# Chapter 109
# A Hybrid Modeling Method for Service-Oriented C4ISR Requirements Analysis

**Ying ZHANG, Xiaoming Liu, Zhixue Wang and Li Chen**

**Abstract** With the rapid growth of complexity as well as the continuously innovational concept of network-centric warfare, it is difficult to analyze and design C4ISR systems based on the existing component technology or the object-oriented technology. However, both the workflow method and the AI planning method for service modeling are not very suitable for analysis and design service-oriented C4ISR requirements. This paper imports Service-Oriented Computing (SOC) design paradigm to C4ISR requirements analysis, and proposes a hybrid modeling method based on multi-ontologies. Accompanied with an instance, the method provides a new, flexible, reusable solution for C4ISR requirements evolution and system reconstruction.

**Keywords** SOC · C4ISR · Service modeling · Ontology

## 109.1 Introduction

Service modeling technology is one of the hot topics and has received significant attentions in SOC [1–4]. There are many researches proposed various methods, such as Refs. [5–13]. These methods could be divided into two kinds [14]. The one kind is based on the workflow technology. This method decompose the business process into activities, roles, rules and processes in order to integrate the enterprise comprehensive resources automatically or semi-automatically. The notable

Y. ZHANG (✉) · X. Liu · Z. Wang · L. Chen
Institute of Command Automation, PLA University of Science and Technology, Nanjing, 210007, China
e-mail: zhywl66@163.com

X. He et al. (eds.), *Computer, Informatics, Cybernetics and Applications*, Lecture Notes in Electrical Engineering 107, DOI: 10.1007/978-94-007-1839-5_109,

advantage of this method is modeling and compositing services with high efficiency. Another kind is AI planning method. Dynamic and flexibility are the notable advantages of this method.

However, those two kinds of service modeling are all not suitable for analysis and design service-oriented C4ISR (Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance) [15, 16]. Because the workflow method cannot satisfy the dynamic reconstruction and reconfiguration of C4ISR systems, while the AI planning method cannot deal with such huge scale and complex systems and application. In order to resolve this problem, we proposed a hybrid method combining the two methods mentioned above for C4ISR requirements analysis. The method could support the high-layer analysis by process reuse and low-layer analysis by dynamic service integration. It also support the reuse of models and services by domain knowledge based on multiple ontologies. This method provides a more efficient and flexible method for service-oriented C4ISR systems requirements analysis and modeling.

The rest of this article is organized as follows: Sect. 109.2 introduces the ontology definitions for service-oriented C4ISR requirements analysis. Section 109.3 presents the modeling process in detail with an instance. Section 109.4 draws some conclusions and future work.

## 109.2 Ontology Definitions

There are three reasons of defining multi-ontologies in this paper. Firstly, it is used to give a common semantic among domain experts. Secondly, it is needed to increase modeling efficiency by domain knowledge reuse. At last it could support model checking by logical reasoning.

Domain ontology is used to capture domain specific concepts and relations, which is defined as follows.

**Definition 1 Domain Ontology** Domain Ontology = < DomConcept, DomRelation, DomRule >

in which DomConcept is a finite set of domain specific concepts, while DomRelation is a finite set of domain specific relations associated with a pair of the concepts. DomRule is a finite set of domain model constraints specified in Description Logics [17]

Domain ontology forms domain knowledge which is related to common knowledge in a specific domain. It is the guide for constructing application ontology. We define application ontology as follows.

**Definition 2 Application Ontology** Application Ontology = < AppConcept, AppRelation, AppRef, AppRule >

in which AppConcept is a finite set of application concepts while AppRelation is a finite set of application defined relations. AppRef is a mapping function, specifying the mappings from AppConcept to Domconcept or from AppRelation

to DomRelation. AppRule is a finite set of application model constraints specified in Description Logics according to the rules of DomRule.

Service-oriented C4ISR requirements analysis and modeling regard service as the basic element to support various application of high system. Services could represent a variable logical scope and scale because its concerns could be large or small [18]. In order to support different granularity of service discovery and composition, this paper use activity ontology for coarse-grained modeling while service ontology for fine-grained modeling. We define activity ontology as follows.

