Based on Capability of C4ISR Architectural Modeling and Analysis

Ting-ting ZHANG¹, Hui LI¹ and Yi YAO¹

¹Software Engineering Department PLA University of Science and Technology, JiangSu, NanJing, China

Keywords: C4ISR Architecture, capability requirements, UML modeling, Description Logic

Abstract. C4ISR system is complex system. Such system’s architecture and complex needs of many. Then complex system’s capability requirements are difficult to catch. Description of capability requirements are usually imprecise and do not support architecture analysis.

The paper uses an extended UML modeling technique to analyzing the requirements of C4ISR capabilities. It also introduces a new model convert domain model into the Description Logic ontology. Use Description Logic ontology description the standard DoD Architecture Framework, the proposed method may populate C4ISR architectural analysis.

Introduction

Although the term capability requirements frequently appear in various official documents, there is no official definition for it. According to TOGAF, the capability is defined as “an ability that an organization, person, or system possesses” [1]. Capabilities are typically expressed in general and high-level terms and typically require a combination of organization, people, processes, and technology to achieve, for example, marketing, customer contact, or outbound telemarketing. A requirement, according to IEEE Std610.12, is a condition or capability needed by a user to solve a problem or achieve an objective [2]. This definition leads to two principal categories of requirements: capability requirements and constraint requirements. The capability requirements describe, from the user view, the process to be supported by information systems, and they should be qualified with values of capacity, speed, and accuracy [3]. We refer to it as the user requirements of functions and performance of a complex system or SoS (System of systems) such as C4ISR systems. Usually, an appropriate architecture framework such the DoD AF, MODAF or TOGAF is introduced for modeling the capability requirements and an architectural design for the whole system.

However, some problems might arise if UML is directly applied to the capability based architectural analysis. Firstly, it is a semi-formal and weakly constrained modeling language. As a result, the built models may suffer for inconsistency and concept conflicts, and it is hard to find a logic inference engine to do automatic model check. Secondly, based on abstract and domain-meaningless symbols or constructs, UML is of poor domain applicability [4] and thus the modelers find it difficult to model the capability concepts adequately. Thirdly, when a large-scale C4ISR system is broken down into a number of sub-projects which are undertaken by different sub-contractors, the requirements engineers would debate on some domain-specific concepts which means same things but are understood or explained differently in different systems. Consequently, it would be difficult to integrate and verify the separately built models as a whole.

The paper is organized as follows. Section 2 defines the domain modeling framework for describing the C4ISR capabilities. Section 3 explains in detail the algorithm converting the domain models into the ontology of SHOIN(D) to enable automatic model checking. Section 4 demonstrates the applicability of the method through a case study. And the last section brings a summary.

The research is supported in part by the National High-Tech Research and Development Program of China under the project number 2007AA01Z126, the National Defense Advanced Research Program of China under the project number 51306010202 and the National Natural Science Foundation of China the project number 61273210.
The Framework for C4ISR Requirements Elicitation

A Domain Modeling Framework

As we know domain knowledge is important in requirements elicitation and analysis. The Canadian research agency believes that C4ISR capability analysis requires the expertise and exploits technologies based on the domain knowledge [5]. However, how to capture and describe domain knowledge is a difficult work. To solve the problem, we suggest a domain modeling framework to domain knowledge elicitation and modeling. The figure1 shows the modeling framework.

The top layer, framework layer, reflects the meta-concepts and relations for C4ISR requirements. The concepts and relations, as shown in figure2, derive from the Meta-Model Data Groups of DoD AF.

The second layer, the domain level, describes the domain concepts and relations for C4ISR capability requirements. For the domain of army command and control management, for example, the ArmedCar (shown in figure1) is a domain concept of capability requirements featured by the type of vehicle used in the army operation and attributes like weight, capacity, maximum speed and operation range of the armed car. The domain concepts and relations are the instances of meta-concepts and relations of the framework layer.

How to Capture the Domain Knowledge

The framework layer only gives the definition of the C4ISR capability meta-concepts and provides meta semantics for description of C4ISR capability requirements. It could work as a semantic restriction, but how to capture the domain-specific concepts and relations is an outstanding issue.

