SoS Architectural Requirements Modeling and Analysis

Dong Qingchao¹ *, Xu Junming¹, Qiao Yongjun¹, Ren Huifeng²
¹Naval Aviation University, YanTai 264001, China
²Nanjing University of Information Science & Technology, WuXi 2141051, China
dongqingchao001@163.com

Abstract. Analysing the architectural requirements of System-of-Systems is a daunting task, usually involving a complicated process of enterprise architecture modeling and verifying. So the paper proposes a method of capability-oriented requirements modeling for developing the architecture requirements of System-of-Systems. A multi-layer ontologies framework is suggested to model the architecture requirements. To check the consistency of performance requirements, a formal inference method is designed to verify the requirements models. Finally, a case study is provided to demonstrate the applicability of the method.

1. Introduction

System-of-Systems (SoS), such as command, control, communication, computers, intelligence, surveillance, and reconnaissance (C4ISR) system [1,2], are growing in complexity and importance as more computing devices are networked together to help automate tasks previously done by human operators. One of the key issues is to model the SoS architecture requirements and automate the verification process. However, analyzing the architectural requirements of SoS is a daunting task, usually involving a complicated process of enterprise architecture development using some standard architecture frameworks such as the DoD Architecture Framework (DoDAF) [3] or the MOD Architecture Framework (MODAF) [4].

In this paper, we introduce a method of capability-oriented requirements modeling for developing the architectures requirements of SoS. The architecture requirements can be modeled within a multi-layer ontologies framework, starting from the capability requirements at strategic level down to the domain requirements and IT implementation. We take the C4ISR system as a typical SoS, and present a formal method of modeling SoS architecture requirements based on UML. To enable automatic performance objectives conflict detection, we use the modal logic to formalize the requirements.

2. Capability requirements and multi-layer ontologies framework

2.1 Multi-layer ontologies framework

A requirement is a condition or capability needed by a user to solve a problem or achieve an objective [5]. 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. In another word, they address both functions and performance of what the users want of the to-be systems.
As introduced above, we adopt both ontology and UML techniques to model the enterprise architecture\cite{6,7}. The enterprise architecture requirements analysis and architecture development is processed in two phases and the capability requirements are modeled within a multi-layer modeling framework. The top level, called framework level, reflects fundamental concepts of the enterprise architecture framework and provides the meta-model for the enterprise modeling. Particularly, for developing the architectures of the C4ISR systems, the meta-model can be defined, as shown in figure 1, according to the Capability Meta Model (CMM) of DoDAF2.0. However, the CMM of DoDAF2.0 only concentrates on the functional requirements related concepts, lacking of performance requirements related concepts. Then, we extend the CMM by introducing the performance related concepts, such as Performance and Desired effect. The figure 1 reflects only parts of the extended CMM, since we cannot show the whole model including all concepts and relations in the limited pages.

![Figure 1](image)

**Figure 1.** Performance related concepts and relations in the CMM

The second layer, the domain level, reflects the domain-specific concepts. For the domain of army command and control management, for example, the ArmedCar 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 third layer reflects the application concepts of the capability requirements, which results in the final products of capability requirements analysis. The concepts at this layer are related to real objects and stereotyped by the domain concepts. For example, Saracen FV603 is an application concepts stereotyped by the domain concept ArmedCar.

### 2.2 Performance related relationships

Analyzing the SoS capability requirements is a daunting task, usually involving a complicated process of modeling and checking. Many believe that there are two typical kinds of relationships among SoS performance objectives, And Decompose and Or Decompose\cite{8,9}.

And Decompose, if the performance E is decomposed by the other performances, such as E₁…En, and the realization of E depends on the realization of each one of E₁…En, then there is an And decompose between them. It can be described by such an expression \( E_1 \land \ldots \land E_n \rightarrow E \).

Or Decompose, if the performance E is decomposed by the other performances, such as E₁…En, and the realization of E depends on the realization of any one of E₁…En, then there is an Or decompose between them. It can be described by such an expression \( E_1 \lor \ldots \lor E_n \rightarrow E \).

According to KAOS \cite{10}, we introduce another eight kinds of relations into performance analysis. Support+ relation, the promotion of one performance objective may lead to the promotion of another performance objective.

Support++ relation, the promotion of one performance objective will lead to the promotion of another performance objective.
Support- relation, the reducing of one performance objective may lead to the promotion of another performance objective.

Support-- relation, the reducing of one performance objective will lead to the promotion of another performance objective.

