Frontiers of Information Technology & Electronic Engineering
www.zju.edu.cn/jzus; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

A quality requirements model and verification approach for system of systems based on description logic*

Qing-long WANG†1, Zhi-xue WANG†‡1, Ting-ting ZHANG[1], Wei-xing ZHU[2]
([1]*College of Command Information Systems, PLA University of Science and Technology, Nanjing 210007, China*)
([2]*Information Management Center, PLA University of Science and Technology, Nanjing 210007, China*)
†E-mail: jsxq901901@163.com; wzxcx801@163.com

Received Sept. 25, 2015; Revision accepted Feb. 27, 2016; Crosschecked Feb. 21, 2017

Abstract: System of systems engineering (SoSE) involves the complex procedure of translating capability needs into the high-level requirements for system of systems (SoS) and evaluating how the SoS quality requirements meet their capability needs. One of the key issues is to model the SoS requirements and automate the verification procedure. To solve the problem of modeling and verification, meta-models are proposed to refine both functional and non-functional characteristics of the SoS requirements. A domain-specific modeling language is defined by extending Unified Modeling Language (UML) class and association with fuzzy constructs to model the vague and uncertain concepts of the SoS quality requirements. The efficiency evaluation function of the cloud model is introduced to evaluate the efficiency of the SoS quality requirements. Then a concise algorithm transforms the fuzzy UML models into the description logic (DL) ontology so that the verification can be automated with a DL reasoner. This method implements modeling and verification of high-level SoS quality requirements. A crisp case is used to facilitate and demonstrate the correctness and feasibility of this method.

Key words: System of systems (SoS); Cloud model; Description logic (DL); Requirements verification
http://dx.doi.org/10.1631/FITEE.1500309 **CLC number:** E917; TP391.9

1 Introduction

Due to the rapid development of the society and industry, the developed information system is the result of many autonomous and heterogeneous constituent systems (CSs), having complex constitution and interactions that generate large-scale complex systems of systems (SoSs) (Holt *et al*., 2015). An SoS is an arrangement or a batch of CSs that results when useful and independent CSs are combined into a complicated and larger system that achieves particular capabilities (Petersen *et al*., 2014). The emerging interdisciplinary area of SoS and system of systems engineering (SoSE) is largely driven by societal needs including public services and critical infrastructures, such as health, transport, energy, security, and military (Eusgeld *et al.*, 2011). The variety of complex relationships between CSs facilitates SoS to adapt to different operational environments and different goals. These complex relationships are characterized by multiple connections between CSs, feedback and feed-forward paths, and intricate, hierarchical, and branching topologies. The connections represent intricate networks depending on the characteristics of their linkages. It is clearly impossible to adequately analyze or understand the behavior of a given CS in isolation from SoS or other CSs. Rather, one must consider multiple interconnected CSs and their interdependences (Gao *et al*., 2014). The scale, complexities, and challenges presented by SoS require us go beyond traditional requirements engineering (RE) approaches. However, as is evident from publications

‡ Corresponding author
* Project supported by the National Natural Science Foundation of China (No. 61273210)
ORCID: Zhi-xue WANG, http://orcid.org/0000-0003-2009-6508

derived from major requirements engineering conferences and journals, no significant efforts have been implemented toward addressing specific RE issues for SoSE.

SoSE, which is quite dissimilar to the traditional system engineering that concentrates on the construction of the agreeable systems, tends to concentrate on picking the correct combination of CSs and their interactions to meet a number of constantly varying requirements. The operational capabilities, beyond the limit of the capability that CSs can offer independently, are created by an engineered SoS. The SoS requirements capture properties at the overall SoS-level and CS-level requirements which are specified for particular CSs.

An SoS brings together a set of CSs for a task that none of the systems can accomplish on its own. Each CS maintains its own management, goals, and resources while being coordinated within SoS and adapted to meet SoS goals. SoSE involves a complex procedure of translating capability needs into high-level requirements of SoS, and evaluating how the SoS quality requirements meet the capability needs. One of the key issues is to model the SoS requirements and automate the verification process.

Non-functional requirements (NFRs) play a significant role in RE, especially for the stakeholders to optimize a software solution. Quality requirements are an important class of NFRs (Fotrousi *et al*., 2014) and are of major importance in the development of systems for software-intensive products (Regnell *et al*., 2008). They involve software system attributes such as functional suitability, performance, reliability, usability, security, and portability which are important in achieving stakeholder goals. In RE research and practice, functional requirements (FRs) can be modeled easily with UML or SysML paradigms, while there are few methods available for modeling both FRs and NFRs.

To determine whether the system model satisfies the quality requirements, requirement engineers may have to translate the conceptual model in UML or SysML into an executable model, such as Petri Net or xUML. To facilitate or support verification and validation (V&V) tasks, it is essential to integrate the modeling languages or tools with the interrelated concepts and mechanisms. However, it seems infeasible, because high-level SoS requirements analysis cannot, in the preliminary phase of a large-scale program, look forward into the system design. In addition, the SoS requirements are usually stated with fuzzy concepts, for example, "we need a radar that detects a wider range of targets and provides pictures with higher precision", which makes them even harder to model and verify.

This paper proposes a domain-specific modeling language for both FRs and NFRs modeling, verifies the quality requirements, and strives for addressing the challenges listed above. We take the C4ISR (command, control, communication, computer, intelligence, surveillance, and reconnaissance) system as a typical acknowledged SoS, and present a formal method that models and verifies SoS quality requirements based on fuzzy Unified Modeling Language (UML) and fuzzy description logic (f-DL). Our theory provides a three-layer analysis framework of capability requirements to acquire the domain knowledge for V&V tasks, extends the UML class and association with fuzzy constructs to specify both functional and non-functional characteristics of SoS requirements, introduces f-DL to formalize the UML model, and offers an algorithm to convert the fuzzy UML (f-UML) models into the f-DL ontology so that the verification can be automated with a popular description logic (DL) reasoner such as Pellet (Guizzardi, 2005).

2 Related work

2.1 System of systems and requirements engineering approach

Unlike very large but monolithic complex systems, an SoS has the following distinctions: (1) independent operation of their CSs, i.e., CSs have the ability to operate independently; (2) independent management of their CSs, i.e., their CSs can be independently added in or removed from SoS; (3) evolutionary development, i.e., purposes and functions of their CSs can be modified, removed, or added as required over time; (4) emergent behavior, i.e., the behaviors exhibited in SoS are not associated with any CS; and (5) geographic distribution, i.e., the CSs of the SoS are usually distributed to a large geographical area.

SoS has many different application types, e.g., military (Ender *et al*., 2010), health (Grigoroudis and Phillis, 2013), and critical infrastructures (Eusgeld *et al*., 2011). Identifying the type of SoS at the primary conception stage is important as it influences the entire analysis approach. The SoS engineering guide (ODUSD (A&T), 2008) defines four types of SoS: directed, acknowledged, collaborative, and virtual. The key features differ with respect to the purpose, management, maintenance/change mechanism, example, and RE approach, as summarized in Table 1. The C4ISR system we focus on falls into the acknowledged type because of its recognized objectives and a designated manager (Ncube *et al*., 2013; Petersen *et al*., 2014).

2.2 Capability-based development approach for system of systems

Owing to the increasing complexity and urgency of SoS, many new methodologies are under way to ensure advanced capability-based SoS development for the purpose of offering desired capabilities for various types of possible scenarios and contexts. Obviously, some arrangements of CSs are more robust, perform better, and cost less given the portfolio of capabilities they deliver. The goal is to find the best arrangements given the portfolio of desired capabilities (Bagdatli, and Mavris, 2012). Moreover, capability is actually a kind of requirements specification for sustainably claiming the desired effect under interrelated conditions and performance requirements through integrations of hardware and software solutions to achieve some crucial goals.

The SoS quality requirements development provides a means for attaching an SoS to the desired capabilities (Ge *et al*., 2013; 2014a). SoS architecting is probably the best approach to determine whether or not a to-be SoS could possess the desired capabilities. It should also include the activity of verification and validation to demonstrate which capabilities are principally originated from the composition of CSs with their functionalities satisfying the users' needs.