**Definition 3 Activity Ontology** Activity Ontology = < ActConcept, ActRelation, ActFunction, ActAttribute, ActRef, ActRule >

in which AppConcept is a finite set of activity-related concepts, such as the name of an activity, sub-activities of an activity, etc. AppRelation is a finite set of activity relations, used to represent the relationship between activities, it also include hasSubAct and realized by relation. The former is used to represent the relationship between an activity and its sub-activity, while the latter represents the relationship between an activity and services contribute to realize it. ActFunction= Input∪ Output ∪ Precondition ∪ Effect∪ Attribute is a finite set of activity attributes. Input is the input parameter set of activity, and Output is the output parameter set of activity. Precondition is the prerequisite set of activity, while Effect is the result set of activity. Attribute is a finite set of activity attributes. ActRef is a mapping function, specifying the mappings from activity ontology to activity concept in domain ontology or application ontology. ActRule is a finite set of activity constraints specified in Description Logics according to the rules of AppRule or DomRule.

Service is the instance of service ontology which is a reusable assets published in service repository. A service usually has at least one function and can be reused in a variety of domain. We define service ontology as follows.

**Definition 4 Service Ontology** Service Ontology = < ServiceName, ServiceFunction, ServiceRef >

in which ServiceName is the name of a service. ServiceFunction = Input ∪ OutPut ∪ Precondition ∪Effect ∪ Attribute is functional description of the service. Input represents the input parameter set of the service, Output is the output parameter set of the service. Preconditions represent prerequisite set for service implementation. Effect is the effect set of the service. Attribute is a finite set of service attributes, generally used to describe the QoS properties of service. For C4ISR systems, the main considerations of QoS are reliability, availability, safety, timeliness, accuracy and so on. ServiceRef is a mapping function that maps service ontology to a service concept in activity ontology, or a service (or activity) concept in domain ontology.

## 109.3 Modeling Process

In this paper, we introduce a hypothetical instance of C4ISR systems to illustrate the service-oriented C4ISR requirements modeling. The instance is belonged to military logistics domain, and the specific application is maintaining a broken-down tank. Regarding domain ontology as a shared ontology, accompanied with application ontology, activity ontology and service ontology construct the domain knowledge repository for modeling. The modeling process consists of four phases, as follows.

**Phase 1**, capturing the domain knowledge that means domain conceptual modeling. Domain ontology defines all concepts and relations within a specific domain. Therefore, domain experts firstly need to define domain ontology when modeling domain model. In this paper, we use UML as the modeling language and example of the logistics domain model (fragment) is shown in Fig. 109.1. In the whole modeling process, all used concepts and relations should be within domain knowledge. If there are necessary concepts and relations are not defined in the domain knowledge repository, domain experts should add them to the repository.

**Phase 2**, creating the initial application model. Define the application ontology under the guide of domain ontology for the specific application. Application ontology describes concepts and relationships for specific application in domain. It is the static reusable asset of domain knowledge. Therefore, the application model also should be stored in the repository after built. In the future modeling process, modeler should firstly find whether there is available application model in domain knowledge repository or not, in order to improve the modeling efficiency by reuse. The initial application model for maintaining a broken-down tank is shown in Fig. 109.2. We use UML stereotype represent the mapping relationship between application concepts and domain concepts. For instance, the application concept "ThreeHours" is mapping from the domain concept "DesiredFinishedTime", and so on.

**Phase 3**, for each activity established in the initial application model, analyze and modeling activity process. Activities are divided into two kinds: simple activities and complex activities. The former is performed by an atomic activity while the latter is performed by several atomic activities. Based on activity ontology, the atomic activity is an activity that has no sub-activity. The detail steps are as follows: use Algorithm 1 (as shown in Fig. 109.3) in the activity repository to find whether there is existing activity process that could be reused this time or not. If there is, modeler could reuse the activity process directly. Otherwise, the modeler could modify existing activity process according to specific application requirements then reuse, or model a new activity process to supplement and improve the repository.