Conflict+ relation, the promotion of one performance objective may lead to the reducing of another performance objective.

Conflict++ relation, the promotion of one performance objective will lead to the reducing of another performance objective.

Conflict- relation, the reducing of one performance objective may lead to the reducing of another performance objective.

Conflict-- relation, the reducing of one performance objective will lead to the reducing of another performance objective.

3. Capability requirements modeling and analysis

The interoperability issue might arise when multiple system models exist and the models are built by different modelers with different tools, and therefore model analysis is needed. We focus on the conflict hidden in the performance objectives, and suggest an approach of logic reasoning to verify consistency of the concepts of capability requirements through logic inference. We have pointed out that the conflicts among performance objectives would lead to a failure of SoS implementation, and the conflicts can be classified into two kinds: direct conflict and potential conflict.

Definition 1 Direct conflict: Direct conflict refers to that the change of one performance objective will lead to the reducing of another performance objective, such as Conflict++. In capability requirements analysis, all the direct conflicts should be avoided.

Definition 2 Potential conflict: Potential conflict refers to that the change of one performance objective may lead to the reducing of another performance objective, such as Conflict+. In capability requirements analysis, all the potential conflicts should be detected and pointed out to the modeler.

The semantic of performance objective conflict is transitive, so analyzing the conflict is a daunting task, involving a complicated process of semantic reasoning. To detect all the possible conflicts, we suggest four kinds of performance objective reasoning rules: weakening rule, contradictory rule, elimination rule and transitive rule.

Definition 3 Weakening rule (WR): there is a strong and a weak influence between different performance objectives at the meantime, then the influence can be weakened. It includes two rules. Here the symbol $S^+$ denotes Support+. And so on, for each of the other symbols.

$$WR1: \frac{E_1 \xrightarrow{S^+(C^-)} E_2, \ E_1 \xrightarrow{S^+(C^-)} E_2}{E_1 \xrightarrow{S^+(C^-)} E_2} \Rightarrow E_1 \xrightarrow{S^+(C^-)} E_2$$

$$WR2: \frac{E_1 \xrightarrow{S^+(C^-)} E_2, \ E_1 \xrightarrow{S^+(C^-)} E_2}{E_1 \xrightarrow{S^+(C^-)} E_2} \Rightarrow E_1 \xrightarrow{S^+(C^-)} E_2$$

Definition 4 Contradictory rule (CR): there is a positive and a negative influence between different performance objectives at the meantime, then it is a serious contradiction. It includes four rules.

$$CR1: \frac{E_1 \xrightarrow{S^+(C^-)} E_2, \ E_1 \xrightarrow{S^+(C^-)} E_2}{E_1 \xrightarrow{S^+(C^-)} E_2} \Rightarrow \text{Contradiction}$$

$$CR2: \frac{E_1 \xrightarrow{S^+(C^-)} E_2, \ E_1 \xrightarrow{S^+(C^-)} E_2}{E_1 \xrightarrow{S^+(C^-)} E_2} \Rightarrow \text{Contradiction}$$

$$CR3: \frac{E_1 \xrightarrow{S^+(C^-)} E_2, \ E_1 \xrightarrow{S^+(C^-)} E_2}{E_1 \xrightarrow{S^+(C^-)} E_2} \Rightarrow \text{Contradiction}$$

$$CR4: \frac{E_1 \xrightarrow{S^+(C^-)} E_2, \ E_1 \xrightarrow{S^+(C^-)} E_2}{E_1 \xrightarrow{S^+(C^-)} E_2} \Rightarrow \text{Contradiction}$$
**Definition 5** Elimination rule (ER): the change of one performance objective will not influence another performance objective, and then the relationship between them can be deleted. It includes two rules.

ER1: $E_1 \xrightarrow{S^+(C^+)} E_2$, $E_1 \xrightarrow{S^-(C^-)} E_2 \Rightarrow$ Delete the relationship

ER2: $E_1 \xrightarrow{S^+(C^+)} E_2$, $E_1 \xrightarrow{S^-(C^-)} E_2 \Rightarrow$ Delete the relationship

**Definition 6** Transitive rule (TR): it refers to that the transitive influence between three or more performance objectives. It includes eight rules.

TR1: $E_1 \xrightarrow{C^+(S^-)} E_2$, $E_2 \xrightarrow{C^+(S^-)} E_3 \Rightarrow E_1 \xrightarrow{C^+(S^-)} E_3$, if any relationship in the precondition is weak effect, the conclusion would be weak.