A number of SoS architecting approaches are investigated. Ge *et al*. (2014a) proposed a graph model for conflict resolution (GMCR) methodology based on the graph model for conflict resolution. The approach makes significant use of the GMCR paradigm flexible design and the inherent realistic concept to solve the problem of CSs decision analysis that occurs in constructing a SoS with expected capabilities. In particular, the approach builds the systematic model at the SoS level for analysis and decision processing to allow for possible compromises among different stakeholders. Moynihan *et al*. (2009) conducted research into the portfolio analysis (PALMA) machine, which is a portfolio optimization tool based

Table 1 System of systems characterizations

Item	SoS type			
	Directed	Acknowledged	Collaborative	Virtual
Purpose	Specific purposes	Recognized objectives	No clear objective	Centrally agreed upon purpose is lacking
Management	Central authority	A designated manager	Voluntary collaboration	No central management authority
Maintenance/ Change mechanism	CSs maintain an ability to operate independently. However, each CS is subordinate to the central authority	Changes in the systems are based on the collaboration between SoS and CSs	Enforcement and maintenance of standards	CSs are relatively invisible to one another
Example	The U.S. Army future combat system	Ballistic missile defense system	Global financial system	Global information grid
RE approach	Classical RE methods; The SoS central authority clearly defines each CS, and also coordinates and controls the SoS RE allocation and evolution	Each CS performs RE procedure independently, and the SoS central authority also performs RE Occasional cooperation of RE artifacts occurs	Each CS performs the RE procedure; The SoS central authority is restricted to describe its global goals; The collaboration of high levels of RE artifacts occurs	No SoS central authority Each CS performs the RE procedure in an irregular and informal way

SoS: system of systems; CSs: constituent systems; RE: requirements engineering

on MITRE. They proposed a novel investment modeling and analysis systems portfolio at any given budget level. More specifically, the PALMA tool can be used to construct a mission tree to hierarchically decompose high-level SoS mission goals into their lowest functions. Huynh *et al.* (2011) defined the architecting problem as an assignment. To solve the architecting problems, they made use of the orthogonal array experiment to optimize the most cost-effective architecture.

In other words, the above capability-based approaches enable an interactive decision-support process for SoS architecting by providing a well-informed optimization means within the capability fields. However, since the SoS development requires more rigorous analysis before CSs are developed and implemented, formalization of requirements description and verification is necessary. We propose a formal analysis approach based on DL for capability-based SoS modeling and analysis to address the modeling and verification problems of function and quality features for SoS requirements.

2.3 Verification and validation approaches and techniques for system of systems

There are various V&V approaches and techniques. Chapurlat *et al*. (2006) classified them into three different categories: non-formal, semi-formal, and formal. The non-formal V&V techniques usually based on human expertise are rarely automated without a high level of formalization. The semi-formal techniques are generally based on the executable model. This technique is a popular approach in a set of projects (Ahmed and Robinson, 2007). However, they cannot keep out incorrect results due to the omission of the scenarios (Chapurlat and Braesch, 2008). The formal techniques are strongly based on formal methods and require a very high level formalization of the modeling language. In particular, the language provides a sufficient mathematical semantic to find errors, mistakes, or inconsistencies through reasons associated with the requirements models.

Although Unified Modeling Language (UML) and Systems Modeling Language (SysML) show the most development for object-oriented SoS architectural modeling, especially in software engineering and system engineering, they and their variants, such as executable UML (xUML), do not have the formal execution semantics to support the formal V&V approaches (Khan, 2010; Haimes, 2012). Therefore, an optional approach is to translate the UML or SysML model into synthesizing executable models in the form of various executable formalisms specified by other modeling languages with a sufficient mathematical semantic. For example, the models are usually translated into variant Petri net (PN) executable models. Ge *et al*. (2013; 2014b) proposed a novel executable modeling approach for SoS architecture analysis and evaluation, by making full use of the Department of Defense Architecture Framework (DoDAF) meta-model (DM2) as an architectural data meta-model and eXtensible Markup Language (XML) technology, and facilitate the automated translation from the architectural model into colored Petri net (CPN) executable models by using the mapping rules between both meta-models. Wang and Dagli (2011) proposed an executable system architecting paradigm for discrete-event system modeling and analysis based on model-driven architecture (MDA), developing a new translation procedure for transforming SysML models into CPN models with a conversion scheme, and proposing a correlative simulation analysis and evaluation method to refine the architecture design, and finally to check the system models for functionality and behavior.

Overall, the above executable modeling methods depend on the model transformations in the principle of MDA. These executable models are based on the CPN using the most popular executable formalism and are well appropriated for information systems which are constructed of many communication and synchronous processes. Unfortunately, owing to the complication of SoS architecture, it causes many problems in the process of modeling and translation. One of these is that it is hard to keep semantic equivalence in the procedure of model translation. Moreover, it is unable to avoid state explosion during CPN-based model execution.

3 Quality requirements modeling for the domain of C4ISR systems

To provide a well-formed approach, we narrowed our research on the domain of the C4ISR

system and focused on the efficiency requirement which is the most important quality requirement, where meeting the right level of efficiency is vital for achieving the mission goals. Insufficient efficiency leads to disappointment and consequent churning, such that stakeholders may decide to abandon the solution and seek an alternative instead. The approach is based on a three-layer modeling framework that helps refine informally expressed FRs and NFRs to unambiguous and measurable descriptions. The methodology is evaluated through the case study in Section 6.

3.1 Three-layer modeling framework

Dong *et al*. (2012) posed a three-layer requirements analysis framework, called the capability requirements analysis framework (CRAF), to address the problem of domain knowledge acquisition by taking full advantage of the framework. In particular, the approach provides semantics to build systematic requirements models for the description of both domain user needs and architectural constraints.

We propose an SoS requirements analysis framework for the C4ISR system based on the framework for classifying the systematic requirements models into three layers, namely, meta, domain, and application.

The model at the meta layer based on the DoDAF capability meta model (CMM), corresponds to the UML model at the M2 layer, to facilitate the model to SoS requirements by taking full advantage of additional and refined concepts, such as ‘mission goal’, ‘capability’, and ‘efficiency’ (Fig. 1).

The model at the domain layer is corresponding to the UML model at the M1 layer. The concepts at the domain layer, instantiated by the meta concepts, describe and constitute the domain knowledge to facilitate and verify the quality requirements of SoS.

The bottom layer, namely the application layer, is corresponding to the M0 layer of UML. The system requirements models consist of application concepts and relationships that are instantiated from the domain concepts and relationships to describe user needs. The quality problem is described at this layer and solved by model verification.

3.2 UML extension with fuzzy constructs

The quality requirements are usually stated in terms of operational effectiveness of the capabilities or systems. The statements may be full of uncertain and vague concepts which cannot be modeled with an ordinary UML tool. For example, early detection (Fig. 2) is the requirement that an air target should be detected as early as the mission requires. The concept ‘early’ is by nature a qualitative concept that varies from mission to mission. It is difficult to set a certain bound by quantitative specification.

Ma *et al*. (2011; 2012) proposed a fuzzy extension of UML based on fuzzy mathematics (fuzzy UML), which poses the fuzzy class, to address the databases problem of fuzzy concepts modeling and analysis for complex objects. The fuzzy class in fuzzy UML defines a set of fuzzy objects retaining similar structures and behavior with three classes of fuzziness. Namely, (1) if the objects, instantiated in a class, is vague, the class is obviously fuzzy and specified with

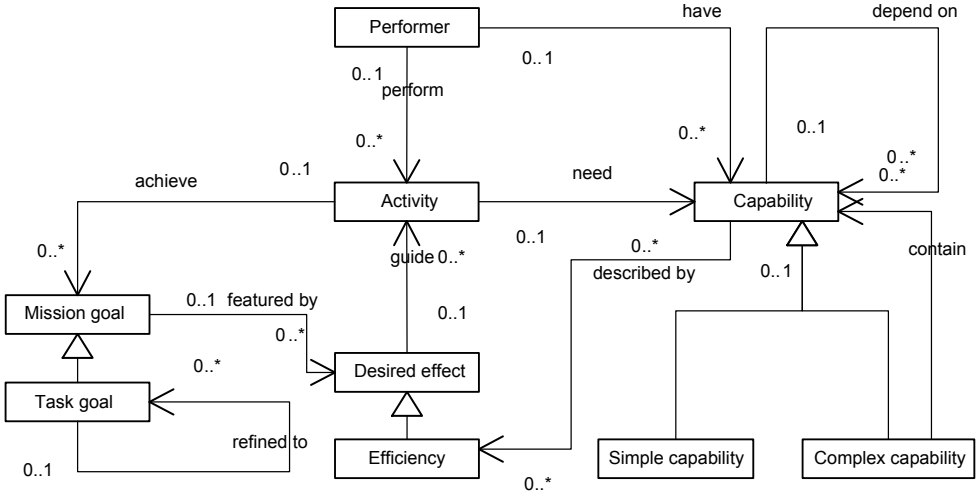

Fig. 1 Meta-model for system of systems requirements of C4ISR systems

an extra attribute (μ) of the domain [0, 1] to describe the membership degree of the instantiated objects; (2) if the attribute of a class is fuzzy, the class directly becomes fuzzy with a keyword FUZZY in its attribute; (3) if a class is fuzzy, its subclass and superclass are also fuzzy classes.

