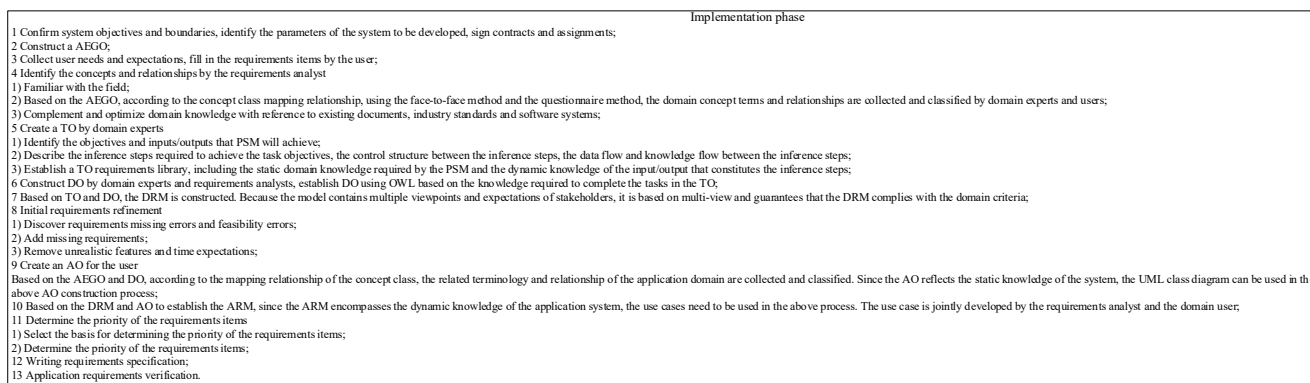


Figure 4. The work steps of RKMOF-based software requirements elicitation method implementation phase

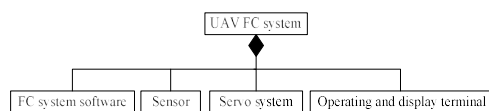


Figure 5. UAV FC systems concept classes

system function that the user can observe and is therefore relatively stable. An example of the use case diagram is shown in Figure 6. (to be continued)

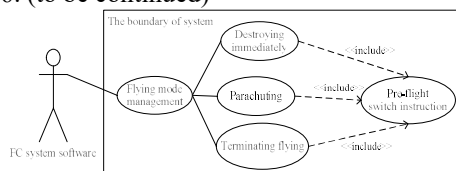


Figure 6. An use case diagram of UAV FC system software

Activity diagram can be used to model workflows between different components, illustrating the conditions of the use

2) Implementation phase

Figure 4 presents the work steps of RKMOF-based software requirements elicitation method implementation phase. This paper selects UML class diagram, use case diagram and activity diagram to assist AO modeling and application description elicitation activities to obtain detailed level application software specification document.

(1) UML class diagram application

AO can be constructed by extracting and classifying concepts and relationships in the application domain. Since AO reflects the static knowledge of the system, UML class diagram can be used in the above activities. Figure 5 shows an example of AO concept classes.

(2) UML use case and activity diagram application

The ARM encompasses the dynamic knowledge of the application system, therefore the requirements analysts and domain experts (users) can adopt the UML use case and activity diagram to model, as follows,

a) Clear use case boundaries; b) Identify the scenario, including preconditions, postconditions, and steps; c) Distinguish between primary scenario and secondary scenario; d) Adopt UML use case and activity diagram to represent use cases. The use case diagram reflects the

cases and the state of the system, and therefore is variable. An example of the activity diagram is shown in Figure 7.