TR2: $E_1 \xrightarrow{S^+(C^-)} E_2$, $E_2 \xrightarrow{S^+(C^-)} E_3 \Rightarrow E_1 \xrightarrow{S^+(C^-)} E_3$

TR3: $E_1 \xrightarrow{S^+(C^-)} E_2$, $E_2 \xrightarrow{S^+(C^-)} E_3 \Rightarrow E_1 \xrightarrow{S^+(C^-)} E_3$

TR4: $E_1 \xrightarrow{S^+(C^-)} E_2$, $E_2 \xrightarrow{S^+(C^-)} E_3 \Rightarrow E_1 \xrightarrow{S^+(C^-)} E_3$, if any relationship in the precondition is weak effect, the conclusion would be weak.

TR5: $E_1 \xrightarrow{S^-(C^+)} E_2$, $E_2 \xrightarrow{S^-(C^+)} E_3 \Rightarrow E_1 \xrightarrow{S^-(C^+)} E_3$

TR6: $E_1 \xrightarrow{S^-(C^+)} E_2$, $E_2 \xrightarrow{S^+(C^-)} E_3 \Rightarrow E_1 \xrightarrow{S^+(C^-)} E_3$, if any relationship in the precondition is weak effect, the conclusion would be weak.

TR7: $E_1 \xrightarrow{S^-(C^+)} E_2$, $E_2 \xrightarrow{C^+(S^-)} E_2 \Rightarrow E_1 \xrightarrow{C^+(S^-)} E_2$

TR8: $E_1 \xrightarrow{S^-(C^+)} E_2$, $E_2 \xrightarrow{C^+(S^-)} E_2 \Rightarrow E_1 \xrightarrow{C^+(S^-)} E_2$, if any relationship in the precondition is weak effect, the conclusion would be weak.

It is appropriate to apply Modal Logic reasoning system [10] to describe the semantics of these rules. Take Support+ relation between performance E1 and performance E1 as an example, the semantic can be described in such a modal logic expression: $\textit{Enhance}(E_1) \rightarrow \square \textit{Enhance}(E_2)$. So we apply the Modal Logic reasoning system to illustrate the correctness of these performance objective reasoning rules, and take the TR2 as an example.

Table 1. Correctness proofing process of the TR2 rule

| Assertions | Reasoning condition |
|------------|---------------------|
| 1 $\textit{Enhance}(E_1) \rightarrow \square \textit{Enhance}(E_2)$ | Known condition |
| 2 $\textit{Enhance}(E_2) \rightarrow \square \textit{Reduce}(E_3)$ | Known condition |
| 3 $\square \textit{Enhance}(E_2) \rightarrow \square \square \textit{Reduce}(E_3)$ | Applying Theorem 8.4 [11] for the assertion 2 |
| 4 $\textit{Enhance}(E_2) \rightarrow \square \square \textit{Reduce}(E_3)$ | Applying Transfer Theorem [11] for the assertion 1 and assertion 3 |
| 5 $\textit{Enhance}(E_2) \rightarrow \square \textit{Reduce}(E_3)$ | Applying Idempotent Theorem [11] for the assertion 4 |
| 6 END | |

4
With these performance objective reasoning rules, we can reason and detect all the potential errors among performance objective of SoS capability requirements. To realize a reliable performance requirements analysis.

4. A case study

In the following section, the architecture requirements model, shown in figure 2, concentrates on the performance character of the capability requirements. A few of them, such as the performance of Effective strike, which are critical to the mission of Aerial target strike. Here we pay attention to the conflict detection of the performance objectives. For the model shown in figure 2, we can detect a potential conflict through different inference paths.

Path1: the promotion of Strike target quantity objective may weaken the Accurate recognition rate objective; the reducing of Accurate recognition rate objective may weaken the Target effective early warning objective; the reducing of Target effective early warning may weaken the Effective strike objective. Then we can infer that the promotion of Strike target quantity objective may weaken the Effective strike objective.

Path2: the promotion of Strike target quantity objective may raise the Missile strike effect objective; the promotion of Missile strike effect objective will raise the Target effective strike objective; the promotion of Target effective strike objective will raise the Effective strike objective. Then we can infer that the promotion of Strike target quantity may raise the Effective strike objective.

From these two reasoning paths, we can find a conflict that the promotion of Strike target quantity objective may raise and weaken the Effective strike objective at the same time. For capability performance analysis, the capability requirements is built from the domain model and the application model. Table 2 shows the symbolic representation of the capability performance objectives. Table 3 shows the conflict detection reasoning process.