We introduce the first class of fuzziness to address the problem of modeling and analysis encountered in evaluating the quality requirements. The fuzzy concept is modeled as a fuzzy class that is displayed as a dashed box in class diagram and specified with the membership attribute. To evaluate the membership degrees μ of the instantiated objects, we propose an efficiency evaluation function (EEF), which will be discussed in Section 4.

3.3 UML improvement with domain concepts

UML, a general-purpose modeling language, is short on domain applicability and requires domain ontology to specify the domain modeling semantics. Therefore, the meta-model for SoS requirements (Fig. 1) provides DoDAF-compliant ontology to improve the domain applicability of UML. The extended modeling constructs can be used to facilitate both certain and uncertain modeling characteristics of SoS requirements. The fuzzy class of meta object facility (MOF) is extended by the class stereotype such as 'efficiency', and the association is extended by the association stereotype such as 'described by'.

The imported UML extension profile helps requirement engineers model the SoS requirements for the C4ISR system using the domain-specific modeling language. Fig. 2 provides a partial model of the city air defense application domain to illustrate several core concepts related to 'air warning' and 'target interception' tasks. The requirements descriptions are stated in Section 5.

The capability 'detection' is required by the city-air-defense mission for the desired effect that the target should be detected as early as possible. The concept of 'early detect', a vague and uncertain concept, is conveniently modeled as a fuzzy class to describe its fuzziness. Then, the other two concepts 'effectively intercept' and 'accurately identify' can be modeled by the same approach. Each of the fuzzy classes, which has the stereotype 'efficiency' that comes from the meta-model of the SoS requirements for C4ISR systems, encapsulates a specific concept of the quality requirements.

The quality requirements analysis focuses on evaluating the fuzzy membership attribute μ. For example, the requirement engineer determines whether the requirement of 'early detect' is satisfied by evaluating the attribute μ that describes the degree

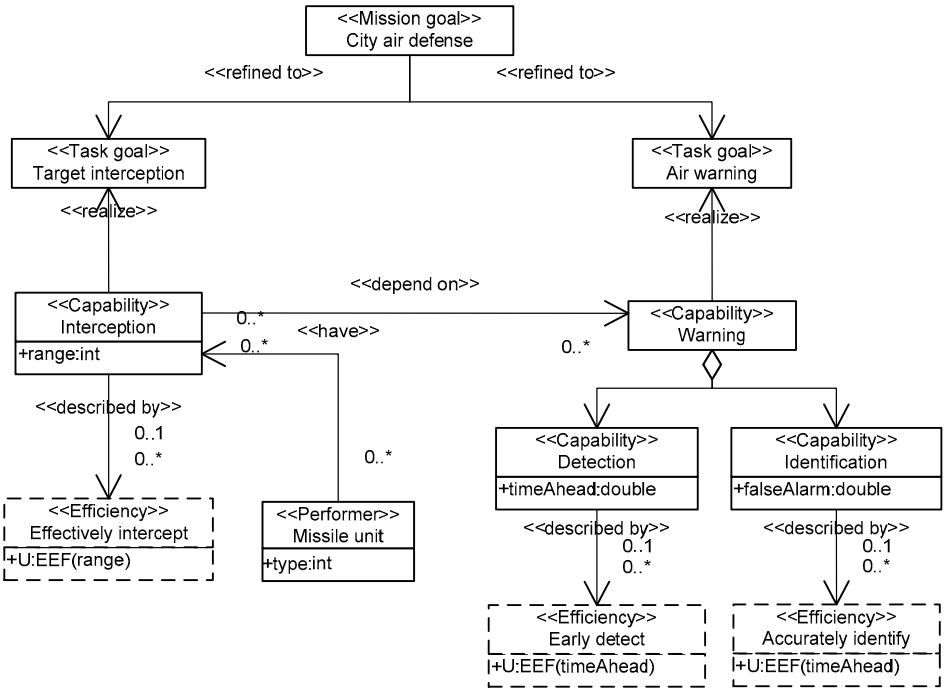

Fig. 2 Partial model of the city air defense application domain

to which the target is early detected. The specific value of the attribute μ, calculated by constructing the EEF, reflects the objects membership degree in different conditions. Therefore, EEF is a very important part of the whole efficiency evaluation process that reflects the satisfied degree of quality requirements. The whole evaluation method will be discussed in the next section.

4 Efficiency evaluation function structure for describing quality requirements

Ontologically speaking, the quality requirement is defined as a basic perceivable or measurable characteristic that inheres in and existentially depends on its subject (Regnell *et al*., 2008; Fotrousi *et al*., 2014). The subject can be a mission goal, a capability, etc. In this theory, efficiency is an important quality requirement that actually causes constraints over FRs. It is a key role in the specification of an NFR and is crucial to selecting an appropriate alternative solution for stakeholders. We propose an EEF based on the cloud model to address the problem of evaluating the efficiency countered in analyzing the quality requirements of SoS, by taking full advantage of the intrinsic flexible and adaptive characteristics of EEF.

4.1 Efficiency evaluation function based on a cloud model

Definition 1 (Efficiency evaluation function) EFF is usually used to calculate the possibility or probability of achieving the desired effect of a mission or a task goal with several quantitative attributes of the fuzzy class, and is encountered in a capability application to an activity with the domain range of 0 to 1.

Theoretically, the variety of different domains makes it difficult to construct a common EEF. To construct an EEF correctly and conveniently, the following factors should be taken into account during the thought process: (1) the variations in missions or tasks, because the performance may vary greatly when encountered in the capability application model for different tasks or missions; (2) the variations in operational environments, because the operational environment plays an important role in the efficiency evaluation. For instance, the speed of a vehicle can be over 100 km/h in the operational environment of an expressway; however, the speed may not be more than 50 km/h in the operational environment of a desert. Moreover, EEF should be constructed through satisfying the following constraints:

Accuracy: EEF may be constructed with the sample data given by domain experts according to their prior knowledge. In this case, the data must be relevant to the domain, and the efficiency values produced must not contradict with the domain knowledge.

Flexibility: Engineers should provide a family of EEF for a capability so that they can easily modify the EEF to fit the new mission in case the mission is changed.

Easiness to use: EEF must be constructed easy to use by designing an algorithm to estimate the efficiency with any valid value of quantitative attributes.

We believe the cloud model, defined as follows, is applicable to address the problem of efficiency evaluation for C4ISR capability requirements.

Definition 2 (Cloud model) Let U be a universal set of discourse described by precise numbers, and C be a set of qualitative concepts quantified by U. If a concept $c\in C$ is randomly valued as a number $x\in U$ with a certainty degree of x, there must be a random number with a stable tendency $\mu(x)$. If μ: $U\rightarrow[0, 1]$, $\forall x\in U$, and $x\rightarrow\mu(x)$, the distribution of x on U is defined as the cloud, where every x becomes a cloud drop.