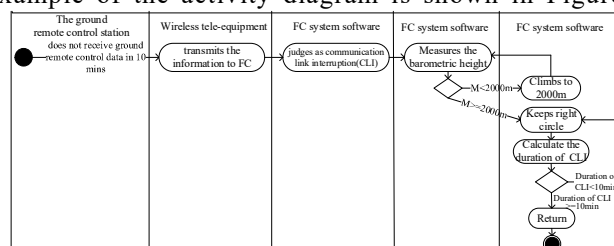


Figure 7. An activity diagram of UAV FC system software

RETO is located on the upper level of AO. Thus it needs to be mapped to the application layer and form an application layer RET. As shown in Figure 7, the optional path is represented by a diamond. The diagram can represent parallel expressions, thus the same action and control flow can be represented in the application layer RET. The application

layer RET describes the objectives, inference steps and control flow of the requirements elicitation activity. Therefore activity diagram can be transformed into application layer RET. The UAV FC system software communication link interrupt processing RET is shown in Figure 8.

```

Name:RET
{ID: (omitted)
Requirement-Elicitation Task: = <Task ID, (UAV is in the process of flying) && (Wireless
teleequipment does not receive ground remote control data in 10 mins)>
Requirement-Elicitation Task-PSM: = < Competence, Operational specification, Requirements >
Competence: = <(FC system software judges the system to be communication link interruption)
&& (UAV outputs the height and attitude information in normal condition) && (UAV obtains the
sequence of actions of FC system in corresponding condition)>
Operational specification: = < Inference Steps and Control Flow between the Inference Steps:
{
If (Wireless teleequipment does not receive ground remote control data in 10 mins )
FC system software judges the system to be communication link interruption
FC system software determines the state of UAV
If (barometric height<2000m)
FC system software determines the height of climbing
Else
FC system software determines to hold UAV keeping right circle
If (the duration of communication link interruption < 10 mins)
FC system software determines to hold UAV keeping right circle
Else
Return
Return this requirements item
}
Requirements: = < Requirement-Elicitation Concepts ∈ APPConcepts, Requirement-Elicitation
Relations ∈ APPRelations, Requirement-Elicitation Rules ∈ APPRule >
Requirement-Elicitation Taskmap: {(Requirement-Elicitation Task Concepts ∈ APPConcepts) →
DomConcepts, DomConcepts → Concepts}
}

```

Figure 8. The UAV FC system software communication link interrupt processing RET (application layer)

e) Consider an anomaly scenario. Specific steps are below,

①Map the SREP from AEGO layer to DO layer;

Place the SREP in the AEGO layer is too abstract to be difficult to use, therefore it needs to be mapped to the DO layer. The specific steps are based on the *Dommap* in the DO definition and are its inverse mapping.

②Weave the SREP into RET through aspect-oriented scene weaving.

The RET shown in Figure 8 only considers the normal scenario, and does not consider the abnormal scenario, thus it is possible to ignore certain key errors. The weaving of SREP into RET will greatly enrich the knowledge structure. Besides, in the actual use scenario, it is often the case that the action of the system or environment exists in multiple requirements items. This is called “crosscutting concerns” and Figure 9 shows an example. The requirement “Ensure Flight Control Switch Pressed” crosses two system safety requirements items. Therefore it is a “crosscutting concerns” of requirements.

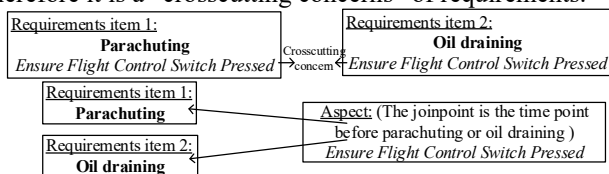


Figure 9. An instance of crosscutting concerns

Aspect-oriented programming (AOP) methods play an important role in the elicitation of security requirements [11]. As is well known, use case diagram can be used for coarse-grained functional descriptions, while activity diagram can model the workflow of use case diagram in more detail. Meanwhile, the occurrence of activities, the implementation of

functions, and the implementation of workflows in activity diagram in some scenarios are similar to those in SREP. Therefore, this paper uses SLAF [12] to weave the SREP into the RET according to the conditions of the scenario, i.e., the corresponding event sequence of the activity diagram.

The scenario in the activity diagram is defined as follows,

Definition11 The scenario in the activity diagram is a sequence of events that describes agent action, environmental state change, and interaction between the agent and the environment.

