

Requirements Knowledge Model Construction and Requirements Elicitation Method of Avionics Systems Software based on Multi-ontology

Xuan Hu, Jie Liu

Information Security Research Center, CEPREI, No. 110, Dongguanhuang Road, Guangzhou, China

Key laboratory of Ministry of Industry and Information Technology, No. 110, Dongguanhuang Road, Guangzhou, China
huxuan@ceprei.com, 47507346@qq.com

Abstract—Avionics systems software usually has the characteristics of complex structure, wide function coverage, high reliability, and high safety. These characteristics have led to the increasing scale and complexity of software systems, and the increasing variety of research & development (R&D) personnel. This further intensify the knowledge-intensive trend of software development, and make the software requirements elicitation activities more complicated. In the knowledge engineering community, an ontology is an explicit specification of a conceptualization. This paper uses the ontology method to construct a requirement knowledge framework and 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. This paper builds a requirements knowledge multi-ontology framework which is divided into GO (generalization ontology), TO (task ontology), DO (domain ontology), and AO (application ontology). And it decomposes the GO into structure ontology and action ontology in the framework. It makes up for the deficiency of undifferentiated knowledge representation of the GO. Then, this paper proposes and integrates the concept of software requirement error pattern into the multi-ontology framework in a consistent form. In addition, this paper evaluates the quality of the constructed ontology and the evaluation results show that the constructed ontology in this paper has a high quality. Finally, this paper facilitates the requirements elicitation based on the multi-ontology framework to avoid errors and improve the quality and reliability of software products.

Keyword—error, ontology, requirements elicitation, software requirements

I. INTRODUCTION

SOFTWARE requirements elicitation (SRE) is the most critical knowledge-intensive activity in a software development process; however, implementing effective requirements elicitation and obtaining correct, complete, consistent, and unambiguous requirements specifications remains a problem that plagues system analysts and software developers. These issues also exist in avionics systems SRE activities and have a significant impact on their quality. An important reason for the above problems is the lack of an effective knowledge sharing bridge between system developers and domain users [1]. In practical applications, some system developers lack knowledge about software system problem domains. They passively wait for domain users to provide requirements and develop requirements specifications based on their own understanding. This inevitably cause a part of the requirements to be misunderstood, resulting in the generation of low-quality requirements specifications. On the other hand, although the domain knowledge possessed by domain users and domain experts plays an important role in high-quality requirements elicitation activities, they do not know how to accurately express requirements that developers can understand in accordance with software development guidelines. The above two reasons ultimately cause the **ambiguity** of requirements elicitation. In addition, the increasing scale and complexity of software systems increase the difficulties in acquiring complete knowledge and cause the **incompleteness** of requirements elicitation. Moreover, different teams with multiple-views and multi-paradigm development methods are widely used in the development of such complex software systems, which increases the heterogeneity of software requirements specifications (SRSs) and cause the **inconsistency** of SRSs [1].

A knowledge-based requirements elicitation method can be used to solve the above problems; its purpose is to use domain analysis and experience to help software system stakeholders understand the application domain and define requirements. The key is to model the domain knowledge as a shareable knowledge framework. Under this framework, domain users can more easily and conveniently express their needs, while the domain developers can understand the requirements more accurately. Moreover, the heterogeneity

Manuscript received Dec. 2, 2019. This work was supported by the National Key Research and Development Program of China under Grant 2019YFC0118800, and a follow-up of the invited journal to the accepted & presented paper entitled “Researches on Software Requirements Elicitation approach of the Aviation Electronics Systems based on Multi-ontology” of the 22th International Conference on Advanced Communication Technology (ICACT2020).

Xuan Hu, she is a senior engineer of CEPREI Laboratory, China. (E-mail: huxuan@ceperi.com)

Jie Liu, he is the director of the Information Security Research Center of CEPREI Laboratory, China. (corresponding author, email: 47507346@qq.com)

brought about by multiple viewpoints and paradigms can be minimized. Ontology is a logic theory that explains the expected meaning of formal vocabulary, i.e., its ontological commitment to a specific conceptual world. The expected model of logic language uses the vocabulary constrained by ontology commitment. And an ontology reflects this commitment (and potential conceptualization) indirectly by approximating the expected model [2]. In the knowledge engineering community, an ontology is a formal and explicit specification of a shared conceptualization [3, 4]. Therefore, introducing the ontology method into the SRE process is an effective way to solve the above problems. By adopting the ontology method, the requirement knowledge can be expressed as an ontological concept and association; therefore, it is clear, complete, and consistent and is conducive to the sharing and reuse of knowledge. Literature [5] designed an ontology in a case study for co-simulation in a model-based system engineering tool-chain. They argued that an ontology refers to content about the types of objects, their properties, and their relationships, which represent domain-specific knowledge.

This paper proposes a requirements knowledge multi-ontology framework (RKMOF) integrating information about software requirement error pattern (SREP) to solve the problems of ambiguity, incompleteness, and inconsistency in the requirements elicitation of avionics systems software. Then, this paper uses a criteria-based evaluation approach to evaluate the quality of the software requirements elicitation ontology, including ontology validation and ontology verification. Ontology validation checks if the correct ontology has been built, whereas ontology verification checks if the ontology has been built correctly [6]. Moreover, this paper studies the software requirements elicitation method based on the RKMOF. The rest of this paper is divided into the following sections: section 2 presents the state of the art of ontology-based requirements elicitation and ontology evaluation. Section 3 describes the construction of avionics systems RKMOF and obtains the quadruple representation including the generalized ontology (GO), the domain ontology (DO), the task ontology (TO) and the application ontology (AO). Section 4 describes the composition of avionics systems software requirements knowledge ontology, including its constituent elements and hierarchical structure, presents the definition and ontology representation of SREP, and describes the evaluation of avionics systems SRE ontology in terms of ontology validation and ontology verification. Ontology validation is achieved by applying two validation methods [6]. The first is the ontology content evaluation, and the second is answering competency questions. Ontology verification is achieved using two methods, also [6]. The first is the ontology taxonomy evaluation, and the second is the improved FOCA methodology [7] adding the metrics of ontology cohesion and ontology coupling [8]. On this basis, section 5 describes the RKMOF-based SRE method. Section 6 shows the results of engineering applications.

II. RELATED WORK

The current state of the art of ontology-based requirements elicitation and ontology evaluation is presented in this section.

A. Ontology-based requirements elicitation

Nowadays the typical representatives of ontology-based requirements elicitation methods include, the requirements elicitation method based on a multi-ontology framework [9], the automatic elicitation method based on an enterprise ontology [10], the method of using a DO as the domain knowledge for requirements elicitation [11], the use of semi-formal representation of semantic guidance consisting of domain concepts, associations, and axioms to assist requirements elicitation activities [12], the use of formal ontology representation of original requirements based on the top-down refinement of meta-model to elicit and analyze requirements [13]. The first two methods mainly solve the problems of ambiguity and inconsistency in the process of requirements elicitation. The third method focuses on the ontology-based requirements elicitation and analysis, and the quality measurement of requirements document; however, it does not conduct in-depth research on how to construct a high-quality ontology. The semantic guidance proposed by the fourth method uses the concepts, associations, and axioms of a DO to provide a series of suggestions for requirements engineers to obtain requirements. The fifth method uses ontologies to elicit and analyze requirements, and studies how to evaluate the completeness and consistency of requirements to a certain extent. None of the above methods can simultaneously solve the problems of ambiguity, incompleteness, and inconsistency of requirements elicitation; thus, the scope of use and effect are limited.

Based on the above review and analysis, this paper proposes an SRE method based on a RKMOF integrating the information about SREP to solve the problems of ambiguity, incompleteness, and inconsistency simultaneously.

B. Ontology evaluation

Ontology evaluation can be defined as “a technical judgment of the content of the ontology with respect to a frame of reference during every phase and between phases of their life cycle” [14]. To achieve the best results and high-quality ontology, one needs to choose from the available list of aspects of ontology to be evaluated; the right approach to evaluation; the right mix of criteria to be evaluated; and also the right tools to be used [15].

1 ASPECTS

Aspects include the vocabulary, syntax, structure, semantics, representation, and context of the ontology, which are defined according to literature [15-17].

2 APPROACHES

The different known methods and techniques can be mainly assigned to four different kinds of approaches: technology-based, quality-attribute based, data-driven and application or task-based evaluation [15, 18]. Technology-based evaluation investigates the syntax, consistency and formal semantics and thereby ensures the correctness and usability of the ontology. Its typical representative is OOPS!, a web-based tool which is accompanied by a catalogue of potential and common pitfalls [19]. However, this approach cannot tell anything about the quality of the content and applicability of the ontology [20, 21]. Quality-based approach offers a quantitative evaluation which relies on a set of predefined metrics that measure individual quality attributes of an ontology. Yet, some of

those quality metrics tend to be hard to measure and might need human experts to evaluate [17]. Its typical representatives include, OntoClean methodology, OntoMetric [22], OntoQA [23], etc. Data-driven evaluation approach concentrates on the usability of an ontology considering its future application and has also been the current focus of recent research [24-26]. This approach attempts to analyze how adequate an ontology covers the domain but is not applicable to determine the correctness or clarity of the ontology [27, 28]. Application or task-based evaluation approach would typically involve evaluating how effective an ontology is in the context of a specific application [29]. This approach exhibits a limitation: the result obtained from one task may not be useful for another task as each task is different [18], i.e., it is not suited for a general evaluation, because every ontology must be evaluated individually depending on the application context [27].

3 CRITERIA

This kind of evaluation approach is done by humans who try to assess how well the ontology meets a set of predefined criteria, standards, requirements, etc. [30] Various criteria have been proposed in literature to evaluate the quality of ontology [15-17, 31]: consistency, completeness, accuracy, conciseness, correctness, computational efficiency, adaptability, clarity.

4 TOOLS

Various tools have been developed to support the task of ontology evaluation, each concerned with different aspects of evaluation. There exist tools for checking the consistency, the structure or modeling mistakes of the ontology [21]. Various available tools include: ODEClean, ODEval, AEON, Eyeball, Moki, XD-Analyzer, OQuaRE, OntoCheck, OntoQA, OntoClean, OntoMetric, ACTiverank, OOPS!, ODEval, oQual [15].

Based on the above review and analysis, this paper adopts a criteria-based evaluation approach to evaluate the ontology quality, including ontology validation and ontology verification.