The three numerical characteristics, namely, expected value (Ex), entropy (En), and hyper-entropy (He), are used to describe a cloud model. Ex, the representative value for the concept described by the cloud, is used to represent the mathematical expectation of the cloud drop with the distribution of a universal set. En, specified by fuzziness and randomness of the concept, is used to measure the uncertainty of the qualitative concept and reflect the acceptable value distribution. He, specified by both fuzziness and randomness of the entropy, is used to measure the uncertainty of the entropy, namely, the entropy of entropy. If He grows, the cloud drops will become more random and discrete, and the figure of the cloud will become visually thicker.

Definition 3 (Normal cloud model) Let U be a universal set of discourse described by precise numbers, and C(Ex, En, He) be the qualitative concepts quantified by U. If a concept c is randomly valued as a number $x\in U$, x is subject to the Gaussian distribution

$x \sim N(\mathrm{Ex}, (\mathrm{En}')^2)$, $(\mathrm{En}')^2$ is a random value subject to the Gaussian distribution $\mathrm{En}' \sim N(\mathrm{En}, \mathrm{He}^2)$, and the certainty degree of x for c is subject to $\mu(x) = \exp\left[-\frac{(x-\mathrm{Ex})^2}{2(\mathrm{En}')^2}\right]$, then the distribution of x on U is defined as a normal cloud.

The normal cloud model is a symmetrical distribution and fundamental model for the cloud model. The normal membership function and the normal distribution function are widely used in social science and natural science. The two functions belong to different theories, probability theory and fuzzy theory, and both have various limitations in expressing uncertain concepts. The normal cloud model expresses the uncertainty of concepts by mapping qualitative concepts into numerical values with the help of a specific structure generator constructed using expectations, entropy, and hyper entropy. The specific structure may not only loosen the precondition of the normal distribution, but can also be allowed by using the expectation functions of the normal membership distribution, instead of accurately defining the membership functions. Therefore, the specific structure is generally more applicable for conversions between quantitative values and qualitative concepts. For the domain analysis of the C4ISR system, it is more convenient to use a normal cloud model to quantitatively express the fuzzy and qualitative information of the C4ISR requirements, and thereby construct an EEF to evaluate the effectiveness of SoS.

4.2 Estimation method

4.2.1 Efficiency evaluation function construction

This section discusses how to use the cloud model theory and experimental data to build an EEF step by step, and then shows how the built EEF is flexible and adaptive to mission change. The steps of EEF construction are as follows (Fig. 3):

1. Begin the construction process by collecting efficiency data, either from experiments through testing and simulation on C4ISR systems or by expert experience, and classifying them into a number of data groups for future calculations.

2. Use the collected efficiency data to build individual EEF cloud models by taking advantages of the backward cloud generator of EEF.

3. Synthesize the individual cloud models into one.

4. Compose the expectation function of the synthetic cloud model, i.e., EEF, and apply it to the efficiency evaluation.

While composing EEF, domain experts should care about the factors that may affect EEF when the C4ISR systems are performed for various missions and tasks, and take full advantage of the backward cloud generator and their expertise to choose an appropriate elementary format for the cloud model that matches the sample points of the given data and provides a correlation with actual situations.

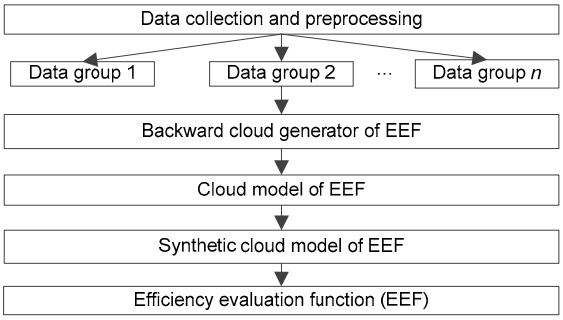

Fig. 3 Construction of the efficiency evaluation function based on a normal cloud model

4.2.2 Design of the backward cloud generator for efficiency evaluation function

m groups of data are collected from the evaluated systems. Each group contains n pieces of data and each piece is assumed to be a cloud drop. m units of the cloud model for EEF can be generated by the backward cloud generator.

When constructing EEF, there are several individual cloud models that need to be combined into a synthetic cloud model. As the individual cloud models are differentiated by importance, their importance weights must be considered. Therefore, a concept of weighted percentage is introduced.

The weighted percentage of a cloud model is the number of data points of the cloud model divided by the total number of data points formulated as

$$\omega = \lambda_i / N, \tag{1}$$

where N is the total number of data points and λ_i is the number of date points for each property owned by the cloud model.

Algorithm 1 describes the backward cloud generator for EEF.

Algorithm 1 Backward cloud generator for efficiency evaluation function

Input: Input date sample points $\boldsymbol{X}_i(x_{i_1}, x_{i_2}, \ldots, x_{i_m})$ (i=1, 2, …, n).

Output: m units of individual cloud model (EC_1, EC_2, …, EC_m) and their numerical characteristics (Ex_1, Ex_2, …, Ex_m, En_1, En_2, …, En_m, He_1, He_2, …, He_m).

1 For $\boldsymbol{X}_i(x_{i_1}, x_{i_2}, \ldots, x_{i_m})$ (i=1, 2, …, n), calculate the means of the sample data:

$$\overline{X_i} = \frac{1}{M_i}\sum_{j=1}^{M_i} x_{ij},\ i = 1, 2, \ldots, m. \tag{2}$$

2 Set the expectation value of each cloud model as

$$(\mathrm{Ex}_1, \mathrm{Ex}_2, \ldots, \mathrm{Ex}_m) = (\overline{X_1}, \overline{X_2}, \ldots, \overline{X_m}).$$

3 Calculate the entropies of the cloud models:

$$\mathrm{En}_i = \sqrt{\frac{1}{M_i - 1}\sum_{j=1}^{M_i}(x_{ij} - \mathrm{Ex}_i)^2},\ i = 1, 2, \ldots, m. \tag{3}$$

4 For each cloud drop (x_{ij}, μ_i), calculate the entropies:

$$\mathrm{En}'^{2}_{ij} = \sqrt{\frac{-(x_{ij} - \mathrm{Ex})^2}{2\ln \mu_i}},\ i = 1, 2, \ldots, m. \tag{4}$$

5 Calculate the standard deviation of each En_i' and finally determine the hyper-entropy He_i:

$$\mathrm{He}_i = \sqrt{\frac{1}{M_i - 1}\sum_{j=1}^{M_i}(\mathrm{En}'_{ij} - \overline{\mathrm{En}'_i})^2},\ i = 1, 2, \ldots, m. \tag{5}$$

Algorithm 1 is used to conveniently generate the numerical characteristics EC (Ex, En, He) for the individual cloud models.

4.2.3 Constitution of the synthetic efficiency evaluation function

To evaluate the efficiency of the C4ISR system, we combine the individual cloud models into a synthetic one by taking their efficiency weights into consideration. The new model can be built in two steps:

First, synthesize the numerical characteristics SEC(Ex, En, He) by using the following formulae:

$$\begin{cases} \mathrm{Ex} = \sum_{i=1}^{m} \mathrm{Ex}_i \cdot \omega_i, \\ \mathrm{En} = \sqrt{\sum_{i=1}^{m} \mathrm{En}_i^2 \cdot \omega_i}, \\ \mathrm{He} = \sum_{i=1}^{m} \mathrm{He}_i \cdot \omega_i, \end{cases} \tag{6}$$

where m is the number of individual cloud models and $\omega_i \left(\sum_{i=1}^{m} \omega_i = 1 \right)$ is the corresponding weight for each cloud model.

Then, based on the synthesized numerical characteristics, build the expectation function of the synthetic cloud model, or the EEF. The formula is as follows:

$$\mu(x) = \exp\left[-\frac{(x - \mathrm{Ex})^2}{2\mathrm{En}^2} \right]. \tag{7}$$

In the efficiency evaluation, the expectation curve may provide a quantitative description of the fuzzy and qualitative C4ISR requirements. Each point on the curve represents an evaluation of the system operation efficiency for a specific circumstance. Therefore, the equation of the curve is actually the EEF. The advantage of a cloud model is that the certainty degree of the cloud drops is irrelevant to the numerical characteristics of the concept and the contents of the expression. This agrees with the human evaluation behavior that a different expert may have a different comprehension on a same efficiency concept, while the collective comprehension tends to be consistent. This may help eliminate possible biases and raise the confidence level of the evaluation results.