Table 2. Symbolic Representation of the Capability Performance Objectives

| Concept                        | Symbolic | Concept                        | Symbolic |
|--------------------------------|----------|--------------------------------|----------|
| Effective Strike               | E        | Artillery strike effect        | E21      |
| Target effective early warning | E1       | Missile strike effect          | E22      |
| Target effective strike        | E2       | Hit rate                       | E221     |
| Target effective damage        | E3       | Strike target quantity         | E222     |
| Recognition time | Operation time |
|------------------|----------------|
| E11              | E223           |
| Surveillance range | Damage degree |
| E12              | E24            |
| Accurate recognition rate | Strike range |
| E13              | E225           |

Table 3. Conflict Detection Reasoning Process

| Assertions | Reasoning condition |
|------------|---------------------|
| 1          | < E222, E13>: Conflict+ Known condition |
| 2          | < E1, E1 >: Conflict- Known condition |
| 3          | < E1, E2>: Conflict- Known condition |
| 4          | < E22, E2>: Support+ Known condition |
| 5          | < E2, E2>: Support++ Known condition |
| 6          | < E22, E1>: Conflict+ Applying TR5 for the assertion 1 and assertion2 |
| 7          | < E22, E2>: Conflict+ Applying TR5 for the assertion 3 and assertion 6 |
| 8          | < E22, E2>: Support+ Applying TR4 for the assertion 4 and assertion 5 |
| 9          | Conflict Applying CR4 for the assertion 7 and assertion 8 |

It reflects only a small fragment of the case for capability performance analysis, since the reasoning process can be automatically operated by the logic inference engineering, just like the Pellet and Racer inference reasoners [12]. Then the system designer can get an automatic performance objectives conflict detection to analyses the architectural requirements of SoS.

5. Conclusion
An analyzing the architectural requirements of SoS is a daunting task, usually involving a complicated process of enterprise architecture modeling and analysis. In this paper, we introduce a method of capability-oriented requirements modeling for developing the architecture requirements of SoS. It first provides a multi-layer ontologies framework to model the functional and performance requirements of SoS. To enable an automatic business objectives conflict detection of the SoS architectures requirements, we use the model logic to formalize and reason the SoS architecture requirements.

References
[1] Khadka R, Sapkota B. Model-driven approach to enterprise interoperability at the technical service level [J]. Computers in industry, 2013, 64 (8):951-965.
[2] Liao S, Sun B. A knowledge-based architecture for planning military intelligence, surveillance, and reconnaissance [J]. Space Policy, 2003, 19 (3):191-202.
[3] US Department of Defense. DoD Architecture Framework Version 2.0, Volume 1: Introduction, Overview, and Concepts, Manager’s Guide[R]. 2009.
[4] MOD Partner. MOD Architecture Framework Version 1.2.004 [ EB/OL ] . http://www.mod.uk/modaf, 2010.
[5] IEEE Std 610.12, 1990. IEEE Standard Glossary of Software Engineering Terminology. Standard Coordinating Committee of the Computer Society [S], IEEE, 1990.
[6] Yang L, Yu M. Autonomous process collaboration framework for C4ISR system interoperaion[J]. Journal of System Engineering and Electronics, 2017, 28(6):1098-1113.
[7] Zhang T, Liu X. Capability-oriented Architectural Analysis Method Based on Fuzzy Description Logic[J]. Journal Computer Science and Information Systems, 2016, 13(1):65-74.
[8] Lamsweerde V, Darimont R, Letier E. Managing conflict in goal-driven requirements engineering [J]. IEEE Transactions on Software Engineering, 1998, 24(11): 908-926.
[9] Lamsweerde V, Letier E. Handling obstacles in goal-oriented requirements engineering [J]. IEEE Transactions on Software Engineering, 2000, 26(10):978-1005.
[10] Giancarlo G, Luís F. An ontology-based approach for evaluating the domain appropriateness and comprehensibility appropriateness of modeling languages [J]. Lecture Notes in Computer Science, Model Driven Engineering Languages and Systems, 2005 (3713): 691-705.
[11] Chellas F. Modal Logic[M]. Cambridge University Press, 1990.
[12] Qi Y, Wang Z. Modeling and verifying SoS performance requirements of C4ISR systems [J].
     Journal of System Engineering and Electronics, 2015, 26(4):754-763.