III. AVIONICS SYSTEMS RKMOF

To achieve high-quality requirements elicitation requires various of knowledge; therefore, knowledge aided design system (KADS) can be used to study knowledge system modeling [9]. The knowledge layers in this model are clearly divided, and each layer of knowledge has good maintainability and reusability; however, its shortcomings are also obvious: the lack of strong bonds between the knowledge at all layers; incomplete knowledge. Fig. 1. shows an example of a portion of knowledge system of an avionics system, which contains elements such as classes, instances, relationships, etc. However, these are not enough for building a complete knowledge system; in addition, there is also a lack of division of knowledge layers. In general, a complete and high-quality knowledge system should include: concepts (classes), object properties, data properties, restrictions on properties, property characteristics, relationships, class hierarchy, instances, mappings and rules. Therefore, for the above knowledge model to play a role in knowledge sharing and reuse, it is also necessary to integrate relatively independent knowledge layers to form a knowledge system. Ontology method is an effective way to achieve this goal.

Ontologies can be classified into different types, according to the level of abstraction of the universe of discourse the ontology models. According to Guarino [2], there are four types of ontologies: top-level ontologies, domain ontologies and task ontologies, application ontologies. Top-level ontologies describe very general concepts which are independent of a particular problem or domain. Domain ontologies and task ontologies describe general domains. These types of ontologies specialize in the concepts introduced in the top-level ontology. Application ontologies describe concepts depending on a particular domain and task,

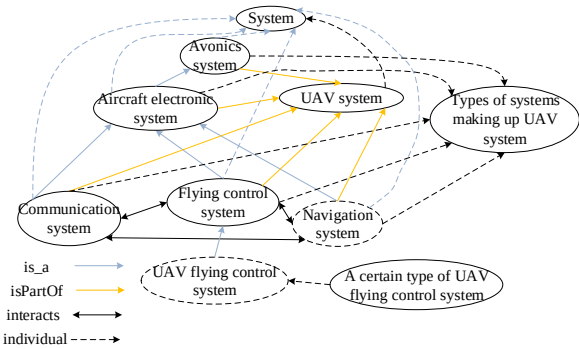


Fig. 1. Example of portion of knowledge system of avionics system

which are often specializations of both the related ontologies. On this basis, combined with existing research results [10, 32] and practical experience, this paper obtains the tuple of avionics systems RKMOF as follows:

Definition 1 RKMOF = <GO, DO, TO, AO>,

where:

GO, DO, TO and AO respectively represent the generalization ontology, domain ontology, task ontology and application ontology of avionics systems.

The relationships between various ontologies in this definition are shown in Fig. 2. A GO can be mapped to a TO, instantiated to a DO and eventually an AO. The GO can be divided into structure ontology and action ontology. Both the DO and the TO can be reused in the same domain, transformed into a domain requirements model (DRM) through domain analysis and an application requirements model (ARM) by reuse. In this framework, error data, domain knowledge and industry standards are the sources of DO. This framework is multi-viewpoint because domain experts, users and developers can all participate in the process of framework construction.

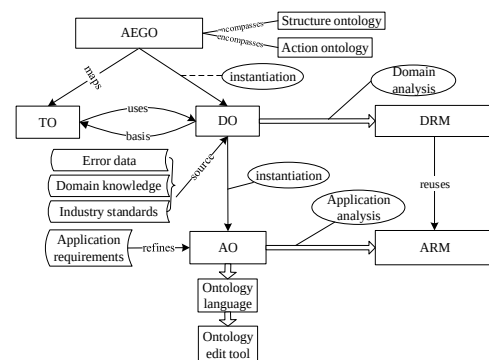


Fig. 2. Avionics systems RKMOF

IV. CONSTRUCTION OF AVIONICS SYSTEMS REQUIREMENTS KNOWLEDGE ONTOLOGY

This section presents the constituent elements of the requirements knowledge ontology of avionics systems and studies the hierarchical structure of the ontology.

A. Constituent Elements of Avionics Systems Requirements Knowledge Ontology

The universe of ontology constituent elements U is: $U = \{\text{Concepts, Object properties, Data properties, } p^R, p^C, \text{Inherit-hierarchies, Relationships, Instances, Mappings, Rules}\}$. Concepts, Inherit-hierarchies, Relationships, and Instances build the basic skeleton of a knowledge ontology. All other elements are attached to and refine the basic skeleton. Object properties and Data properties are two types of properties. The Object properties connect instances together, and the Data properties connect instances and values together. p^R indicates the restrictions on properties, including the restrictions on the type, range, and maximum/minimum number of property values. p^C indicates property characteristics. Mappings represent the mappings between different levels of ontology. Rules include axioms and custom rules. The Rules can be used to constrain information, prove correctness, or derive new information. The Rules can also be used to express richer relationships between concepts.

B. Hierarchies of Avionics Systems Requirements Knowledge Ontology

1 Avionics systems GO construction

Definition 2 Generalized ontology = $\langle \text{Concepts, Object properties, } p^C, \text{Inherit-hierarchies, Relationships, Rules} \rangle$.

It is clearly that the construction of a GO can be achieved by constructing its concepts (classes), class hierarchy, relationships, properties, and property characteristics. The specific steps are shown in Fig. 3.

- 1 Determine the scope and purpose of an ontology and clarify the relevant knowledge of an avionics system
- 2 Acquire preliminary knowledge based on expert understanding and experience, and stakeholder needs
- 3 Acquire the description of conceptual terms/properties, hierarchies and relationships through conceptualization-relationship and other data analysis
 - a Analyze the concept hierarchies of a GO
 - b Analyze the concept relationships of a GO
 - c Build the concept space of a GO
- 4 Build an initial informal ontology

Fig. 3. Specific steps of GO construction

1) GO concept classes and class hierarchy

Fig. 4. shows a portion of the class hierarchy of GO. The concept classes marked with an “*” are non-terminal concept classes, and the rest are terminal concept classes. Define an inheritance relationship as:

Definition 3 An inheritance relationship is a mechanism of a sub-class automatically sharing the properties and structure of a parent class in the GO concept class hierarchy.

Then, the sub-class and the parent class of non-terminal concept classes form an inheritance relationship. Therefore, a new class can be implemented based on an existing concept class by taking the content defined by the existing class as its own content and adding new content.

Fig. 4. also shows that the GO concepts contain both static and dynamic ontology elements.

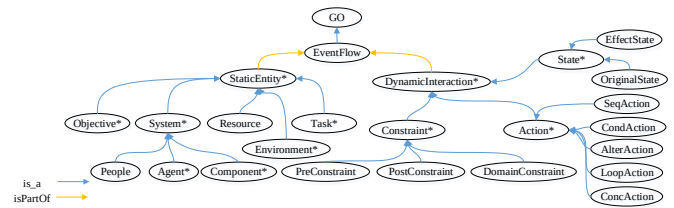


Fig. 4. Portion of the class hierarchies and relationships of GO

2) Decomposition of GO

The GO concept classes and relationships are indistinguishable representations of knowledge. This paper further divides the GO into structure ontology ($O_{\text{structure}}$) and action ontology (O_{action}), which describe the most basic structure and functions of a system, using the first-order predicate logic as: $GO = O_{\text{structure}} \cup O_{\text{action}}$.

● $O_{\text{structure}}$

Structure is a set of constraints on system actions, including a series of systems, people, agents, components, environments, static attributes, and dynamic interactions. A system has one or more tasks, and each task is defined by a series of goals that need to be facilitated. The system consumes some resources when performing actions and follows certain restrictions. The concept types of $O_{\text{structure}}$ are shown in Fig. 5.

Schema Structure	
Signature	
hasMember:	P Environment P System P Objective P Task Resource
hasSubObjective:	P Objective
hasSubTask:	P Task
System:=	People Agent Component
Environment:=	ExpEn ImpEn
has:	Task+→Objective
need:	Task+→Resource
interact:	System+→Environment
interact:	Environment+→System
Constraints	
Member	$\neq \emptyset$, Objective $\neq \emptyset$, Resource $\neq \emptyset$

Fig. 5. Concept types of $O_{\text{structure}}$

● O_{action}

An action describes the state transition of a system or an environment. An initial state triggers an action and the action generates a new state. An action is usually executed in the form of transition from an input to an output (with constraints, such as time, memory size, etc.). In addition, actions can be divided into simple actions and complex actions. The simple actions have the characteristics of the smallest granularity and the indivisibility of actions. The complex actions include sequential actions, conditional actions, alternative actions, loop actions, and concurrent actions. The concept types of O_{action} are shown in Fig. 6.

3) Concept relationships and concept space of GO

GO concept relationships can be obtained based on the class hierarchies. By integrating the concepts (classes), the class hierarchies, the concept relationships, the properties, and the property characteristics into a whole, a portion of the concept space of GO is obtained as shown in Table I. The left side of arrow is a source concept node, and the right side is a destination concept node. It can be proved that the inheritance relationship is a partial order relationship, expressed as: $a \rightarrow b$. The relationship “Interact” is a symmetric relationship.

Action := SimAction ComAction	
ComAction :=	
SeqAction CondAction AlterAction LoopAction ConcAction	
Actor := System Environment	
Schema SimAction //simple action	
Signature	
hasSubObjective: P Objective	
triggeredBy: OriginalState	
produce: EffectState	
Constraints	
hasSubObjective $\neq \phi$, operateOn $\neq \phi$	
Schema SeqAction //sequence action	
Signature	
hasSubObjective: P Objective	
subAction: SeqAction	
triggeredBy: OriginalState	
produce: EffectState	
Constraints	
hasSubObjective $\neq \phi$, subAction $\neq \phi$, predecessor $\neq \phi$, successor $\neq \phi$	
Schema CondAction //conditional action	
Signature	
hasSubObjective: P Objective	
subAction: CondAction	
triggeredBy: OriginalState	
produce: EffectState	
Constraints	
hasSubObjective $\neq \phi$, subAction $\neq \phi$, cond $\neq \phi$	
Schema AlterAction //alternative action	
Signature	
hasSubObjective: P Objective	
subAction: AlterAction	
triggeredBy: OriginalState	
produce: EffectState	
Constraints	
hasSubObjective $\neq \phi$, subAction $\neq \phi$	
Schema LoopAction //loop action	
Signature	
hasSubObjective: P Objective	
subAction: LoopAction	
triggeredBy: OriginalState	
produce: EffectState	
Constraints	
hasSubObjective $\neq \phi$, subAction $\neq \phi$	
Schema ConcAction //concurrent action	
Signature	
hasSubObjective: P Objective	
subAction: ConcAction	
triggeredBy: OriginalState	
produce: EffectState	
Constraints	
hasSubObjective $\neq \phi$, subAction $\neq \phi$	

 Fig. 6. Concept types of O_{action}

TABLE I

PORTION OF ONTOLOGY CONCEPT SPACE BASED ON STRUCTURE AND ACTION CONCEPT TYPES

Concept types	Concept relationships	Relationship value types	P^C
Structure	Objective	P Objective	partial order
	Task	P Task	partial order
	has	Task $\rightarrow$ Objective	-
	needs	Task $\rightarrow$ Resource	-
	interacts	System $\rightarrow$ Environment	symmetry
Action	interacts	Environment $\rightarrow$ System	symmetry
	triggeredby	OriginalState	-
	produces	EffectState	-

2 Avionics systems DO construction

Definition 4 A domain is a collection of systems with similar or identical functions used to address specific domain issues. When dealing with different application needs, the systems have variability.