Here we argue that the C4ISR requirements elicitation be initiated with mission analysis by modeling task goals. The original task goals could be break down into functional goals, which are realized by a series of activities. Through the activity modeling and analysis, the domain experts can capture the activity related domain-specific concepts, such as entity, capability, system, resource and so on.
In this way, the domain experts could capture the domain-specific concepts and relations within the DoD Architecture Framework. But there will be consistency and rationality problems in the domain knowledge. The consistency problem means that the domain knowledge has a logical contradiction or breaks the semantics of Meta-Model of the DoD AF. The rationality problem means that the task goal is unreasonable according to the domain knowledge, such as the air-defense unit desires to perform the air-defense activity (ADA), but it does not have the ADA related capabilities. These problems are hard to be checked by manual method. We propose to convert the domain models into domain ontology, specified in Description Logic, and verify the consistency and rationality by an automated inference engines like Pellet and Racer. The details of the model checking method will be introduced in section3.

How to Model the Domain Knowledge

UML is found out of some shortcomings for domain applicability [6]. The literature [7] suggests a construction method of domain-specific modeling language (DSL) based on UML Profile mechanism.

With the similar idea, we define new stereotypes of UML constructs according to the Meta-Model of DoD AF to provide a capability oriented DSL for C4ISR architecture modeling. Specifically, the UML Class at MOF (Meta Object Facility) level is elaborated as the meta-concepts such as “Capability”, “Resource”, “Activity” etc. and the Association as meta-relations such as “Perform”, “Realize”, “Conflic_with” etc. The figure3 gives an example of the city air defense domain model, which reflects domain-level requirements for the City Air Defense system. Furthermore, the domain model can be exported as a new UML extension profile for the program level requirements modeling.

Domain Model Checking Based on DL

Domain Model Conversion

Daniela et al [8]. present a way of translating a UML model into the DL ontology and reasoning on DL. Upon their work, we provide an algorithm as follows to convert the capability requirements models into the capability knowledge ontology, specified in SHOIN(D) a subsystem of DL.

Algorithm: convert the UML domain model into the domain ontology

Input: Meta-Model Data Groups; domain models

Output: domain ontology

Step1: create the concepts and their instances.
For every meta-concept, create a same name concept C in Tbox.
For every domain concept Cw, which is mapped to meta-concept C, create a same name instance Cw in Abox, and declare the be-instance-of relationship between Cw and C.
Step 2: establish the relationships among concepts and instances.
For every meta-relation $R(C_1, C_2)$, create in Tbox a same name relation $R$ and then add in Tbox a SHOIN(D) rule: $C_1 R C_2$.
For every domain relation $Rw(Cw_1, Cw_2)$, which is mapped to a meta-relation $R$, create in Abox the relation $R(Cw_1, Cw_2)$.

Step 3: establish the generalization among concepts and instances.
For every meta-generalization $R(C_1, C_2)$, create in Tbox a relation named Super-sub, and then add in Tbox such rules: $C_2 C_1$, $C_2$ Super-sub $C_1$.
For every domain generalization relation $Rw(Cw_1, Cw_2)$, create in Abox the relation Super-sub$(Cw_1, Cw_2)$.

Model Restriction Rules
The model restriction rules are some domain-general restrictions in capability analysis of C4ISR system. They can be used for checking the consistency and rationality of the domain model. In our method, we recommend describing the model restriction rules with nRQL, a DL supported ontology query language, which can be executed on the inference engine such Racer to find the problem related concepts and relations. Followings are some of the rules described with nRQL.

Consistency Restriction Rules
[CR1] A and B are meta-concepts, and a and b are their instances in domain layer. If there is a generalization between A and B, then there should be a generalization between a and b. It is described by nRQL as “retrieve nil (?a ?b Generate)”, where Generate denotes the generalization between A and B.
[CR2] The domain experts can define some basic meta-concepts in the framework layer. Once defined these concepts, the basic meta-concept should have at least one instance in domain layer. It is described by nRQL as “retrieve nil (?x BasicConcept)”, where BasicConcept denotes the basic meta-concept.
[CR3] The domain experts can define some basic meta-relations in the framework layer. Once defined these relations, the basic meta-relation should have at least one instance in domain layer. It is described by nRQL as “retrieve nil (?x ?y BasicRelation)”, where BasicRelation denotes the basic meta-relation.
[CR4] A and B are meta-concepts, and a and b are their instances respectively. If A and B do not have any relation with each other, then a and b can not have any relation. It is described by nRQL as “retrieve nil (and (?a A)(?b B)(?a ?b Relation))”, where BasicRelation denotes the basic meta-relation.