**Phase 4,** creating the final application model by matching the atomic activity and services. In this paper, we use activity represents coarse-grained modeling while service represents fine-grained modeling. Activity is implemented by activity process which consists of atomic activities. And atomic activity is realized

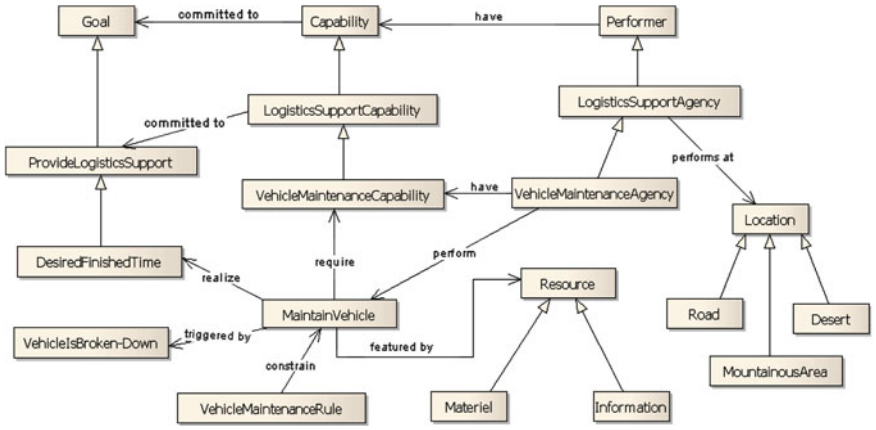


**Fig. 109.1** The logistics domain model (fragment)

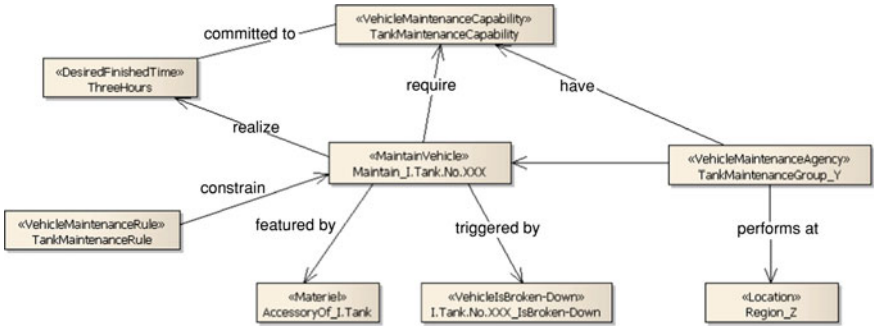


**Fig. 109.2** The initial application model of maintaining a broken-down tank

by service which could be single service or services composition. Therefore, this phase is including service discovery and composition.

Because the concepts and relations of activity ontology and service ontology are all included in the same domain knowledge repository, we could regard atomic activity as service requester. The activity function describes inputs, outputs, pre-conditions and effects of an activity could be regarded as service request description. Then according to the service request, discovery in the service repository if there are services could satisfy the request or not, i.e. service discovery.

Semantic similarity is an important facet of service discovery. We divide service matching into three aspects: name matching, inputs/outputs matching and attributes matching. Inputs and outputs matching is the keyword item among all aspects. Based on the shared ontology, i.e. domain ontology, locate concepts of

ALGORITHM 1 Activity Process Construction Algorithm

INPUTS: activity set $AppActSet$ which is modeled in initial application model

OUTPUTS: the atomic activities set $APSet$ which implement the activities in $AppActSet$

INITIALIZATION: load the activity repository of domain knowledge repository, $APSet = NULL$, let the current activity matched atomic activities set $AP = NULL$, i.e. $AP.ActSet = NULL$, $AP.RelaSet = NULL$.

ALGORITHM STEPS:

Step1: Select the first activity in $AppActSet$ as the current activity $Activity$

Step2: Retrieve the $Activity$ by its name in activity repository to find it is included or not. If not, do Step3. Otherwise, do Step4.

Step3: There is no reusable activity process for $Activity$ in current activity repository. Therefore, modeler need to build new concepts, relations, attributes, rules of $Activity$ based on activity ontology definition, then do Step7.

Step4: There is reusable activity process in current activity repository for $Activity$. Check whether $Activity.hasSubAct$ is empty or not. If it is empty, do Step5. Otherwise, do Step6.