A DO is a special ontology used to describe the knowledge of the specified domain. It presents a description of domain concepts and relationships, domain activities, and domain characteristics and laws obtained by instantiating the corresponding content in the GO.

1) Definition of DO

Definition 5 DO = <DomConcepts, Object properties, Data properties, P^R , P^C , DomInherit-hierarchies, DomRelationships, Rules, Dommappings>, where:

DomConcepts = < DomConceptname, DomConcept-ID>;
 DomInherit-hierarchies = < DomInherit-ID, Argument <Argument1 <cardinality, (status)>, Argument2 <cardinality, (status)>>>, (Argument1, Argument2 $\in$ DomConcepts);
 DomRelationships = < DomRelationships-name, DomRelationships-ID, Argument <Argument1 <cardinality, (status)>, Argument2 <cardinality, (status)>>>, (Argument1, Argument2 $\in$ DomConcepts).

DomInherit-hierarchies are the inheritance relationships of domain concept classes. An inheritance relationship is defined as a binary relationship possessing two parameters. “Cardinality” and “Status” can be used to characterize each parameter. DomRelationships refer to the relationships in the DO except the inheritance relationships. Dommappings are the generic relationships of domain concepts. They are a total function from the DomConcepts to the Concepts. They map the concept classes in the DO to the concept classes in the GO. According to this function, the equivalence relationship on the DomConcepts can be defined as:

Definition 6 $a \equiv_{\text{domain}} b$ iff Dommappings (a) = Dommappings (b) = t, a, b $\in$ DomConcepts, t $\in$ Concepts. This equivalence relationship can be recorded as $[t] \equiv_{\text{domain}}$.

2) Concepts set and concept relationships set of DO

The construction steps of DO concepts set and concept relationships set can refer to the corresponding steps of GO, and is not repeated here. This paper limits the domain to flying control (FC) systems. For the specific contents of the concepts set and concept relationships set of FC systems, please refer to the reference [32].

3 TO construction

A TO is a collection of vocabularies that describes the structure of a problem-solving method (PSM). It provides several primitives, according to which domain experts can describe the PSM context and make the process of integrating domain knowledge into the PSM context easier. Since this paper studies the requirements elicitation method based on the ontology, the definition of requirements elicitation TO (RETO) is given as follows [32]:

Definition 7 RETO = <Requirements-Eliciting Task, Requirements-Eliciting Task-PSM, Requirements-Eliciting Taskmappings>.

Specifically expressed as:

Ontology name: RETO

Type: TO

{ID: // identifier

Requirements - Eliciting Task: = <Task ID, Circumstance>,
 Requirements - Eliciting Task - PSM: = <Competence, Operational specification, Requirements>, Competence: = <Input Action, Output Action, Objective>, Operational specification: = <Inference Steps, Control Flow between the Inference Steps, Data Flow between the Inference Steps>,
 Requirements: = <Requirements - Eliciting Task Concepts, Requirements - Eliciting Task Relationships, Requirements - Eliciting Task Facts, Rules>, Requirements - Eliciting Taskmappings: {Requirements - Eliciting Task Concepts $\rightarrow$ Concepts}

4 AO construction

The definition of AO is given below:

Definition 8 AO = <AppConcepts, Object properties, Data properties, P^R , P^C , AppInherit-hierarchies, AppRelationships, Appmappings, Rules>.

The detailed descriptions of AO constituent elements are similar to the DO and are not repeated here.

Similarly, the equivalence relationship on the AppConcepts can be defined according to the Appmapping.

Definition 9 $a \equiv_{\text{app}} b$ iff Appmappings (a) = Appmappings (b) = t, a, b $\in$ AppConcepts, t $\in$ Concepts. This equivalence relationship can be recorded as $[t] \equiv_{\text{app}}$.

C. Definition and Ontology Representation of SREP

SREP is a summary of experience of errors in requirements

engineering. Therefore, the definition of SREP can be given with reference to the definition of software error pattern [33].

Definition 10 The SREP refers to the error produced in the software requirement development stage, which occurs repeatedly in a specific error lifetime scenario, spreads in the subsequent design and implementation, and may cause a system (component) to fail to perform the expected function or affect the maintainability of the system. Such errors are general and common in a specific scenario and can be corrected by various means.

The definition shows that the core components of an SREP are “scenario”, “error-manifestation” and “solution”. In addition, “severity” should also be included.

Definition 11 The definition of SREP ontology (SREPO) is: SREPO = <Concepts, Object properties, Data properties, P^R, P^C, Inherit-hierarchies, Relationships, Rules, Instances>.

The SREPO concept space is shown in Fig. 7. The concept classes marked with an “*” are non-terminal concept classes, and the rest are terminal concept classes.

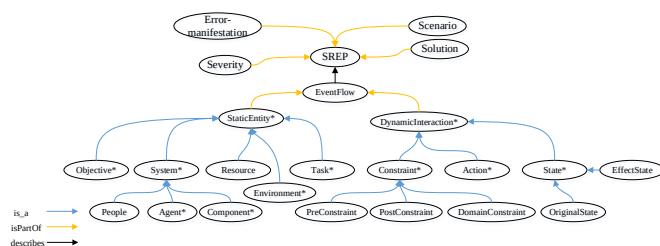


Fig. 7. SREPO concept space

This paper improves the ontology representation of the domain-related requirements error pattern proposed in [33], acquires a new ontology representation of requirements error pattern, and integrates it into the RKMOF. As a summary of experience, the role of SREP is to restrict new requirements elicitation activities in the form of rules. Since programming in logic (Prolog) can be used not only to represent factual knowledge such as states, concepts, etc., but also to represent the causality of things, i.e., rules, this paper uses the Prolog to represent the SREP. Then, this paper incorporates the SREP into a rule base.

This paper first builds a fact library based on the RKMOF, and then converts the Web Ontology Language (OWL) instances in the fact library into the facts that conform to Prolog syntax. The rules in the rule base are written in Prolog grammar, and no further conversions are required. The concepts and predicates used by these rules should also follow the definitions in the RKMOF, and maintain conceptual consistency with the fact library.

D. Ontology Evaluation

The development of ontology description languages and tools aids developers in building ontologies according to specific applications. However, due to the complexity of application semantics, ensuring ontology quality remains an important issue. In addition, the widespread use of ontologies has led to an explosive growth in the number of ontologies on the Internet. Ontologies enable reuse, but different ontologies have notable differences in domain coverage, comprehensibility, and accuracy. Thus, it is difficult for users to grasp ontology features as a whole and understand their application. Based on the above two points, it is necessary to evaluate ontology quality. According to ontology evaluation results, developers can reconstruct an ontology to optimize its structure, thereby creating high-quality ontologies.

Meanwhile, users can also select an optimal ontology between different ontology systems. It should be noted that, this paper takes an unmanned aerial vehicles (UAV) FC systems SRE ontology in the application layer of RKMOF of the avionics systems for ontology evaluation.

1 Ontology validation

1) Ontology content evaluation

This method checks the content of the ontology based on the following main criteria [15-17, 31]: consistency, completeness, accuracy, conciseness, expandability, and sensitiveness. The criteria and their compatibility to UAV FC systems SRE ontology are shown in Table II.

TABLE II
ONTOLOGY CONCEPT EVALUATION

Criteria	Satisfaction
Consistency	Yes, since no contradictory knowledge can be inferred from all definitions and axioms. Also, reasoner shows no errors.
Completeness	Yes, it is complete based on specifications determined in the design phase of the ontology.
Accuracy	Yes, the activity of interviewing experts has been conducted. And experts participate in the ontology construction process.
Conciseness	Yes, the ontology is concise since does not contain any unnecessary concepts.
Expandability	Yes, it is easily expanded since there is no need to make big changes in a set of well-defined definitions when adding new definitions.
Sensitiveness	The ontology is not sensitive since small changes in definition will not alter a set of well-defined contents.

2) Competency questions evaluation

The competency questions for determining the scope and designing purposes of UAV FC systems SRE ontology are used here for the evaluation. Each competency question is answered and justified based on the UAV FC systems SRE ontology components. Answers and justifications are shown in Table III. Competency questions ensure that the ontology implementation fulfills the scope of UAV FC systems SRE ontology.