The scenario in this definition describes the primary scenario in which normal agent action and environmental state change occur. For example, “the UAV parachutes”, the verb “parachutes” is analyzed as a data flow concept. The optional path is represented by “if...else...” which describes the control flow in the programming language. The scenario segment in the activity diagram represents a secondary scenario representing anomalous action, which is derived from SREP. The description template of the scenario segment is shown in Figure 10. The concepts of “secondary scenario”, “Event”, “Condition”, and “Sequence of events” may correspond to concepts in AOP. A “secondary scenario” is equivalent to an “aspect” in AOP. An “event” is equivalent to a “joinpoint” in AOP. The “condition” is equivalent to a “pointcut” in AOP, and the “sequence of events” is equivalent to an “advice” in AOP. The “event” and “condition” determine where the scenario segment is woven.

```

[Name of a secondary scenario][object1, object2,.....]
Event[occurrence place]
Condition[occurrence condition]
{
.....
Sequence of events.
.....
}

```

Figure 10. The description template of the scenario segment

VI. CASE STUDIES

In this paper, some parts of the UAV FC system were selected for ontology modelling. The effectiveness of the ontology-based requirements elicitation method was verified by a comparative experiment. Since the software had to develop several subsequent versions after ontology modelling, the version 3.3.x was used in the usual method for requirements elicitation; in the version 3.3.(x+1), the ontology-based method of this paper was adopted.

Table 2 records the error number distribution of each error-manifestation of SREP. An inspection was conducted using the same group of inspectors for comparative experiment. The Group I RS was developed based on the usual requirement elicitation method, while the Group II RS was developed based on the ontology-based method of this paper. It shows that the total number of errors in the Group I RS is much higher than that of Group II. Besides, the severity level of errors detected in Group I is higher, and it is more significantly in the functional errors, interface errors, safety errors, environmental errors, etc. The direct cause can be initially identified as, Group II used an ontology-based method, while Group I used only a general method.

TABLE 2. THE ERROR NUMBER DISTRIBUTION OF EACH ERROR-MANIFESTATION OF SREP OF A CERTAIN TYPE OF UAV FC SOFTWARE

Error-manifestation	Group I				Group II			
	C	I	A	T _i	C	I	A	T _i
Functional error	4	1	1	6	0	1	1	2
Performance error	0	0	1	1	0	0	1	1
Interface error	3	6	0	9	1	2	0	3
Safety error	4	1	0	5	1	0	0	1
Environmental error	2	2	0	4	0	0	0	0
Maintenance error	0	1	1	2	0	0	2	2
Documentation error	0	3	4	7	0	2	4	6
T _s	13	14	7	-	2	5	8	-
T	34				15			

Note: C, I and O stand for severity level “critical”, “important” and “average”. T_i stands for “total number of each type of error”, T_s stands for “total number of each severity level of error”, T stands for “total number of errors”.

This paper will evaluate the quality of RS based on the metrics in [6]. The quality measure results for RS document I and II are given in Table 3.

TABLE 3. QUALITY MEASURE RESULTS OF DOCUMENT I AND II

Metrics	Document I	Document II
Correctness	31/40=77.5%	38/41=92.7%

From this result, it can be seen that the difference in the “correctness” between Document I and Document II is more obvious. Ontology is the semantic basis for building a specific problem domain. Ideally, all requirements items should be able to find the corresponding element in the ontology. Therefore, **(the number of items that can be mapped to the ontology / the total requirements items)** can reflect the proportion of the mapped elements. The higher the ratio is the higher quality of the RS documents. This ratio of Document I to Document II is significantly lower. Some of the requirements items in Document I were not included in the ontology library, which indicates that some of the requirements items in Document I do not match the actual application. This fact also explains the results of the requirements inspection in Tab. 2. It can be seen that the requirements knowledge ontology will have a major impact on the entire requirements development process. In general, the quality of RS documents obtained based on ontologies for requirements development will be higher than the quality of RS documents obtained by requirements development based on general methods. Therefore, the ontology element, i.e., the knowledge element, should be fully integrated in the early stage of the requirements development process and in the requirements elicitation activities.

