A Service-Oriented Method for System-of-Systems Requirements Analysis and Architecture Design

Ying ZHANG
Institute of Command Automation, PLA University of Science & Technology, Nanjing, China
Email: zhywl66@163.com

Xiaoming LIU, Zhixue WANG, Li CHEN
Institute of Command Automation, PLA University of Science & Technology, Nanjing, China

***Abstract*—Nowadays there are various problems, including the rapid growth of complexity, continually changing requirements and the alignment of IT with business strategy, challenge traditional design principles in analyzing large-scale complex systems, while Service-Oriented Computing offers a flexible integration architecture to meet the dynamic requirements of system-of-system. This paper introduce a service-oriented modeling method for System-of-Systems requirements analysis based on a three-tier framework with multi-ontologies. The method proposed in this paper provides a reusable and flexible solution for System-of-Systems requirements modeling and enable the high layer requirements description of System-of-Systems mapped to service-oriented system design architecture directly. A case study is given to demonstrate the applicability of the method.**

***Index Terms*—System-of-Systems, service-oriented modeling, requirements analysis, ontology**

I. INTRODUCTION

System-of-Systems (SoS) is defined as an interoperating collection of component systems that produce results unachievable by the individual systems alone[1]. From its earlier application in military, SoS development have extended to many and various domain, such as transportation, healthcare, internet, search and rescue, space exploration, and so on[2]. Analyzing the requirements of SoS is a daunting task, usually involving a complicated process of enterprise architecture development using some standard architecture frameworks such as Generalized Enterprise Reference Architecture and Methodology (GERAM)[3], Treasury Enterprise Architecture Framework (TEAF)[4], Zachman Framework[5], The Open Group Architectural Framework (TOGAF)[6], DoD Architecture Framework (DoDAF)[7], etc. However, the rapid growth of complexity accompanied with the continually changing requirements makes traditional design principles challenging. The challenge would come from following aspects.

Firstly, the architecture framework usually provides a number of viewpoints to model the architecture, but the viewpoints focus on what should be described rather than the concrete modeling method[8].

Secondly, it may be difficult to keep the low level IT requirements consist with the high level business concepts, which is a key in maintaining business value[9].

Thirdly, existing enterprise architecture proposals are represented in quite different ways, which results in great difficulty in interoperating between different models with different approaches and verifying the architecture as a whole[10].

In this paper, we propose a reusable method of service-oriented and ontology-based requirements modeling for developing SoS architecture within a three-tiers framework. The method can map the strategic level requirements down to the IT implementation by regarding service as basic granularity. It also provides rigorous modeling semantics through defining multiple ontology, which is important for domain knowledge reuse. Compared with the current methods, it provides a more efficient and more flexible way of requirements analysis for the large-scale complex systems.

The rest of this article is organized as follows: Section II introduces background and related works. Section III describes the three-tier modeling framework and multiple ontologies definitions. Section IV proposes the modeling process in detail. Section V gives an instance to demonstrate the applicability of our method. Section VI draws some conclusions and future work.

II. BACKGROUND AND MOTIVATION

The SoS discussed in this paper mainly refers to the C4ISR (command, control, communication, computers, intelligence, surveillance, and reconnaissance) systems[11-12], a typical military SoS. In order to ensure interchangeability and interoperability between systems, analyzing the requirements of C4ISR system usually use architecture frameworks such as the DoDAF[7] and the MODAF[13]. Recently, Capability-Based Planning (CBP)[14] and Capability Engineering[15-16] are proposed for C4ISR requirement acquisition, analysis and project management. And a series of files and standards

© 2012 ACADEMY PUBLISHER
doi:10.4304/jsw.7.2.358-365

are published, such as JCIDS (Joint Capabilities, Integration and Development System)[17], etc.

Service-oriented Computing (SOC) and Service-oriented Architecture (SOA)[18-20] with characteristic of loose coupled, dynamic binding and independent of development technologies, platforms and organizations, offers a new paradigm for C4ISR capability requirement analysis. The U.S. Department of Defense issued DoDAF 2.0[7] in 2009. Compared with the previous version, DoDAF 2.0 extends the previous three viewpoints of the architecture to more specific viewpoints, which including add the SvcV (Services Viewpoint). Wu et al.[21] proposed a reuse method of service-oriented modeling. Duncan et al.[22] researched how to support military capability by SOA, especially focused on the assessment framework of military capability, and measurement and monitoring of QoS (Quality of Service).