4.3 Analysis on the calculation results for efficiency evaluation

The concept ‘effectively intercept’ demonstrated in Fig. 2 represents the efficiency of the capability concept of ‘interception’ possessed by the missile unit. In particular, the missile unit provides two types of missile units to assign to different air defense tasks, namely, short-range and intermediate-range. Owing to the length of the paper, it is better to take the short-range air defense missile as an example to illustrate the development procedure for the EEF of ‘effectively intercept’ (EEFEI).

In the development of EEFEI, the domain experts provide four sets of sample points and use the backward cloud generator to build four cloud models: EC_1(10.0, 6.0, 0.5), EC_2(12.0, 7.0, 0.5), EC_3(14.0, 8.0, 0.5), and EC_4 (16.0, 9.0, 0.5) (Fig. 4).

The four cloud models are then combined into one synthetic cloud model, and EEFEI is finally built

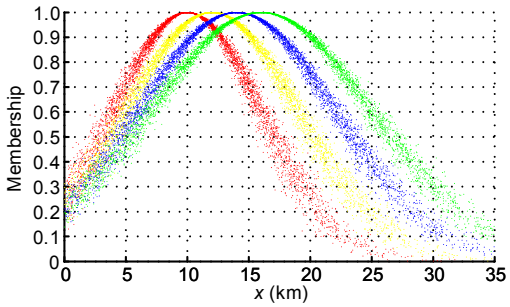

Fig. 4 Cloud models for efficiency evaluation functions of 'effectively intercept'

by calculating the expectations of the performance curve.

Applying the formula of the synthetic cloud model, the domain experts synthesize the numerical characteristics as EC_S(Ex=13.4, En=7.8, He=0.5), where ω_1=0.2, ω_2=0.2, ω_3=0.3, and ω_4=0.3, and thereby build the EEFEI for short-range missile as follows:

$$\mu(x)=\exp\left[-\frac{(x-13.4)^2}{2\times 7.8^2}\right]. \tag{8}$$

From the synthetic cloud (Fig. 5a), we can estimate that the best effective striking point for the short-range missile is 14 km, where the efficiency of the interception capability reaches the maximum.

As pointed out earlier, EEF may be sensitive to the change of the mission or task. Provided that the existing task for short-range air defense is replaced by an intermediate protection task, experts should provide a set of sample data for the new task points based on experiments, and use the backward cloud generator to generate new models, such as: EC_1(21.0, 11.0, 0.5), EC_2(25.0, 13.0, 0.5), EC_3(29.0, 15.0, 0.5), and EC_4(33.0, 17.0, 0.5). The new synthetic cloud model is built as EC_M(Ex=26.2, En=13.8, He=0.5), where ω_1=0.3, ω_2=0.3, ω_3=0.2, and ω_4=0.2. Finally, the EEFEI for the intermediate-range missile is built as follows:

$$\mu(x)=\exp\left[-\frac{(x-26.2)^2}{2\times 13.8^2}\right]. \tag{9}$$

Fig. 6 provides a comparison of two efficiency curves for different tasks.

From Fig. 6, we can see that the two curves join at (18, 0.83), which may lead to an assumption that

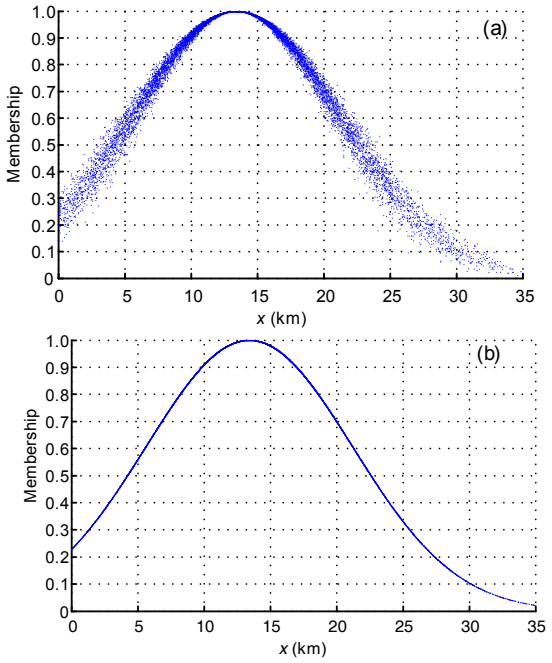

Fig. 5 Synthetic cloud model (a) and efficiency evaluation function of 'effectively intercept' (b) for a short-range air defense missile

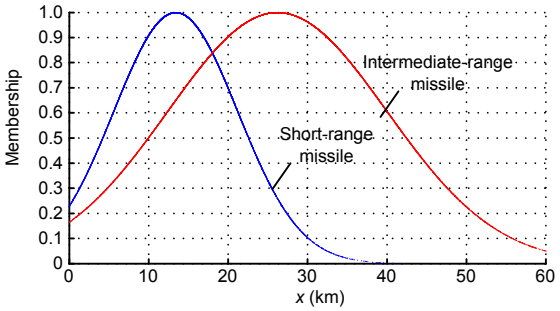

Fig. 6 Comparison of efficiency evaluation functions of 'effectively intercept' for different tasks

0.83 can be used as the valve value to distinguish the two tasks, and hence to choose the appropriate EEFEI. This assumption is reasonable, because it is in full accord with our domain knowledge that when the flying target is within 18 km, the short-range air defense missile is usually applied; otherwise, the intermediate-range air defense missile will be applied.

5 Quality requirement verification based on description logic

5.1 f-SHIN

Description logic (DL) is a logical reconstruction of the frame-based knowledge representation languages, allowing for the simple and

well-established declarative semantics of the Tarski-style. In essence, DL is the theoretical opposite of the Web Ontology Language (OWL), which plays a special role in the representation and reasoning of ontology. To handle fuzzy and vague problems, some researchers have extended f-DL. In our study, we choose f-SHIN, a subsystem of f-DL, to describe the necessary quality requirements, because it has a strong ability of representation and decidability, and takes advantage of some convenient reasoners such as Pellet.

Here is a brief introduction to the background of f-SHIN. N_C, N_R, and N_I are pairwise disjoint sets of atomic fuzzy concepts and atomic fuzzy relations and individuals, respectively.

The definition of the fuzzy concepts in f-SHIN can be inductively summarized as follows:

1. For all, if $A \in N_C$, then A is a fuzzy concept.

2. For all, if $o \in N_I$, then $\{o\}$ is also a fuzzy concept.

3. Let C, D be fuzzy concepts, and let $R \in N_R$, $S \in N_R$ be simple. Then $\neg C$, $C \sqcup D$, $C \sqcap D$, $\forall R.C$, $\exists R.C$, $\geq pS$, and $\leq pS$ are fuzzy concepts, where $p \in \mathbb{R}$.

The fuzzy interpretation of f-SHIN is defined as: $I=<\Delta^I$ and $\bullet^I>$, where Δ^I is a nonempty set of the domain and $\bullet^I$ is a fuzzy interpretation mapping function defined as follows:

1. Each abstract individual o is an element in the domain $(o)^I \in \Delta^I$.

2. Each atomic fuzzy concept has a membership degree function $A^I: \Delta^I \rightarrow (0,1]$.

3. Each atomic fuzzy relationship R has a membership degree function $R^I: \Delta^I \times \Delta^I \rightarrow (0,1]$.

4. $\bullet^I$ maps fuzzy concepts and roles into subsets of Δ^I and $\Delta^I \times \Delta^I$, which are described in more detail in Stoilos *et al*. (2007).

The knowledge base (KB) of f-SHIN is a triple $\sum=(T, H, A)$. The expression form of the fuzzy concept axiom is $C \sqsubseteq D$ or $C \equiv D$ in fuzzy TBox T. For any fuzzy interpretation I, I satisfies $C \sqsubseteq D$ (resp. $C \equiv D$) iff $\forall a \in \Delta^I$, $C^I(a) \leq D^I(a)$ (resp. $C^I(a)=D^I(a)$). The detailed descriptions of H and A are discussed in Stoilos *et al*. (2007).

5.2 Model transformation

To use the DL inference engine to verify the quality requirements, it is prerequisite to transform the f-UML models of the SoS requirements into the f-DL ontology.