TABLE III
COMPETENCY QUESTION ANSWERS AND JUSTIFICATIONS

CQ#	CQ	Justification
CQ1	Why build a UAV FC systems SRE ontology?	Engineering experience has shown that the avionics systems software usually has the characteristics of complex structure, wide function coverage, high reliability and high safety. These characteristics have led to the increasing scale and complexity of software systems, and the increasing variety of research & development (R&D) personnel. This further intensify the knowledge-intensive trend of software development, make the software requirements elicitation activities more complicated, and also cause the issues of ambiguity, incompleteness, and inconsistency. An ontology is an explicit specification of a conceptualization. Therefore, introducing the ontology method into the SRE process is an effective way to solve the above problems. By adopting this ontology method, the requirement knowledge is expressed as the ontology concept and association; therefore, it is clear, complete, and consistent and is conducive to the sharing and reuse of knowledge. In addition, it greatly promotes the quality of UAV FC systems SRE and SRSs.
CQ2	Who needs this ontology?	The stakeholders, including system analysts/software developers, customers/users and domain experts, all need this ontology. The purpose of an ontology is to use domain

		analysis and experience to model domain knowledge as a shareable knowledge framework to help stakeholders understand the application domain and define requirements. Under this framework, system analysts and software developers can understand the requirements accurately, and the heterogeneity brought about by multiple viewpoints and paradigms can be minimized. Moreover, the implementation of a higher quality UAV FC systems SRE activity based on this ontology benefits the development of UAV FC systems software and the entire system, helps to obtain products with fewer defects and higher quality, and enables developers to shorten the development cycle and reduce costs in a highly competitive market. Meanwhile, because the domain knowledge possessed by customers, users, and domain experts plays an important role in high-quality SRE, this ontology can help them accurately express requirements that developers can understand in accordance with software development guidelines and improve system development quality and efficiency.
CQ3	Who is responsible for managing and maintaining this ontology?	This ontology exists as a part of UAV FC systems software and system development. Software and system developers are responsible for the management and maintenance of ontologies. In addition, there is a dedicated team in the development team responsible for the initial ontology development and subsequent ontology management and maintenance.
CQ4	What are the main contents in this ontology?	Contents include the GO, DO, and AO level ontology-related concepts, properties, hierarchies, and relationships. These are specifically based on: 1) the UAV FC systems software-related concepts and relationships, 2) the SREP-related concepts and relationships.
CQ5	When is this ontology needed?	This ontology is needed when developing the same or similar UAV FC software system, which can realize the sharing and reuse of the ontology.
CQ6	How is this ontology managed and maintained?	A dedicated team is responsible for the management and maintenance of the ontology. According to actual usage and user feedback, the ontology is continuously updated, including adding new necessary content and deleting outdated content. In addition, logs are used to record the management and maintenance process.

2 Ontology verification

1) Ontology taxonomy evaluation

The taxonomy evaluation method is used for checking the taxonomy of the ontology based on main criteria mentioned in [34]. These criteria and their compatibility to UAV FC systems SRE ontology are shown in Table IV.

TABLE IV
ONTOLOGY TAXONOMY EVALUATION

Criteria	Satisfaction	
Inconsistency	Circularity errors	No error, reasoner shows no errors
	Partition errors	No error, reasoner shows no errors
	Semantic errors	No error, reasoner shows no errors
Incompleteness	Incomplete	No error, all concepts of the knowledge specified in the design phase are included.
	Partition errors	No error, because all the instances of the base classes belong to the sub classes.
Redundancy	Grammatical redundancy	No error, each class has only one definition.
	Identical formal	No error, there is no two

definition of some classes	classes with the same definition.
Identical formal definition of some instances	No error, in domain ontology there are no instances

2) Improved FOCA evaluation

● FOCA method and its shortcomings

FOCA is a method that can be used for evaluating the quality of an ontology. FOCA includes determining the type of ontology, a questionnaire to evaluate the components, a framework to follow, and a statistical model that calculates the quality of the ontology. FOCA goes through three verification steps, as shown in Fig. 8. [7]. Ontology type verification defines two types of ontology: a domain or task ontology and an application ontology. Questions verification possesses questions to serve the goals. Quality verification calculates the ontology quality.

The real world is a complex network system, and an ontology is the result of modeling the real world. Therefore, the ontology can also be regarded as a complex network. It is an abstract network model of the real world; however, for the purpose of knowledge sharing and reuse, the network is expressed as a specific network structure using semantic standards such as RDF / OWL and other languages. Newman, a famous scholar of complex networks, argued that modularity should be one of the basic characteristics of

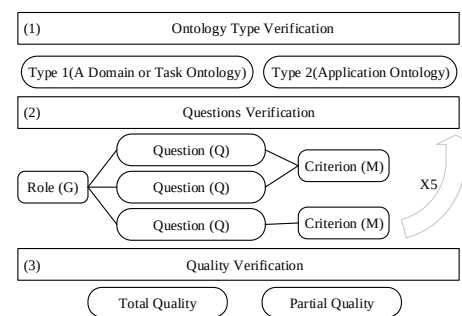


Fig. 8. FOCA method

complex networks [35]. Therefore, the evaluation of ontology quality should also consider ontology modularity. The notion of an ontology module relates to the design of ontology composed of well-defined components that can be managed and reused independently. An ontology module is used to reduce ontology complexity and to increase understandability, testability, maintainability, and reliability. The FOCA evaluation criteria do not include a quantitative evaluation of ontology modularity. This paper adds the quantitative indicators reflecting the degree of ontology modularization to the FOCA to evaluate the ontology quality.

● Ontology module and directed acyclic graph

An ontology module is a collection of the closely related concepts, relationships, and axioms reflecting a common theme. The ontology module is divided or extracted from an original ontology and is part of the original ontology (a sub-ontology) [8]. The sub-ontology needs to meet certain indicators or conditions to satisfy specific applications. The modularization of an ontology helps reduce complexity and enhances comprehensibility, testability, maintainability, and reliability. The structure of an ontology module is consistent with the structure of a software module in the software engineering and should meet the principle of “high cohesion, low coupling”.

Ontology classes are arranged in a hierarchy from the general (high in the hierarchy) to the specific (low in the hierarchy). Despite the hierarchical organisation, most ontologies are not simple trees. Rather, they are structured as directed acyclic graphs (DAGs). This is because it is possible for classes to have multiple parents in the classification hierarchy, and furthermore ontologies include additional types of relationships between entities other than hierarchical classification (which itself is represented by *is_a* relations). All relations are directed and care must be taken by the ontology editors to ensure that the overall structure of the ontology does not contain cycles, as illustrated in Fig. 9. [37].

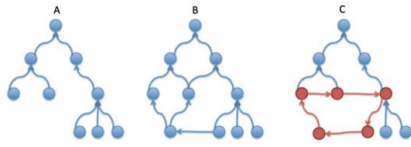


Fig. 9. (A) simple hierarchical tree, (B) DAG, (C) graph that contains a cycle, indicated in red.

● Cohesion and coupling of ontology

Cohesion and coupling are two important indicators to reflect the ontology modularity. Therefore, this paper selects these two indicators to reflect the ontology modularity, and then takes them as an aspect of ontology quality evaluation. The structure of an ontology is consistent with object-oriented structure and should also meet the principle of “high cohesion, low coupling”. The cohesion describes the degree to which multiple concepts are combined into one module. It reflects the closeness of concepts. The higher the cohesion of an ontology module is, the closer relationship between the concepts and the more similar semantics of the concepts in the ontology module. This benefits the understanding, reuse, and maintenance of an ontology. Ontology module coupling can be regarded as the degree of correlation between ontology modules. The classes in one module are closely related to the classes in other modules, then the first ontology module has a high coupling value. The stronger the ontology module coupling is, the harder to understand, change, and correct the ontology. And it increases the complexity of the systems using the ontology.

In short, the cohesion reflects the closeness of the relationship between the concepts in an ontology module, and the coupling reflects the closeness of the connection between the ontology modules. Both reflect the characteristics of ontology structure and semantics, and should also be fully considered during the ontology quality evaluation and added to the FOCA method to achieve its improvement.

● Evaluation metrics of cohesion and coupling

In this paper, the metrics of cohesion Coh [8], and AOC [38], and the metrics of coupling NSHR [8], and NSNR [8] are used.

❖ $Coh(M)$ is the ratio of the sum of the strength of relation to the number of all possible relations in a module M .

$$Coh(M) = \begin{cases} \frac{\sum_{c_i \in M} \sum_{c_j \in M} \frac{sr(c_i, c_j)}{|M|(|M|-1)}}{2} & \text{if } |M| > 1 \\ 1 & \text{otherwise} \end{cases} \quad (1)$$

$$sr(c_i, c_j) = \begin{cases} \frac{1}{distance(c_i, c_j)} & \text{if relations exist between } c_i \text{ and } c_j \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where,

$distance(c_i, c_j)$ = a minimal path length between c_i and c_j . For class $c_i \in M$, $c_j \in M$, M : Module, $| \cdot |$: cardinality. If there is no

concept in M , then $Coh(M)=0$. If there is only one concept in M , $Coh(M)=1$ because this concept does not depend on any other concept, and it is the closest structure. The range of $Coh(M)$ is $[0, 1]$ because the largest relation number in a DAG is the number of edges of the full connected graph.

❖ AOC is the original ontology cohesion.

$$AOC = \frac{\sum_{i=1}^n Coh(M_i)}{n} \quad (3)$$

where,

n is the number of modules partitioned by the original ontology, and $Coh(M_i)$ is the cohesion of the ontology module M_i .

❖ An ontology has two types of relationships: hierarchical relationships and non-hierarchical relationships. The hierarchical relationships are stronger than other relationships, because the classes associated through the hierarchical relationships share and inherit more information between classes. This paper measures the coupling based on NSHR and NSNR. Using the proposed ontology coupling metrics, the consistency between modules and their original ontology should be considered. To be consistent with the original ontology, modules should preserve the classes and axioms of their original ontology. Thus, an ontology should be modularized to lessen the number of disconnected relations.

NSHR is the ratio of the number of hierarchical relations that are disconnected after modularization to the total number of relations. More disconnected hierarchical relations mean that more information about the hierarchical relation is lost. NSHR can be formulated as follows:

$$NSHR(M) = \sum_{c_i \in M} \sum_{c_j \in O-M} nshr(c_i, c_j) \quad (4)$$

O is the original ontology, M is the module, and $-$ is the difference operation. $nsnr(c_i, c_j)$ is the number of hierarchical relations between c_i and c_j that is disconnected after modularization.

NSNR is the ratio of the number of disconnected non-hierarchical relations to the total number of relations after ontology modularization. NSNR can be formulated as follows:

$$NSNR(M) = \sum_{c_i \in M} \sum_{c_j \in O-M} nsnr(c_i, c_j) \quad (5)$$

O is the original ontology, M is the module, and $-$ is the difference operation. $nsnr(c_i, c_j)$ is the number of non-hierarchical relations between c_i and c_j that is disconnected after modularization.

In the following example, an original ontology O (a UAV FC systems SRE ontology) is partitioned into five modules, $M1$, $M2$, $M3$, $M4$, and $M5$, as shown in Fig. 10. This figure explicitly shows the disconnected relations between modules for explanation.