Step5: The current activity $Activity$ is a simple activity, add $Activity$ to $AP.ActSet$, then do Step7.

Step6: The current activity $Activity$ is a complex activity, and it has reusable activity process, add all sub-activities of the process to $AP.ActSet$, while add the corresponding relations between sub-activities to $AP.RelaSet$, then do Step7.

Step7: Add $AP$ to $APSet$, complete the processes of current activity. Then select the next activity in $AppActSet$ do Step2 again, until process all activities in $AppActSet$

Step8: Return $APSet$

**Fig. 109.3** The activity process construction algorithm

activity ontology and service ontology in domain ontology, then compute the semantic similarity between the concepts in domain ontology. Currently, there are many domestic and foreign scholars have researched various methods for computing semantic similarity, which is out of this paper's scope. We use the computing algorithm proposed in Ref. [19], the semantic similarity between concept A and concept B is calculated as follows, shown in Eq. 109.1. Max represents the max distance from the root node to any leaf node. Len(A,B) represents the shortest distance between concept A and concept B.

$$sim(A,B) = \frac{2Max - len(A,B)}{2Max} \tag{109.1}$$

The detail matching method is as follows. For each atomic activity, use the service composition algorithm based on the canonical structure, which proposed

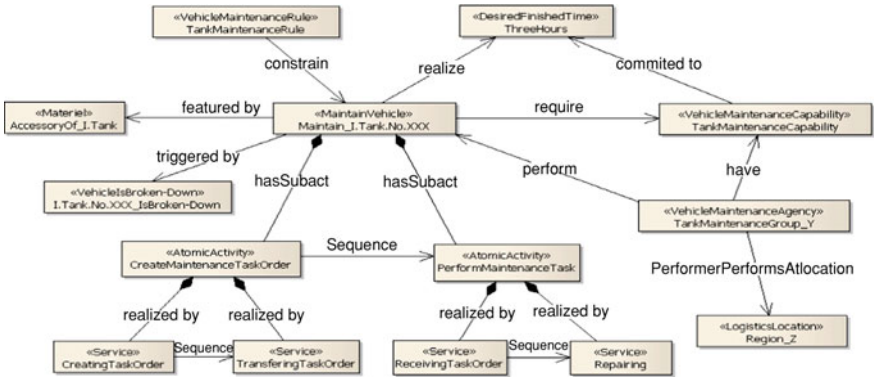


**Fig. 109.4** The final application model of maintaining a broken-down tank

by our team members in previous work [20], to find the matched service set which could implement the atomic activity. If the matching is fail, then clue the modeler either modify the inputs and outputs then matching one more time or define a new service. Otherwise, if the matching successfully with only one service set, it is just the matched service we need. Or, if there are more than one service sets when matching is successful, which means there are several optional solution for performing the activity, thus the models could choice the one which as the biggest value computed by Eq 109.2, or choice by his opinion.

$$\begin{aligned} Similarity_{aux}(Act, Ser) = {} & w_1 * Sim_{Name}(Act.Name, Ser.Name) \\ & + w_2 * Sim_{Fun}(Act.Precondition, Ser.Precondition) \\ & + w_3 * Sim_{Fun}(Act.Effect, Ser.Effect) \\ & + w_4 * Sim_{Fun}(Act.Attribute, Ser.Attribute) \end{aligned} \tag{109.2}$$

in which $w_i, 1 \le i \le 4$ is the weight. $Sim_{Name}(Act.Name, Ser.Name)$ represents the semantic similarity between service names, $Sim_{Fun}(Act, Ser)$ represents the semantic similarity between service function that including precondition similarity, effect similarity and attribute similarity, which are described respectively as $Sim_{Fun}(Act.\Pr econdition, Ser.\Pr econdition)$, $Sim_{Fun}(Act.Effect, Ser.Effect)$, and $Sim_{Fun}(Act.Attribute, Ser.Attribute)$.

At last, composite all the services into the initial application model, then obtain the final application model with service as the basic granularity, as shown in Fig. 109.4. It is noteworthy that the application model must consistent with the domain model. We ensure the correctness and consistency by ontology reasoning technology and model checking with the Description Logic [21].