It is assumed that the SoS requirements have been modeled with the three-layer modeling framework, and the domain model and application model are constructed. Then the procedure of model transformation consists of two steps: (1) mapping all the classes in the f-UML domain models into the f-DL concept set and mapping all the relationships to the f-DL axiom set, and (2) mapping all objects and their relationships in the f-UML application models into the f-DL assertion set. The detailed description of the transformation is given in Algorithm 2.

Algorithm 2 Model transformation

Input: f-UML domain models; f-UML application models.
Output: Requirements ontology.

1 Create axioms in Tbox T;
2 Create concepts:
  For every class C in the domain model, create a concept C of the same name in Tbox T;
3 Create axioms:
  For every generalization between subclass C and superclass D in the domain model, create an axiom $C \sqsubseteq D$ in Tbox T;
  For every association R between class C and class D in the domain model, create an axiom $C \sqsubseteq \forall R.D$ in Tbox T;
4 Create assertions in Abox A;
5 Create from instances:
  For every object c in the application model which belongs to class C with the membership degree n, create an assertion $< c: C > \triangleright\triangleleft n$ in Abox A;
6 Create from instance links:
  For every link l in the application model which links two objects a and b, create an assertion $< a:b > \triangleright\triangleleft n$ in Abox A, where $\triangleright\triangleleft \in \{=, <, \leq, \geq, >, \geq\}$ and $n \in [0,1]$;

The ontology built by the algorithm may contain numerical numbers and cannot be accepted by an ordinary DL reasoner like Pellet and Racer, because the DL reasoners do not support mathematical calculations on numerical numbers. To solve this problem, we applied the crisping technique suggested by Fernando *et al*. (2009) to transform these fuzzy concepts further into crisped ones.

Here is an example to explain the transformation process. In the domain model, the capability concept ‘intercept’ is described by the efficiency concept ‘effectively intercept’ and instantiated as the object

'ei1: effective intercept' that has a membership value of 0.9 in its application model (Fig. 5). The transformation procedure has the following steps: firstly, an axiom 'Interception $\sqsubseteq \forall$ described by effectively intercept' is created in TBox, and an assertion 'ei1: effectively intercept=0.9' is created in the ABox. Secondly, for the instance 'ei1', the crisped concept 'effectively intercept=0.9' is generated and extends the efficiency concept 'effectively intercept'. A crisp concept 'interception=1' is also generated and extends the distinct concept 'interception'. Finally, the axiom setup in step 1 is replaced by a new axiom, 'interception=1 $\sqsubseteq \forall$ described by effectively intercept=0.9'. and the assertion is replaced by the new assertion 'ei1: effectively intercept=0.9'.

5.3 Quality requirements verification

In the application model, the quality problem arises when the system is unable to meet the efficiency needs of the task. The solution is to discover the quality problem by verifying the requirements models through the following steps: (1) use the transformation algorithm in Section 4.2 to transform the f-UML models into f-DL ontology; (2) use Semantic Web Rule Language (SWRL) to add quality constraints into the ontology; (3) use the DL reasoner to check the consistency and integrity of the ontology.

The quality constraints are primarily used to check whether the SoS requirements models have conflicts. For example, there are two architectural constraints for C4ISR systems:

Constraint 1 (C_1) If capability C_1 depends on capability C_2, efficiency E_1 of capability C_1 must not be greater than efficiency E_2 of capability C_2.

The SWRL formulas are described as

$$\begin{aligned}&\text{capability}_{=1}(C)\wedge\text{capability}_{=1}(C_1)\\&\wedge\text{efficiency}_{=m}(E_1)\wedge\text{efficiency}_{=n}(E_2)\\&\wedge\text{describedby}_{=1}(C_1, E_1)\wedge\text{describedby}_{=1}(C_2, E_2)\\&\wedge\text{dependon}_{=1}(C_1, C_2)\\&\rightarrow\text{swrlb:greaterThan}(n, m).\end{aligned}$$

The 'greaterThan' is herein one of the predicates defined to compare two numbers.

Constraint 2 (C_2) If a capability C comprises a set of sub-capabilities C_i, C_{i+1}, …, C_n, the efficiency E of capability C must not be greater than the minimum efficiency E_i of the sub-capabilities C_i ($1\leq i\leq n$).

The SWRL formulas are described as

$$\begin{aligned}&\text{capability}_{=1}(C)\wedge\text{capability}_{=1}(C_1)\\&\wedge\text{efficiency}_{\geq m}(E)\wedge\text{efficiency}_{\geq n}(E_1)\\&\wedge\text{describedby}_{=1}(C, E)\wedge\text{describedby}_{=1}(C_1, E_1)\\&\wedge\text{aggregation}_{=1}(C, C_1)\\&\rightarrow\text{swrlb:greaterThan}(n, m).\end{aligned}$$

6 Case study

A notional example of C4ISR SoS in the domain of city air defense proposed by Dong *et al*. (2012) and Ge *et al*. (2014b) was used to illustrate the applicability and feasibility of the proposed modeling and verification approach. It is based on the operational concept that the C4ISR SoS in the domain of city air defense is anticipated to provide the capabilities of protecting a bounded key city area against intruders (or threats) (Fig. 7). Two types of capability, air warning and target interception, play a pivotal role. Target interception depends on air warning, especially on its sub-capabilities, and air target detection and identification. Air target detection should be as early as possible to detect the intruders or threats to leave enough time to intercept. The target identification should be as accurate as possible to identify the types of air targets and possible threats. Any detection or identification of low efficiency may cause a failure in the mission of air target interception. Subsequently, an interception missile was launched to intercept and employ the different types of missile units according to the threat level and status of the intruder. There are two types of interception missiles, short-range and intermediate-range, in the application of the task. The intruders or threats must be killed within a specified distance (e.g., 20 km) or time (e.g., 300 s); otherwise, they will become viable threats leakers and will be able to attack the targets in the key city area.

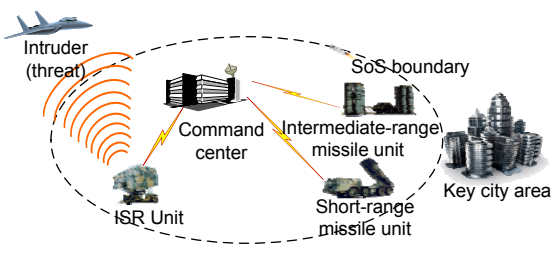

Fig. 7 Operational concept graphic of the C4ISR system of systems in the domain of city air defense

The proposed method in Section 3 facilitates building the application model for the C4ISR SoS requirements in the domain of city air defense (Fig. 8). First, some key efficiency concepts were modeled. For example, the concepts of 'effectively intercept', 'early detect', and 'accurately identify' clarified the desired efficiency of the capability of 'interception', 'detection', and 'identification', respectively. Obviously, the concept of 'effectively intercept' relies on the concept of 'early detect' and 'accurately identify' to realize the expected assumptions. The construction method of EFF has been introduced in Section 4. The efficiency evaluation of EFF is shown in the property box and sends back the specific efficiency value.

The quality problems are: the interception efficiency (evaluated to be 0.9) of the intermediate-range interception missile is larger than the detection efficiency (evaluated to be 0.85). The interception capability depends on the detection capability; thus, it violates constraints C_1 and C_2. The domain interpretation of the conflict is: with the detection capability the threat target is found 7 min ahead, which cannot support effective interception that has a 90% possibility to destroy the air threat target.

For quality verification, the SoS requirements ontology is constructed from the domain model and application model. It takes type 1 as an example of the missile to demonstrate the verification procedure in detail. Table 2 displays the contents of TBox and ABox.

The reasoning is processed in the following way:

A new assertion a16 'intercept1: capability$_{=1}$' can be deduced from assertion a1 and axiom A1.

A new assertion a17 'ei1: efficiency$_{=0.9}$' can be deduced from assertion a2 and axiom A2.

A new assertion a18 'w1: capability$_{=1}$' can be deduced from assertion a4 and axiom A5.

A new assertion a19 'we: efficiency$_{=0.85}$' can be deduced from assertion a5 and axiom A6.

A new assertion a20 'd1: capability$_{=1}$' can be deduced from assertion a5 and axiom A10.