❖ $r_{NSHR}(M)$ is the ratio of the NSHR of ontology module M to the total number of hierarchical relationships in the corresponding part of original ontology. The calculation formula is as follows:

$$r_{NSHR}(M) = \frac{NSHR(M)}{K} \quad (6)$$

where,

K is the total number of hierarchical relationships in the corresponding part of original ontology.

$r_{NSNR}(M)$ is the ratio of the NSNR of ontology module M to the total number of non-hierarchical relationships in the

corresponding part of original ontology. The calculation formula is as follows:

$$r_{NSNR}(M) = \frac{NSNR(M)}{L} \quad (7)$$

where,

L is the total number of non-hierarchical relationships in the corresponding part of original ontology.

❖ $Aver_{r_{NSHR}}$ represents the average value of $r_{NSHR}(M_i)$ of all ontology modules after the division of original ontology.

$Aver_{r_{NSNR}}$ represents the average value of $r_{NSNR}(M_i)$ of all ontology modules after the division of original ontology.

The calculation formula is as follows:

$$Aver_{r_{NSHR}} = \frac{r_{NSHR}(M_i)}{n} \quad (8)$$

$$Aver_{r_{NSNR}} = \frac{r_{NSNR}(M_i)}{n} \quad (9)$$

where,

n is the number of modules after the division of original ontology.

❖ $S_{coupling}$ is the comprehensive coupling value of a module.

The calculation formula is as follows:

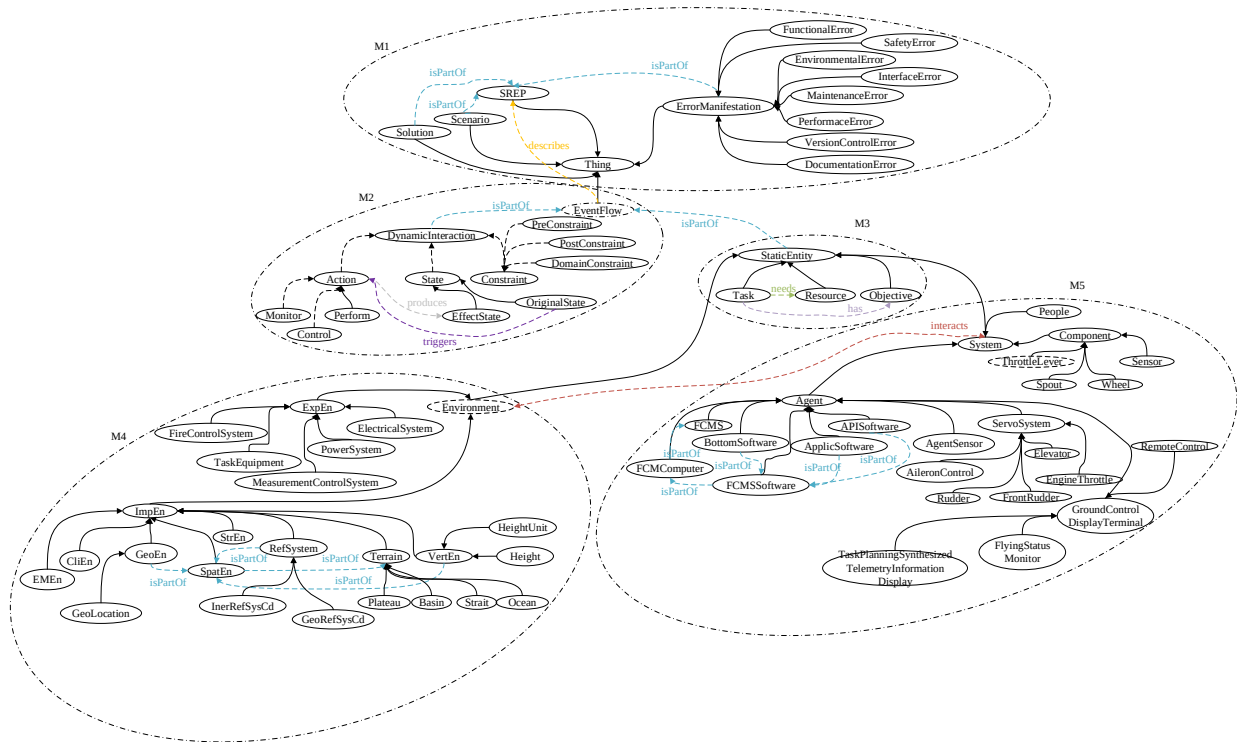


Fig. 10. Ontology modules M1~M5

$$S_{coupling} = \alpha * Aver_{r_{NSHR}} + \beta * Aver_{r_{NSNR}} \quad (10)$$

where,

α represents the weight of $Aver_{r_{NSHR}}$, β represents the weight of $Aver_{r_{NSNR}}$, $\alpha + \beta = 1$. This paper argues that the importance of hierarchical relationships is greater than that of non-hierarchical relationships. Generally speaking, $\alpha > \beta$. The values of α and β can be set according to experience.

● Calculation of values of cohesion and coupling

In this paper, the values of cohesion and coupling are calculated based on the ontology module division shown in Fig. 10.

❖ Calculation of cohesion value

Using (1) for M1 to M5, Coh(M1)=0.14, Coh(M2)=0.23, Coh(M3)=0.83, Coh(M4)=0.14, Coh(M5)=0.14. Substituting these values into (3), AOC=0.296. It should be noted that the AOC value is low, indicating that the relationship between the concepts in the ontology is not very close. The main reason for the above results is that the hierarchical structure of the conceptual classes in the ontology is not sufficiently complete; meanwhile, there are

limited connections between the concepts other than the hierarchical relationships.

❖ Calculation of coupling value

Using (4) to (9) in turn for M1 to M5, the values of NSHR, NSHR, $r_{NSHR}(M)$, $r_{NSNR}(M)$, $Aver_{r_{NSHR}}$, and $Aver_{r_{NSNR}}$ of each module can be obtained as Table V. Let $\alpha = 0.7$, $\beta = 0.3$, Substituting the values of $Aver_{r_{NSHR}}$ and $Aver_{r_{NSNR}}$ into (10), $S_{coupling} = 0.1706$. According to experience, this value is less than 0.2, indicating that the coupling is not high, and the consistency between the ontology modules and the original ontology is good. This also shows that the correlation between ontology modules is not high, and the connection is loose. The concepts of each module focus on its own theme; therefore, it is easier to understand and reuse the ontology module.

TABLE V
COUPLING METRICS FOR M1 TO M5

No.	NSHR	$r_{NSHR}(M)$	NSNR	$r_{NSNR}(M)$
M1	1	0.077	1	0.25
M2	1	0.083	2	0.4
M3	2	0.4	1	0.33
M4	1	0.04	1	0.2
M5	1	0.04	1	0.17

$Aver_{rNSHR}$	0.128	$Aver_{rNSNR}$	0.27
----------------	-------	----------------	------

● Improved FOCA evaluation method

This paper uses the improved FOCA to evaluate the ontology quality.

❖ Ontology type verification

FOCA defines two types of ontology, a domain or task ontology and an application ontology. The UAV FC systems SRE ontology is an AO (type 2); therefore, a type 2 ontology should answer Q4 instead of Q5 for Goal 2 shown in Table VI.

❖ Questions verification

When a cohesion metric and a coupling metric are added, it needs to answer the 13 questions in Table VI (should answer Q4 instead of Q5). These answers should then be scored by the evaluator. The set of questions corresponding to Goal 2 is expanded by adding two questions of “**Was the ontology cohesion metric value acquired?**” and “**Was the ontology coupling metric value acquired?**” The scores partly refer to the experimental data in [38]. The cohesion metric values and corresponding scores are shown in Table VII. The coupling metric values and corresponding scores are shown in Table VIII. These 13 questions serve five goals. The goal/question/metric (GQM) approach for the improved FOCA is shown in Table VI.

TABLE VI
GQM FOR IMPROVED FOCA

Goal	Question	Metric
1. Check if the ontology complies with Substitute.	Q1. Were the competency questions defined?	Does the document define the ontology objective? Does the document define the ontology stakeholders? Does the document define the use of scenarios?
	Q2. Were the competency questions answered?	Completeness
	Q3. Did the ontology reuse other ontologies?	Adaptability
	Q4. Did the ontology impose a minimal ontological commitment? (application ontology)	Conciseness
2. Check if the ontology complies Ontological Commitments.	Q5. Did the ontology impose a maximum ontological commitment? (domain/task ontology)	Conciseness
	Q6. Are the ontology properties coherent with the domain?	Consistency
	Q7. Was the ontology cohesion metric value acquired?	Cohesion
	Q8. Was the ontology coupling metric value acquired?	Coupling
3. Check if the ontology complies with Intelligent Reasoning.	Q9. Are there contradictory axioms?	Consistency
	Q10. Are there redundant axioms?	Conciseness
4. Check if the ontology complies Efficient Computation.	Q11. Did the reasoner bring modelling errors?	Computational efficiency
	Q12. Did the reasoner perform quickly?	Computational efficiency
5. Check if the ontology complies with Human Expression.	Q13. Is the documentation consistent with modelling?	Are the written terms in the documentation the same as the modelling? Does the documentation explain what each term is and does it justify each detail of modelling?
	Q14. Were the concepts well written?	Clarity
	Q15. Are there annotations in the ontology that show the definitions of the concepts?	Clarity

TABLE VII

COHESION METRICS VALUES AND CORRESPONDING SCORES

M	M<0.25	0.25≤M<0.5	0.5≤M<0.75	0.75≤M<1
score	25	50	75	100

TABLE VIII

COMPREHENSIVE COUPLING METRICS VALUES AND CORRESPONDING SCORES

N	N<0.25	0.25≤N<0.5	0.5≤N<0.75	0.75≤N<1
score	100	75	50	25

❖ Quality verification

Ontology quality can be calculated in two ways: total quality and partial quality. This paper uses the total quality verification because most goals are considered in the evaluation. Total quality verification is calculated using beta regression models, proposed by Ferrari [39], and shown in (11).

$$\hat{\mu}_i = \frac{\exp\{-0.44 + 0.03(Cov_S \times Sb)_i + 0.02(Cov_C \times Co)_i + 0.01(Cov_R \times Re)_i + 0.02(Cov_{Cp} \times Cp)_i - 0.66LExp_i - 25(0.1 \times NI)_i\}}{1 + \exp\{-0.44 + 0.03(Cov_S \times Sb)_i + 0.02(Cov_C \times Co)_i + 0.01(Cov_R \times Re)_i + 0.02(Cov_{Cp} \times Cp)_i - 0.66LExp_i - 25(0.1 \times NI)_i\}} \quad (11)$$

where,

Cov_S is the mean of the grades from Goal 1;

Cov_C is the mean of the grades from Goal 2;

Cov_R is the mean of the grades from Goal 3;

Cov_{Cp} is the mean of the grades from Goal 4;

$LExp$ is the variable for evaluator experience, with 1 being very experienced and 0 being not experienced at all;

NI is 1 only if some Goal is impossible for the evaluator to answer all the questions;

$Sb=1$, $Co=1$, $Re=1$, $Cp=1$, because the total quality considers all the roles.