Rationality Restriction Rules
[RR1] Any domain goal should be realized by a domain activity. It is described by nRQL as “retrieve nil (Goal ?Y BeRealized )”, where Goal denotes the domain goal concept, BeRealized comes from Meta-Model Data Groups.
[RR3] C1 and CK are domain capabilities, supported by domain activities A1 and AK respectively. If C1 is a complex capability, containing CK, then AK must be performed before A1. It is described by nRQL as “retrieve(?AK ?A1) (and(?C1 ?CK Contain) (?A1 ?C1 Need)( ?AK ?CK Need)(A1 AK Precedence))”.
[RR4] C and C1 are domain capabilities, supported by domain activities A and A1 respectively. If C is an abstract capability, and C1 is its concrete capability, then A1 must be performed before A. It is described by nRQL as “retrieve(?A ?A1)(and (?C ?C1 Generate) (?A1 ?C1 Need)( ?A ?C Need) ( ?A ?A1 Precedence))”, where Generate denotes the generalization between C and C1.
[RR5] A is a domain activity, which is performed by the domain entity E. If A need the support of domain capability C, then the entity E must has the capability C. It is described by nRQL as “retrieve (?E ?C) (and (?E ?C (not Has)) (?E ?A Perform)(?A ?C Need ))”, where Has, Perform comes from Meta-Model Data Groups.

[RR6] E1 and E2 are domain entities, belonging to different operation nodes N1 and N2 respectively. If there is a relationship between E1 and E2, then there must be a communication between N1 and N2. It is described by nRQL as “retrieve (?N1 ?N2) (and (?E1 ?N1 Belong) (?E2 N2 Belong) (?E1 ?E2 Relate) (?N1 ?N2 not Communicate ) )”, where Aggregation, Relate, Communicate comes from Meta-Model Data Groups.

Here we suggest a limited set of model restriction rules, but it suggests an applicable way of domain rules formalization. The domain experts can define their own model restriction rules to achieve a desired model checking effects.

Domain Model Checking

Based on the model conversion algorithm, the domain model is converted into domain ontology. Besides, these model restriction rules are converted into reasoning restriction rules. Therefore, the consistency and rationality checking can be automated with the inference engine. The reasoning will be completed by the engine in a few seconds. The core algorithm of automated inference engine is the algorithm Tableau, details of which can be further referred to [9].

A Case Study

Continuing the example of the city air defense domain analysis in section 2, let us examine the model consistency and rationality in the way introduced in section 3. The figure3 shows a fragment of domain model of the city air defense system.

Here we take a model rationality checking as an example to examine the practicality of the method. Suppose that the modeler has missed the precedence relationship between the domain activity A1 and A21, and wrongly added a precedence relationship between the domain activity A23 and A1. As a result, the domain model breaks three rationality restriction rules (i.e. RR2, RR3 and RR4). Based on our model conversion and checking method, we apply the inference engine Racer to verify these problems. The figure4 shows that Racer retrieves the ontology and has found those pairs of problem concepts (A1, A21) and (A1, A23) which break the rationality restriction rules.
Conclusions
The paper proposes an approach of capability requirements elicitation and analysis. To capture and model domain knowledge, it proposes a domain modeling framework and suggests that the domain knowledge should be built under the constraints of Meta-Model Data Groups of DoD AF with the help of domain-specific modeling language. To guarantee the consistency and rationality of domain models, the domain models are converted into ontology specified in SHOIN (D), and model restriction rules are defined to provide the semantic constraints necessarily held by all domain concepts and relations in the capability models. These rules can be then converted into ontology restriction rules specified in nRQL. Finally, domain model checking is automated by the inference engine such as Pellet and Racer. Based on the research of the literature [10], our further research will be on enriching the model restriction rules to guarantee the quality of the requirements model.

References