## 109.4 Conclusions and Future Works

This paper studies how to apply SOC with its reusable, flexible, loose couple features to C4ISR capability analysis such as C4ISR systems. We propose a hybrid method for service-oriented C4ISR modeling based on multiple ontologies. Service-oriented C4ISR modeling process focuses on the logical modeling rather than the technical details of application implementations. Therefore, if the higher requirement has changed, the modeler can reconstruct the application model flexibly and fast through invoking and integrating services. In addition, this method supports updating services that the new and better services could replace the old ones without influencing the high-level model.

The future work will include supplement the domain knowledge repository in order to improve the efficiency of reuse, the validation methods to ensure the consistency of the modeling better, as well as study how to map the model to code generation in the system design phase under the guide of domain knowledge.

## References

1. Huhns MN, Singh MP (2005) Service-oriented computing: key concepts and principles [J]. IEEE Internet Comput (244):021–028
2. Benatallah B, Motahari Nezhad HR (2005) Service oriented computing: opportunities and challenges, LNCS, vol 3372, pp 1–8
3. Papazoglou MP, Traverso P, Dustdar S et al (2008) Service-oriented computing: a research roadmap [J]. Int J Co-op Inf Syst 17(2):223–255
4. Papazoglou MP, Schmidt JW, Mylopoulos J (2009) Service-Oriented Computing [M]. MIT Press Cambridge
5. Wu B, Jin Z, Zhao B (2008) A modeling approach for service-oriented application [C]. In: Proceedings of the IEEE international conference on web services
6. Lee SY, Lee JY, Lee BI (2006) Service composition techniques using data mining for ubiquitous computing environments. Int J Comput Sci Netw Secur [J] 6(9):110–117
7. Ma Y, Jin B, Feng Y (2005) Dynamic discovery for semantic Web services based on evolving distributed ontologies [J]. Chin J Comput 28(4):603−615 (in Chinese with English abstract)
8. Qiu L, Shi Z, Lin F (2006) Context optimization of AI planning for services composition [C]. In: Proceedings of the IEEE international conference on e-business engineering, pp 610–617
9. Bottaro J, Bourcier C, Escoer et al (2007) Autonomic context-aware service composition [C]. In: Proceedings of 2nd IEEE international conference on pervasive services
10. Mingkhwan P, Fergus O, Abuelma'Atti et al (2006) Dynamic service composition in home appliance networks. Multimedia Tools Appl [J] 29(3):257−284
11. Viroli M, Denti E, Ricci A (2007) Engineering a BPEL orchestration engine as a multi-agent system [J]. Sci Comput Program 66:226–245
12. Chi YL, Lee HM (2008) A formal modeling platform for composing web services [J]. Expert Syst Appl 34:1500–1507
13. Valero V, Emilia Cambronero M, Díaz G et al (2009) A petri net approach for the design and analysis of Web Services Choreographies [J]. J Log Algebraic Progr 78:359–380
14. Deng S (2007) Research on automatic service composition and formal verification [D] (in Chinese). Ph.D thesis of Zhejiang University

15. Liao S, Sun B, Wang R (2003) A knowledge-based architecture for planning military intelligence, surveillance, and reconnaissance [J]. Space Policy 19(3):191–202
16. US Department of Defense. DoD Architecture Framework Version 2.0 [R]. 2009
17. Baader F, Calvanese D, Mcguinness DL et al (2003) The Description Handbook [M]. Cambridge University Press, Cambridge
18. Erl Thomas (2005) Service-oriented architecture (SOA):concepts, technology, and design [M], Pearson Education
19. Resnik P (1999) Semantic similarity in a taxonomy: an information-based measure and its application to problems of ambiguity in natural language [J]. J Artif Intell Res 11:121–126
20. Chen L, Song Z, Zhang Y et al (2010) A method of semantic web service composition based on the canonical structure[J]. J Inf Comput Sci 7(2):463–469
21. Dong Q, Wang Ze, Chen J et al (2010) Method of checking capability model based on description logic [J]. Syst Eng Electron 32(3):533–539 (in Chinese with English abstract)