A new assertion a21 'ed1: efficiency$_{=0.85}$' can be deduced from assertion a6 and axiom A12.

A new assertion a22 'swrlb: greaterThan(0.85, 0.85)' can be deduced from assertions a18, a19, a20, a21, a6, a7, a10 and quality constraint R1.

A new assertion a23 'i1: capability$_{=1}$' can be deduced from assertion a12 and axiom A11.

A new assertion a24 'ai1: efficiency$_{=0.9}$' can be deduced from assertion a13 and axiom A14.

A new assertion a25 'swrlb: greaterThan(0.9, 0.85)' can be deduced from assertions a18, a19, a23, a24, a6, a11, a14 and quality constraint R2.

A new assertion a26 'swrlb: greaterThan(0.9, 0.85)' can be deduced from assertions a18, a19, a23, a24, a6, a11, a14 and quality constraint R2.

A new assertion a26 'swrlb: greaterThan(0.85, 0.9)' can be deduced from assertions a16, a17, a18, a19, a3, a6, and a15.

Apparently, a26 is a wrong assertion, and the quality problem is found.

The above case is a simple example to illustrate how the method is applied in a limited context. However, for a practical system, there may be thousands of concepts in its model, which makes it difficult to manually check all the quality problems. It is convenient to use the ontology reasoner engine to obtain an automatic model validation. The different system solutions require different costs, which facilitates the trade-off analysis among stakeholders

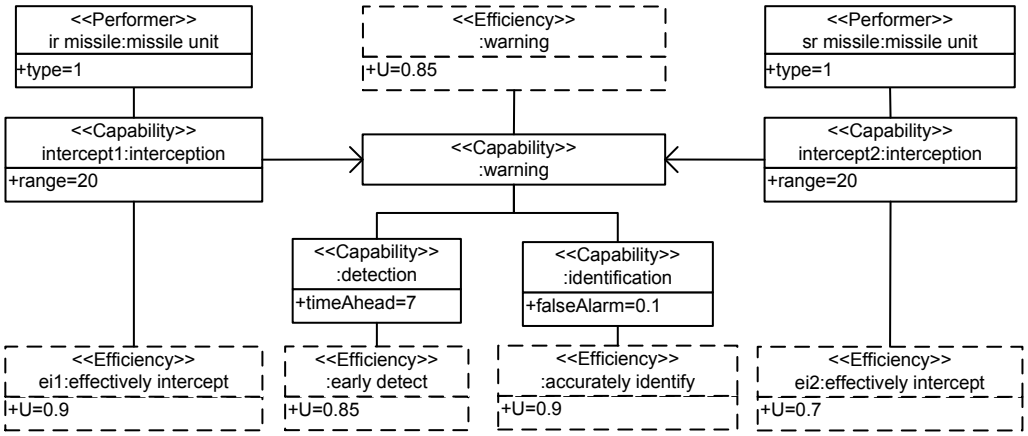

Fig. 8 Application model for the C4ISR system of systems requirements in the domain of the city air defense

to adopt the proper efficiency to balance the quality and cost of the system solution through quality verification.

Obviously, some solutions of systems will be more robust, perform better, and cost less, given the portfolio of capabilities they are able to provide. The goal of SoS requirements engineering is finding the best solution given the set of desired capabilities. Therefore, many researchers have concentrated on the model verification to evaluate the different solutions of systems. The following paragraphs compare our approach and other methods in SoS model verification. The discussion below constitutes the basis of the population of Table 3.

Description ability, heavily influenced by the chosen modeling language, plays a fundamental role in problem modeling. Due to the extension of the UML profile, our approach inherits the strong description ability of high-level abstraction. The methods of Workflow, Temporal Logic, and Petri Nets are unspecialized in the description, since these methods need model transformation between the description model and verification model, and hardly ensure completeness, especially when the description model is overly complicated. The simulation methods (such as discrete event, agent-based) are also strong in description ability. However, each simulation model's implementation depends on the detailed behavioral models, which are almost impossible to obtain in the requirements engineering phase of SoS engineering.

Regarding knowledge reusability, our approach allows enriching domain knowledge by adding a new meta-model to the domain knowledge base. The method of Workflow, Temporal Logic, and Petri Net, however, do not facilitate collection and reuse knowledge. The simulation methods allow for a setup mechanism to help with knowledge reuse.

The verification efficiency is primarily determined by human interference, execution time, and operational cost in the verification process. Our method, which depends on human interference, is graded low. The simulation methods (such as discrete event, agent-based) need long runtime, which is not practical for an all-out SoS in the early phase. The

Table 2 Axioms and assertions in the system of systems requirements ontology

Tbox	Abox
A1: $\text{interception}_{=1} \sqsubseteq \text{capability}_{=1}$	a1: $\text{interception1} : \text{interception}_{=1}$
A2: $\text{effectively intercept}_{=0.9} \sqsubseteq \text{efficiency}_{=0.9}$	a2(ei1): $\text{effectively intercept}_{=0.9}$
A3: $\text{interception}_{=1} \sqsubseteq \forall \text{described by}_{=1}.\ \text{effectively intercept}_{=0.9}$	a3(interception1, ei1): $\text{described by}_{=1}$
A4: $\text{interception}_{=1} \sqsubseteq \forall \text{depend on}_{=1}.\ \text{warning}_{=1}$	a4(w1): $\text{warning}_{=1}$
A5: $\text{warning}_{=1} \sqsubseteq \text{capability}_{=1}$	a5(we): $\text{warning efficiency}_{=0.85}$
A6: $\text{warning efficiency}_{=0.85} \sqsubseteq \text{efficiency}_{=0.85}$	a6(w1,we): $\text{described by}_{=1}$
A7: $\text{warning}_{=1} \sqsubseteq \forall \text{described by}_{=1}.\ \text{warning efficiency}_{=0.85}$	a7(w1,d1): $\text{aggregation}_{=1}$
A8: $\text{warning}_{=1} \sqsubseteq \forall \text{aggregation}_{=1}.\ \text{detection}_{=1}$	a8(d1): $\text{detection}_{=1}$
A9: $\text{warning}_{=1} \sqsubseteq \forall \text{aggregation}_{=1}.\ \text{identification}_{=1}$	a9(ed1): $\text{early detect}_{=0.85}$
A10: $\text{detection}_{=1} \sqsubseteq \text{capability}_{=1}$	a10(d1,ed1): $\text{described by}_{=1}$
A11: $\text{identification}_{=1} \sqsubseteq \text{capability}_{=1}$	a11(w1,i1): $\text{aggregation}_{=1}$
A12: $\text{early detect}_{=0.85} \sqsubseteq \text{efficiency}_{=0.85}$	a12(i1): $\text{identification}_{=1}$
A13: $\text{detection}_{=1} \sqsubseteq \forall \text{described by}_{=1}.\ \text{early detect}_{=0.85}$	a13(ai1): $\text{accurately identify}_{=0.9}$
A14: $\text{accurately identify}_{=0.9} \sqsubseteq \text{efficiency}_{=0.9}$	a14(i1,ai1): $\text{described by}_{=1}$
A15: $\text{identification}_{=1} \sqsubseteq \forall \text{described by}_{=1}.\ \text{accurately identify}_{=0.9}$	a15(intercept1, w1): $\text{depend on}_{=1}$

Table 3 Comparison between several typical verification methods

Metric	Our approach	Workflow	Temporal Logic	Petri Nets	Simulation engine
Description ability	++	+	+	+	++
Knowledge reusability	++	–	–	–	+
Efficiency of verification	++	++	++	+	–
State explosion	–	+	+	++	–

++: strongly support/existent; +: support/existent; –: does not support/non-existent

other three methods, supported by automated tools, facilitate verification with high efficiency.

According to the characteristics of SoS, the system inherits complicated architecture and numerous concurrent constituents whose state space will explode beyond finite computation capability. The problem of state explosion constantly perplexes the various methods based on exhaustive searches, such as Workflow, Temporal Logic, and Petri Net. Fortunately, our method introduces deduction reasoning with the finite space of state space. The simulation method also requires finite space to accomplish the model verification.

Careful analysis demonstrates that our approach facilitates the reuse of knowledge to acquire description ability, and has a modest degree of verification efficiency, compared to the other methods.