By Substituting these values into (11), the following result is obtained,

$$\hat{\mu}_i = \frac{\exp\{-0.44 + 0.03 \times (100 \times 1) + 0.02 \times (87.5 \times 1) + 0.01 \times (100 \times 1) + 0.02 \times (87.5 \times 1) - 0.66 \times 1 - 25 \times (0.1 \times 1)\}}{1 + \exp\{-0.44 + 0.03 \times (100 \times 1) + 0.02 \times (87.5 \times 1) + 0.01 \times (100 \times 1) + 0.02 \times (87.5 \times 1) - 0.66 \times 1 - 25 \times (0.1 \times 1)\}} = 0.9802$$

The total quality of the ontology is 0.9802, which is near to 1. This shows the high quality of the UAV FC systems SRE ontology.

V. RKMOF-BASED REQUIREMENTS ELICITATION APPROACH

In addition to the basic knowledge required for the SRE, the avionics systems RKMOF established in this paper also contains the SREP knowledge, which greatly enriches the prior information of SRE; therefore, the knowledge is relatively complete. The ontology quality evaluation result

shows that the UAV FC systems SRE ontology has high quality.

Fig. 11. shows the schematic diagram of the ontology-based SRE method. It can be seen that each main step of elicitation process has two branches: If there is no existing ontology, an ontology is first established; if there is an existing ontology, it is reused and tailored / expanded according to actual needs. By building ontologies layer-by-layer, the corresponding concepts and relationships are obtained. A DRM can be obtained through the domain analysis on a DO and an ARM can be obtained through the application analysis on the DRM and an AO. The following are the specific steps of the ontology-based SRE method.

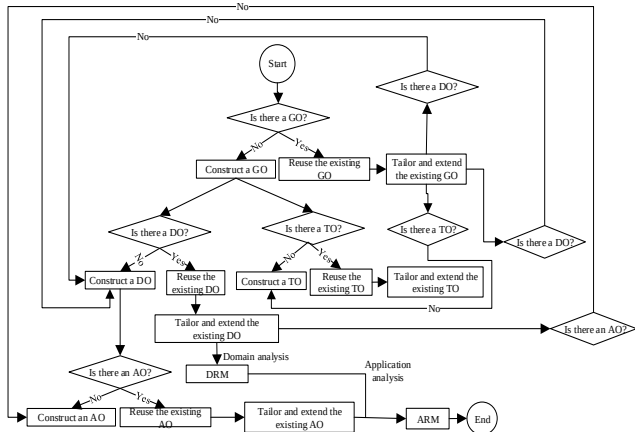


Fig. 11. Schematic diagram of SRE method based on multi-ontology

A. Preliminary stage

The accumulation of experience is crucial to the ontology-based SRE method. Therefore, before formal SRE, it is necessary for testers and domain experts to jointly build a SREP library as shown in Fig. 12.

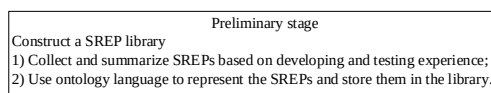


Fig. 12. Specific steps of preliminary stage

B. Implementation stage

Fig. 13 shows the specific steps of implementation stage of the ontology-based SRE method.

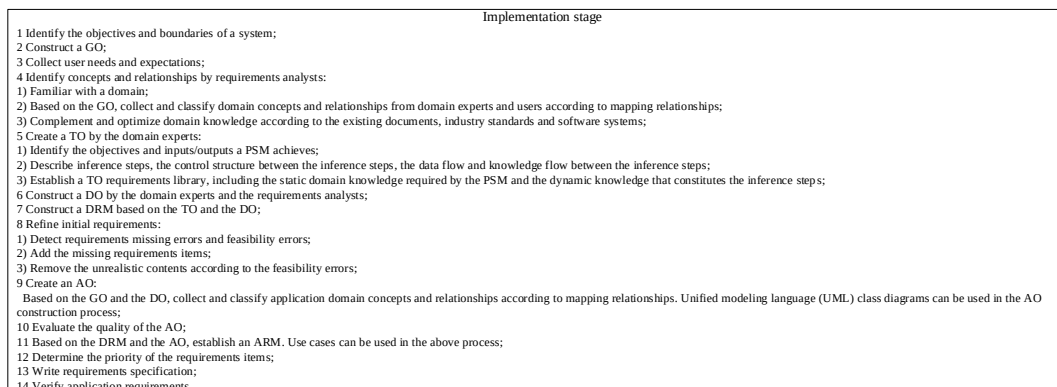


Fig. 13. Specific steps of implementation stage

The steps 1 to 10 in Fig. 13 are introduced earlier; however, it is insufficient for the actual application to build an AO only, because its granularity cannot satisfy the requirements of application software specification, and is indirect. Unified modeling language (UML) has been widely used and has become a standard. Therefore, UML class diagrams, use case diagrams and activity diagrams are used in this paper to assist AO modeling and application requirements eliciting to obtain detailed application software specification.

1 Application of class diagrams

An AO can be constructed by eliciting and classifying the concepts and relationships in an application domain. Because an AO reflects the static knowledge of a system, the UML class diagrams can be used in the process of AO construction. Fig. 14. shows a portion of concept classes (hierarchies) of the UAV FC systems SRE ontology.

2 Application of use case diagrams

An ARM can be established based on a DRM and an AO. Because an ARM reflects the dynamic knowledge of a system, the UML use case diagrams and activity diagrams can be used in the process of ARM construction. The specific steps are:

- 1) Determine the boundaries of use cases;
- 2) Determine scenarios, including: preconditions, postconditions and steps;
- 3) Use the use case diagrams and activity diagrams to represent the use cases.

The use case diagrams reflect the system functions that users can observe; therefore, they are relatively stable. An example of a use case diagram is shown in Fig. 15.

3 Application of activity diagrams

The activity diagrams can be used to model the workflow between different components, and describe the conditions of use cases and system states; therefore, they are variable. An example of an activity diagram is shown in Fig. 16.

As shown in Fig. 16, an alternative path is represented by a diamond. The activity diagrams can represent parallel expressions; therefore, the same actions and control flows can be represented in the RETOs. The RETOs describe the objectives, inference steps and control flows of requirements elicitation activities. Therefore, the activity diagrams can be transformed into the RETOs. UAV FC systems software communication link interruption processing RETO is shown in Fig. 17.