VII. CONCLUSIONS

The RKMOF proposed in this paper fully considers the real needs of the complete knowledge base of the requirements elicitation activities and the entire requirements engineering. It adopts various ontologies to reflect every aspect of requirements knowledge, hierarchically divides and stores knowledge, and integrates static and dynamic knowledge in the multi-ontology framework, and by mapping the layers of the ontology to make it a whole. Meanwhile, the empirical knowledge such as SREP is integrated into RKMOF in a consistent form. The framework is universal and suitable for

any AE system. Based on the framework, the DRM and ARM can be further constructed to guide the requirements elicitation activities to obtain the RS document with fewer errors, thereby improving the quality of the RS document and further improving the reliability of the software system.

REFERENCES

- [1] Guarino. N., Formal ontology in information systems, in Proceedings of the 1st international conference (FOIS'98), Trento, Italy, IOS press, 1998.
- [2] Cf. T. R. Gruber., A translation approach to portable ontologies, Knowledge Acquisition, 5(2): pp. 199-220, 1993.
- [3] Thomas R. Gruber., Toward Principles for the Design of Ontologies Used for Knowledge Sharing, International Journal of Human-computer Studies, 43(5-6): pp. 907-928, 1995.
- [4] Z.Y.Li, et al., Towards a Multiple Ontology Framework for Requirements Elicitation and Reuse, in Proceedings of the 31st Annual International Computer Software and Applications Conference (COMPSAC'07), pp. 189-195, 2007.
- [5] Z.Jin, Ontology-based Requirements Elicitation, Chinese Journal of Computers, 23(5): pp. 486-492, 2000.
- [6] H.Kaiya and M.Sacki, Using Domain Ontology as Domain Knowledge for Requirements Elicitation, in Proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06), pp. 186-195, 2006.
- [7] Farfeleder, S. et al., Ontology-Driven Guidance for Requirements Elicitation, in Extended Semantic Web Conference, ESWC 2011: The semantic web: Research and Applications. pp. 212-226, 2011.
- [8] Y. Lee and W. Zhao, An ontology-based Approach for Domain Requirements Elicitation and Analysis, in Proceedings of the 1st International Multi-Symposiums on Computer and Computational Sciences (IMSCCS'06), pp. 364-371, 2006.
- [9] X.Hu, et al., Construction and Estimation of Requirement Knowledge Ontology for Avionics Electronics System Software, Computer Engineering, 39(2): pp. 56-62, 2013.
- [10] X.Hu, et al., Application of Software Requirement Error Pattern Based on Ontology, Journal of Beijing University of Aeronautics and Astronautics, 35(6): pp. 723-727, 2009.
- [11] H. Itoga and A. Ohnishi, Security Requirements Elicitation via Weaving Scenarios based on Security Evaluation Criteria, the 7th International Conference on Quality Software (QSIC 2007), pp. 70-79, 2007.
- [12] H. H. Zhang and A. Ohnishi, Transformation between scenarios from different viewpoints, IEICE transactions on information and systems, E87D, pp. 801-810, 2004.



XUAN HU received her bachelor's degree in mathematics and the doctor's degree in Aerospace Science and Technology both from Beihang University. She entered the postdoctoral workstation of CEPREI Laboratory of Ministry of Industry and Information Technology of People's Republic of China in 2011. She is currently a senior engineer in Information Security Research Center of CEPREI Laboratory. Her main research interests include software requirements engineering, software reliability, self-adaptive software and highly reliable fault tolerant technology.



JIE LIU received his bachelor's degree in computer science from Xidian University. He is currently the director of the Information Security Research Center (Industrial Control System Information Security Evaluation Laboratory) of CEPREI Laboratory of Ministry of Industry and Information Technology of People's Republic of China. His main research interests include information security, software reliability and security, and software evaluation technology.



YICHEN WANG received the Ph.D. degree in Reliability and System Engineering from Beihang University, Beijing, in 2006. He is currently an associate professor of software reliability in Beihang University. His research interests include complex software reliability, model-based software testing and software quality.