7 Conclusions

In this paper, we regarded NFRs as quality requirements and proposed a modeling and verification approach countered in quality requirements analysis to facilitate stakeholders to adopt the appropriate quality to decrease the cost and risk in the developing procedure of SoS. The approach starts with quality requirements modeling and then extends UML with fuzzy constructs to address the problem of modeling the uncertain and vague concepts in quality requirements. We posed an efficiency evaluation methodology based on a cloud model to construct the efficiency evaluation function and made them unambiguous and measurable. To help stakeholders adopt proper efficiency to balance the quality and cost of the system solution, we proposed a verification method that includes a model transformation algorithm to translate the f-UML models into ontology specified in f-DL for automating the verification through taking full advantage of a DL reasoner such as Pellet.

The proposed method is under test. It needs further application and validation in a large-scale requirements engineering situation. Our future research aims at extending our method to architectural design by covering more viewpoints of DoDAF2.0 and addressing a broader scope of NFRs issues. Moreover, we intend to develop a domain-specific rules description language based on the domain ontology and SWRL. The language will help engineers reuse the domain knowledge to accelerate and simplify quality requirements development in projects. It will also enable domain experts to be free from the lack of formal domain knowledge and also allow for defining more performance constraints.

References

Ahmed, R., Robinson, S., 2007. Simulation in business and industry: how simulation context can affect simulation practice? Proc. Spring Simulation Multiconference, p.152-159.

Bagdatli, B., Mavris, D., 2012. Use of high-level architecture discrete event simulation in a system of systems design. IEEE Aerospace Conf., p.1-13. http://dx.doi.org/10.1109/AERO.2012.6187442

Chapurlat, V., Braesch, C., 2008. Verification, validation, qualification and certification of enterprise models: statements and opportunities. *Comput. Ind.*, **59**(7):711-721. http://dx.doi.org/10.1016/j.compind.2007.12.018

Chapurlat, V., Kamsu-Fogum, B., Prunet, F., 2006. A formal verification framework and associated tools for enterprise modeling: application to UEML. *Comput. Ind.*, **57**(2):153-166. http://dx.doi.org/10.1016/j.compind.2005.06.001

Dong, Q.C., Wang, Z.X., Chen, G.Y., *et al*., 2012. Domain-specific modeling and verification for C4ISR capability requirements. *J. Cent. South Univ.*, **19**(5):1334-1341. http://dx.doi.org/10.1007/s11771-012-1146-7

Ender, T., Leurck, R.F., Weaver, B., *et al*., 2010. Systems of systems analysis of ballistic missile defense architecture effectiveness through surrogate modeling and simulation. *IEEE Syst. J.*, **4**(2):156-166. http://dx.doi.org/10.1109/JSYST.2010.2045541

Eusgeld, I., Nan, C., Dietz, S., 2011. "System-of-systems" approach for interdependent critical infrastructures. *Reliab. Eng. Syst. Safety*, **96**(6):679-686. http://dx.doi.org/10.1016/j.ress.2010.12.010

Fernando, B., Miguel, D., Juan, G.R., 2009. Fuzzy description logics under Godel semantics. *Int. J. Approx. Reas.*, **50**(3):494-514. http://dx.doi.org/10.1016/j.ijar.2008.10.003

Fotrousi, F., Fricker, S.A., Fiedler, M., 2014. Quality requirements elicitation based on inquiry of quality-impact relationships. 22nd IEEE Int. Requirements Engineering Conf., p.303-312. http://dx.doi.org/10.1109/RE.2014.6912272

Gao, J.X., Li, D.Q., Havlin, S., 2014. From a single network to a network of networks. *Nat. Sci. Rev.*, **1**(3):346-356. http://dx.doi.org/10.1093/nsr/nwu020

Ge, B.F., Hipel, K.W., Yang, K.W., *et al*., 2013. A data-centric capability focused approach for system-of-systems architecture modeling and analysis. *Syst. Eng.*, **16**(3):363-377. http://dx.doi.org/10.1002/sys.21253

Ge, B.F., Hipel, K.W., Fang, L.P., *et al*., 2014a. An interactive portfolio decision analysis approach for system of systems architecting using the graph model for conflict resolution. *IEEE Trans. Syst. Man Cybern.*, **44**(10):1328-1346. http://dx.doi.org/10.1109/TSMC.2014.2309321

Ge, B.F., Hipel, K.W., Yang, K.W., *et al*., 2014b. A novel executable modeling approach for system of systems architecture. *IEEE Syst. J.*, **8**(1):4-13. http://dx.doi.org/10.1109/JSYST.2013.2270573

Grigoroudis, E., Phillis, Y.A., 2013. Modeling healthcare system of systems: a mathematical programming approach. *IEEE Syst. J.*, **7**(4):571-580. http://dx.doi.org/10.1109/JSYST.2013.2251984

Guizzardi, G., 2005. Ontological Foundations for Structural Conceptual Models. PhD Thesis, Centre for Telematics and Information Technology, University of Twente, Enschede, the Netherlands.

Haimes, Y.Y., 2012. Modeling complex systems of systems with phantom system models. *Syst. Eng.*, **15**(3):333-346. http://dx.doi.org/10.1002/sys.21205

Holt, J., Perry, S., Payne, R., 2015. A model-based approach for requirements engineering for systems of systems. *IEEE Syst. J.*, **9**(1):252-262. http://dx.doi.org/10.1109/JSYST.2014.2312051

Huynh, T.V., Kessler, A., Oravec, J., 2011. Orthogonal array experiment in systems engineering and architecting. *Syst. Eng.*, **14**(2):208-222. http://dx.doi.org/10.1002/sys.20172

Khan, I., 2010. Methodology for the development of executable system architecture. Proc. 8th Int. Conf. on FIT, p.1-4. http://dx.doi.org/10.1145/1943628.1943677

Ma, Z.M., Zhang, F., Cheng, J., *et al*., 2011. Representing and reasoning on fuzzy UML models: a description logic approach. *Expert Syst. Appl.*, **38**(3):2536-2549. http://dx.doi.org/10.1016/ j.eswa.2010.08.042

Ma, Z.M., Li, Y., Zhang, F., 2012. Modeling fuzzy information in UML class diagrams and object-oriented database models. *Fuzzy Sets Syst.*, **186**(1):26-46. http://dx.doi.org/10.1016/j.fss.2011.06.015

Moynihan, R.A., Reining, R.C., Salamone, P.P., *et al*., 2009. Enterprise scale portfolio analysis at the National Oceanic and Atmospheric Administration (NOAA). *Syst. Eng.*, **12**(2):155-168. http://dx.doi.org/10.1002/sys.20116

Ncube, C., Lim, S.L., Dogan, H., 2013. Identifying top challenges for international research on requirements engineering for systems of systems engineering. 21st IEEE Int. Requirements Engineering Conf., p.342-344. http://dx.doi.org/10.1109/RE.2013.6636746

Office of the Deputy Under Secretary of Defense for Acquisition and Technology and Logistics (ODUSD (A&T)), 2008. Systems and Software Engineering, Systems Engineering Guide for Systems of Systems, Version 1.0. Technical Report No. ODUSD (A&T)SSE, Washington, D.C., USA.

Petersen, K., Khurum, M., Angelis, L., 2014. Reasons for bottlenecks in very large-scale system of systems development. *Inform. Softw. Techol.*, **56**(10):1403-1420. http://dx.doi.org/10.1016/j.infsof.2014.05.004

Regnell, B., Svensson, R.B., Olsson, T., 2008. Supporting roadmapping of quality requirements. *IEEE Softw.*, **25**(2):42-47. http://dx.doi.org/10.1109/MS.2008.48

Stoilos, G., Stamou, G., Pan, J.Z., *et al*., 2007. Reasoning with very expressive fuzzy description logics. *J. Artif. Intell. Res.*, **30**:273-320. http://dx.doi.org/10.1613/jair.2279

Wang, R., Dagli, C.H., 2011. Executable system architecting using systems modeling language in conjunction with colored Petri nets in a model driven systems development process. *Syst. Eng.*, **14**(4):383-409. http://dx.doi.org/10.1002/sys.20184