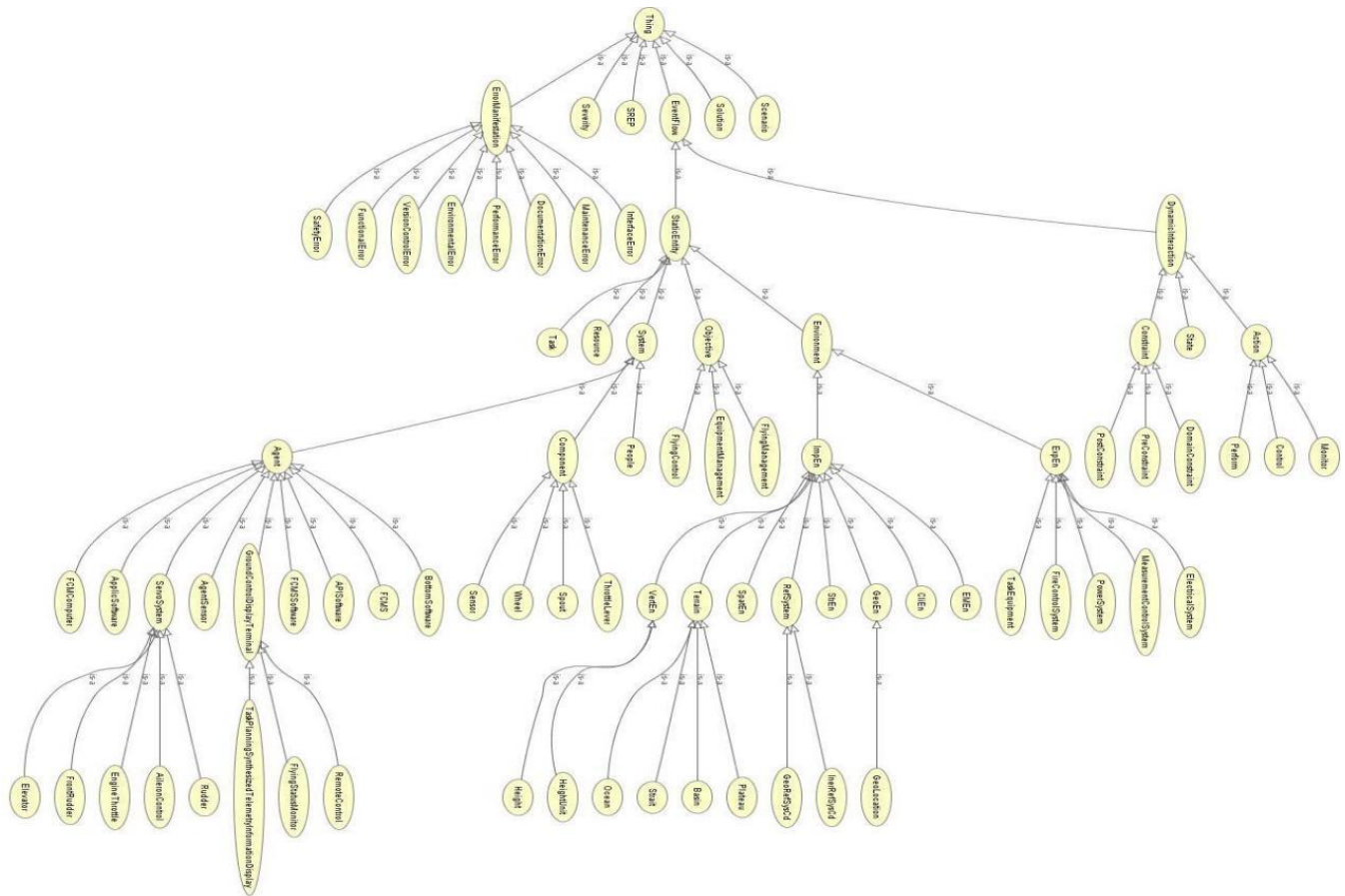


Fig. 14. Portion of concept classes (hierarchies) of UAV FC systems SRE ontology

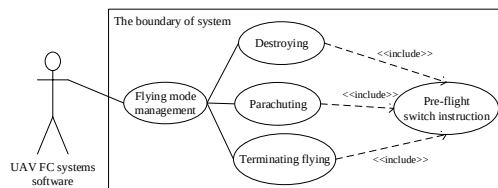


Fig. 15. Example of use case diagram of UAV FC systems software

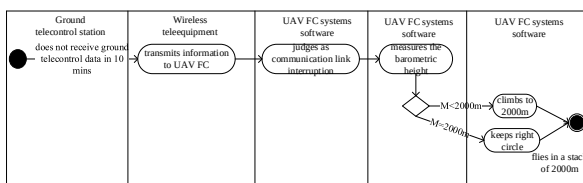


Fig. 16. Example of activity diagram of UAV FC systems software

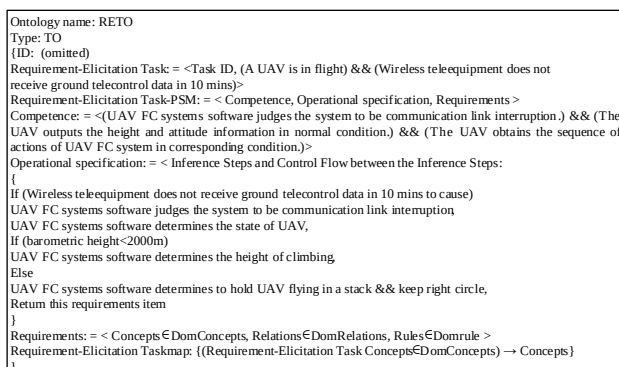


Fig. 17. UAV FC systems software communication link interruption processing RETO

4 Application of SREP

The use of several types of UML diagrams in normal scenarios are introduced earlier; however, abnormal scenarios also affect systems in practical applications.

Since an SREP library is constructed in the preliminary stage, which contains many historical error data; therefore, the SREP can be used to study the SRE in abnormal scenarios.

- 1) Map an SREP from the GO layer to the AO layer.
 Putting the SREP in the GO layer is too abstract to use; therefore, it is necessary to map it to the AO layer. The specific steps are based on the inverse mapping of Appmappings in the AO definition.
- 2) Generate the use cases corresponding to the non-functional requirements in abnormal scenarios based on the SREP.

The following takes the safety requirements of non-functional requirements as an example to illustrate the steps of use cases development:

- Step a): The concerns of software requirements are divided into general concerns and cross-cutting concerns. This paper treats functional requirements as the general concerns and the non-functional requirements as the cross-cutting concerns.
- Step b): All cross-cutting concerns called by each general concern are confirmed.
- Step c): UML use case modeling technology is used to obtain the functional requirements according to the step 2) in 2. This paper uses Rational Rose, a UML modeling tool, to describe the use cases based on the scenarios, including the preconditions, the postconditions, and the steps. Then, the scenarios are further divided into the general scenarios corresponding to the functional requirements and the cross-cutting scenarios corresponding to the non-functional requirements (the safety requirements).

A scenario in an activity diagram is defined as:

Definition 12 A scenario in an activity diagram is a sequence of events describing agent actions, the changes in the state of environment, and the interactions between the agents and the environment.

The description template of a scenario segment is shown in Fig. 18.

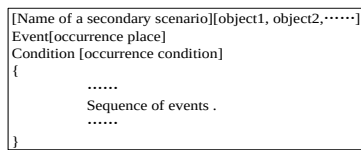


Fig. 18. Description template of scenario segment

- Step d): Rational Rose is used to build the cross-cutting scenarios, and further generate a set of cross-cutting use cases based on the SREP library.
- Step e) The step b) is executed iteratively, and the cross-cutting use cases corresponding to the cross-cutting concerns called by each general concern are integrated into the general scenario corresponding to the functional requirements in turn according to the calling relationships.

An example of use case development is shown in Fig. 19. according to the Step a) to e).

VI. CASE STUDIES

In this paper, a part of a UAV FC system is selected for ontology modeling. The effectiveness of the ontology-based SRE method is verified by a comparative experiment. Since several continuous versions of the software are developed after the ontology modeling, the research selects two continuous versions of the software and adopts a conventional method for the requirements elicitation of a version 3.3.x; after a defined period, the requirements of a version 3.3.(x+1) are elicited based on the ontology-based method this paper proposes. Table IX records the detected SREP error-manifestations and number distributions of these two SRSs by a requirement inspection. The same group of inspectors is used to conduct the comparative experiment. SRS I is developed based on the conventional method, and SRS II is developed based on the ontology-based method proposed in this paper.

The results that the total number of errors in SRS I is much higher than in SRS II. In addition, the severity of the errors detected in SRS I is higher, and they occur in more significant error types such as functional errors, interface errors, safety errors, and environmental errors. The direct cause of the above results can be initially identified as SRS II using the ontology-based method; SRS I uses the conventional method.

Intuitively, because the ontology is a complete set of domain knowledge, the quality information of SRS can be obtained indirectly by considering the correspondence between the SRS and ontology element. This paper evaluated the quality of SRS based on the metrics in [40]. The quality metric results of the SRS I and SRS II are shown in Table X.

TABLE IX
SREP ERROR-MANIFESTATIONS AND NUMBER DISTRIBUTIONS

Error-manifestations	SRS I				SRS II			
	C	I	A	T _t	C	I	A	T _t
Functional errors	4	1	1	6	0	1	1	2
Performance errors	0	0	1	1	0	0	1	1
Interface errors	3	6	0	9	1	2	0	3
Safety errors	4	1	0	5	1	0	0	1
Environmental errors	2	2	0	4	0	0	0	0
Maintenance errors	0	1	1	2	0	0	2	2
Documentation errors	0	3	4	7	0	2	4	6
T _s	13	14	7	-	2	5	8	-
T	34				15			

Notes: C—"critical", I—"important", A—"average". T_t—"total number of errors in each type", T_s—"total number of errors in each severity", T—"total number of errors".

TABLE X
QUALITY METRIC RESULTS OF SRS I AND SRS II

Metrics	SRS I	SRS II
Correctness	23/29=79.3%	28/29=96.6%

From the results, the difference between SRS I and SRS II in "Correctness" is more obvious. An ontology is a semantic basis for building a specific problem domain. Ideally, all requirements items should be able to find the corresponding elements in the ontology. (the number of items that can be mapped to the ontology / the total number of requirements items) can reflect the proportion of the mapped elements. The higher the ratio is, the higher the SRS quality. This ratio of SRS I to SRS II is significantly lower. This shows that some of the requirements items of SRS I are not included in the ontology library, implying nonconformity with the actual application. This fact also explains the results of the requirement inspection in Table IX. Therefore, the requirement knowledge ontology has a major impact on the entire requirement development process. In general, the quality of the SRS obtained based on the proposed ontology is higher than the quality of the SRS obtained based on the conventional method. Therefore, the ontology elements, i.e., the knowledge elements, should be fully integrated in the early stage of the requirement development process.

VII. CONCLUSION

The RKMOF proposed in this paper fully considered the needs of complete knowledge basis for requirements elicitation activities, adopted various ontologies to reflect all aspects of requirements knowledge, divided and stored knowledge hierarchically, and blended static knowledge with dynamic knowledge to make them a whole. At the same time, the empirical knowledge such as SREP was integrated into RKMOF in a consistent form. The framework is universal and suitable for any avionics system. Based on the framework, the construction of DRMs and ARMs can be further realized to guide requirements elicitation activities to obtain SRSs with fewer defects, thereby improving the quality of SRSs and further improving the reliability of software systems.

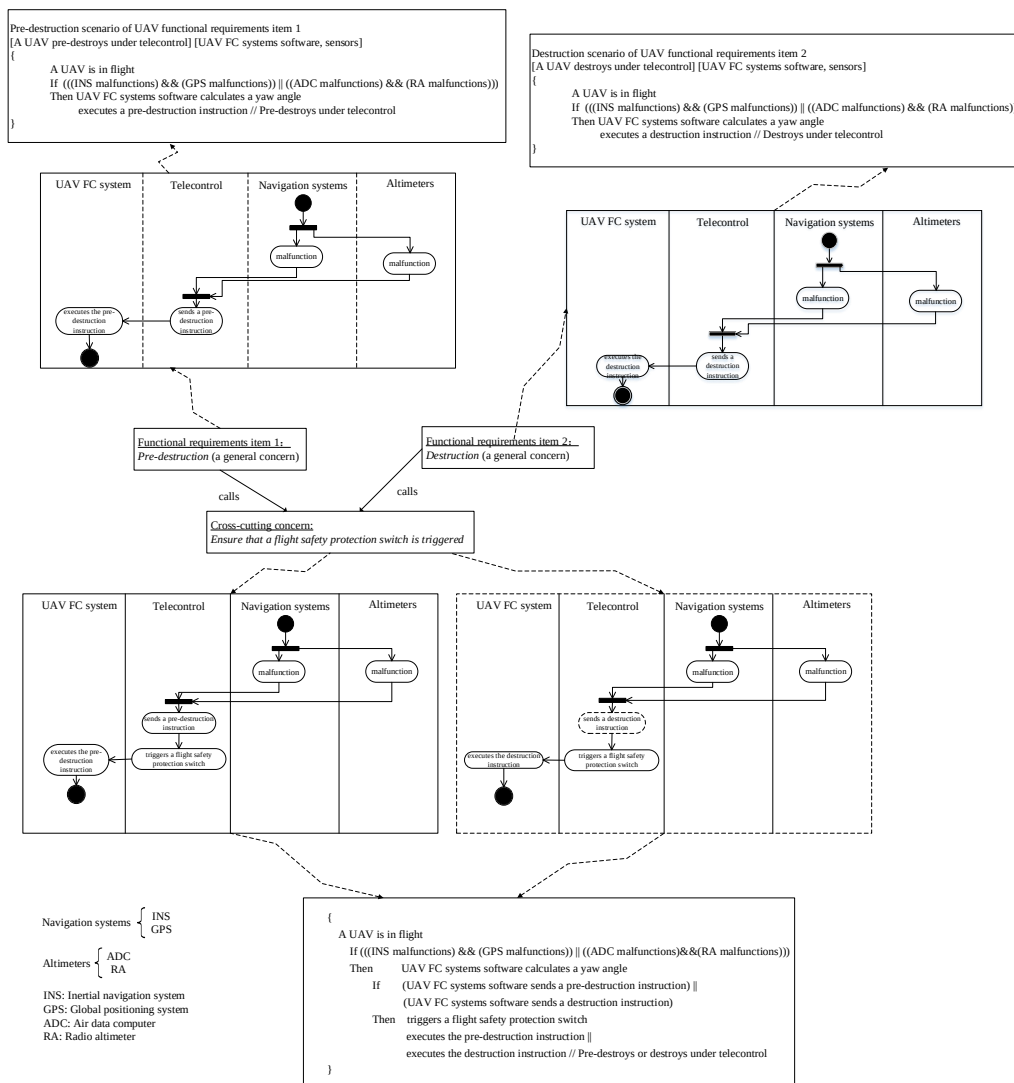


Fig. 19. Example of use case development

REFERENCES

- [1] Xuan Hu, "Study on Ontology-based Avionics Systems Requirements Knowledge Model Construction and Software Requirements Elicitation Method," *Ph.D. dissertation, Dept. of reliability and systems engineering, Beihang University*, Beijing, China, 2009.
- [2] Guarino, N., "Formal ontology in information systems," *Proceedings of the first international conference (FOIS'98)*, Trento, Italy, 1998.
- [3] Cf. T. R. Gruber, "A translation approach to portable ontologies," *Knowledge Acquisition*, 5(2):199-220, 1993.
- [4] Thomas R. Gruber, "Toward Principles for the Design of Ontologies Used for Knowledge Sharing," *International Journal of Human-Computer Studies*, vol. 43, Issues 5-6, pp. 907-928, 1995.
- [5] Jinzhi Lu, Guoxin Wang and Martin Törmgren, "Design Ontology in a Case Study for Cosimulation in a Model-Based Systems Engineering Tool-Chain," *IEEE Systems Journal*, Early Access, pp. 1-12, June. 2019.
- [6] Abeer Abdulaziz Alsanad, Azeddine Chikh, Abdulrahman Mirza, "A domain ontology for software requirements change management in global software development environment," *IEEE ACCESS*, vol 7, pp. 4935249361, Apr. 2019.
- [7] Judson Bandeira, Ig Ibert Bittencourt, Patricia Espinheira, Seiji Isotani, "FOCA: A Methodology for Ontology Evaluation," *eprint arXiv:1612.03353*, 2016. 12.
- [8] Sunju O., Heon Y Y., Joongho A., "Cohesion and coupling metrics for ontology modules," *Information Technology Management*, vol. 12, no. 1, pp. 81-96, 2011.
- [9] Li Zong-yong, WANG Zhi-xue, YANG Ying-ying, WU Yue, LIU Ying, "Towards a Multiple Ontology Framework for Requirements Elicitation and Reuse," *31st Annual International Computer Software and Applications Conference (COMPSAC 2007). IEEE*, 2007.
- [10] Z.Jin, "Ontology-based Requirements Elicitation," *Chinese Journal of Computers*, 23(5): pp. 486-492, 2000.
- [11] Haruhiko Kaiya, Motoshi Saeki, "Using Domain Ontology as Domain Knowledge for Requirements Elicitation," in *Proceedings of 14th IEEE International Requirements Engineering Conference*, pp. 186-195, 2006.
- [12] Farfeleder, S. et al., "Ontology-Driven Guidance for Requirements Elicitation," *ESWC 2011: The Semantic Web: Research and Applications*, pp. 212 - 226, 2011.
- [13] 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*, pp. 364-371, 2006.
- [14] A. Gómez-Pérez, M. Fernandez-Lopez, and O. Corcho, "Ontological Engineering with Examples from the Areas of Knowledge Management," *E-commerce and the Semantic Web*. London, U.K.: Springer, 2004.
- [15] Sarika Jain, Valerie Meyer, "Evaluation and refinement of emergency situation ontology," *International Journal of Information and Education technology*, vol. 8, no. 10, pp. 713-719, 2018.
- [16] D. Vrandecic, "Ontology Evaluation," *Ph.D. dissertation, University of Karlsruhe*, 2010.
- [17] J. Pak and L. Zhou, "A framework for ontology evaluation," in *Workshop on E-Business: Springer Berlin/Heidelberg*, 2009, pp. 10-18.
- [18] Ravi Lourdasamy, Antony John, "A review on metrics for ontology evaluation," in *Proc. ICISC 2018*, Coimbatore, India, 2018, pp. 1415-1421.
- [19] M. Poveda-Villalón, M. C. Suárez-Figueroa, and A. Gómez-Pérez, "Validating ontologies with OOPS!," in *Proc. 18th international conference, EKAW 2012*, Galway City, Ireland, 2012, pp. 267-281.
- [20] N. Casellas, "Ontology evaluation through usability measures," in *Proc. OTM '09*, Vilamoura, Portugal, 2009, pp. 594-603.
- [21] A. Gangemi, C. Catenacci, M. Ciarmita, and J. Lehmann, "Modelling ontology evaluation and validation," in *Proc. ESWC 2006*, Budva, Montenegro, 2006, pp.140-154.
- [22] J. Hartmann, P. Spyns, A. Giboin, D. Maynard, R. Cuel, M. C. Suárez-Figueroa, and Y. Sure, "D1.2.3 Methods for Ontology Evaluation,"

EU-IST Network of Excellence IST-2004-507482 KWEB Deliverable D, vol. 1, 2005.

- [23] S. Tartir, I. B. Arpinar, and A. P. Sheth, "Ontological evaluation and validation," in *Theory and Applications of Ontology: Computer Applications*, Springer, Dordrecht, Netherlands, 2010, pp. 115-130.
- [24] C. Brewster, H. Alani, S. Dasmahapatra, and Y. Wilks, "Data-driven ontology evaluation," in *Proc. 4th International Conference on Language Resources and Evaluation*, Lisbon, Portugal, 2004.
- [25] H. Hlmani and A. D. Stacey, "Contributing evidence to data-driven ontology evaluation: workflow ontologies perspective," in *Proc. 5th International Conference on Knowledge Engineering and Ontology Development*, Vilamoura, Portugal, 2013.
- [26] O. Liubo, Z. Beiji, Q. Miaoxing, and Z. Chengming, "A method of ontology evaluation based on coverage, cohesion and coupling," in *Proc. 2011 8th International Conference on FSKD*, Shanghai, China, 2011, pp. 2451-2455.
- [27] G. Bilgin, I. Dikmen, and M. TalatBirgonul, "Ontology evaluation: An example of delay analysis," *Procedia Engineering*, vol. 85, pp. 61-68, 2014.
- [28] A. Zouaq and R. Nkambou, "Evaluating the generation of domain ontologies in the knowledge puzzle project," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 11, pp. 1559-1572, Nov. 2009.
- [29] Hlmani Hlmani, and Deborah Stacey, "Approaches, methods, metrics, measures, and subjectivity in ontology evaluation: A survey," <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.682.1950&rep=rep1&type=pdf>
- [30] Maziar Amirhosseini, Juhana Salim, "OntoAbsolute as an ontology evaluation methodology in analysis of the structural domains in upper, middle and lower level ontologies," in *Proc. the 2011 International Conference on Semantic Technology and Information Retrieval*, Putrajaya, Malaysia, 2011, pp. 26-33.
- [31] A. Gómez-Pérez, "Ontology evaluation," in *Handbook on Ontologies (International Handbooks on Information Systems)*, S. Staab and R. Studer, Eds. Berlin, Germany: Springer, 2004, pp. 251-273.
- [32] X.Hu, et al., "Construction and Estimation of Requirement Knowledge Ontology for Avionics Electronics System Software," *Computer Engineering*, 39(2): pp. 56-62, 2013.
- [33] 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.
- [34] S. Lovrenčić and M. Čubrilo, "Ontology evaluation-comprising verification and validation," in *Proc. the 19th Central European Conference on Information & Intelligent Systems (CECIIS)*, Varaždin, Hrvatska, 2008, pp. 657-663.
- [35] Newman M E J, "Detecting community structure in networks," *The European Physical Journal B-Condensed Matter and Complex Systems*, 2004, 38(2): 321-330.
- [36] Page-Jones M., "Practical Guide to Structured Systems Design," New York: Yourdon Press, 1980.
- [37] Christophe Dessimoz, Nives Škunca, "The Gene Ontology Handbook," Humana Press, Springer, 2017.
- [38] Liao Lili, Shen Guohua, Huang Zhiqiu, Kan Shuanglong, "An approach for ontology cohesion metrics based on directed acyclic graph," *Computer Engineering & Science*, vol. 37, no. 7, pp. 1297-1303, 2015.
- [39] Ferrari, S., "Beta regression for modelling rates and proportions," *Journal of Applied Statistics*, vol. 31, no.7, pp. 799-815, 2004.
- [40] Hiroya Itoga and Atsushi Ohnishi, "Security Requirements Elicitation via Weaving Scenarios based on Security Evaluation Criteria," *7th International Conference on Quality Software(QSIC 2007)*, IEEE.



Xuan Hu was born in 1983. She received the Ph.D. degree in aerospace science and technology from Beihang University, China, in 2010. She is currently a senior engineer of CEPREI Laboratory. Her main research interests include ontology modeling, software requirement engineering, and software reliability.



Jie Liu was born in 1963. He received the B.S. degree in computer science from Xidian University, China. He is currently the director of the Information Security Research Center of CEPREI Laboratory. His main research interests include information security, software reliability and security, and software evaluation technology.