However, how to map the higher complex requirements to service-oriented system design is still a challenge for service-oriented C4ISR system integration. A capability in C4ISR is defined as "The ability to achieve a desired effect under specified (performance) standards and conditions through combinations of ways and means (activities and resources) to perform a set of activities."[7]. C4ISR capability requirement analysis belongs to a cross analysis of both business requirement and system requirement. The capability requirements describe the expected functions and effectiveness of C4ISR system performance in the user's viewpoint. It reflects business architecture requirements of C4ISR system. On the other hand, service-oriented system design focuses on the IT implementation and technology architecture. Therefore, there might be a gap between the business expectation and its IT implementation, and our research is aiming at bridging the gap.

The underlying principle of our method is presented in Fig. 1, a conceptual model of service-oriented C4ISR capability requirement analysis, which improves our previous work[23]. All capabilities are provided by activities in order to achieve military mission which may be divided into several tasks. Activity is performed by invoking and integrating a range of services which use various resources. The resources may be person, materiel, data or information, as well as process, product, technology and infrastructure, etc. All kinds of resources are encapsulated as services and distributed all over the network. Service, introduced as an important concept, forms an intermediate level which separates the arrangement of resources from capability deployment. This conceptual model, applied for analyzing C4ISR capability requirements and designing the systems, will not only support high-level requirements mapping to system architecture design by modeling the services but also enhance, through dynamic service composition, system flexibility and adaptability to various changes, no matter resulted from requirements, implement technology, or integration environment.

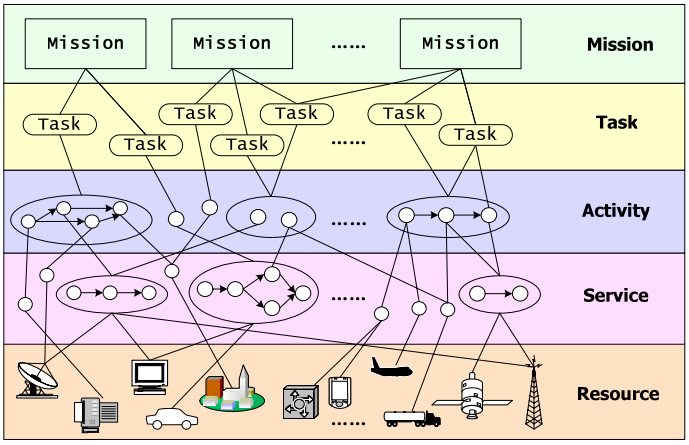

Figure 1. The conceptual model of service-oriented C4ISR capability analysis and integration

III. Hierarchy Modeling Framework And Ontologies Definition

A. Three-tier modeling framework

Service-oriented analysis and design emphasis on integrating and reusing services to implement systems. In this paper, we introduce a service-oriented modeling method for SoS with a three-layer framework based on multi-ontologies as shown in Fig. 2. There are three reasons of defining multi-ontologies in this paper. The first reason is to provide a uniform semantic for service-oriented analysis and modeling, in order to avoid the semantic interoperability problems[24]. The second reason is to increase modeling efficiency by domain knowledge reuse. The last one is to support model checking by logical reasoning.

Meta ontology is belonged to the top level, meta concept level, which including the fundamental concepts, relations and rules of SoS and guiding the whole modeling process.

Domain ontology, which as a projection of meta ontology in an specific domain is belonged to the second layer, i.e. domain level. In this level, domain experts capture domain specific concepts and relations which have explicit meanings and widely used in the domain, and prescribe the guidelines or rules for modeling the application requirements.

© 2012 ACADEMY PUBLISHER

The bottom layer is the application layer. In this layer, application ontology describes concepts, relations and rules of an application by inheriting and extending meta ontology or domain ontology. Activity plays an important role in this level, because it expands the activity concept of application ontology as well as is involved in matchmaking with service. The final model and specification of SoS requirements is established in this layer.

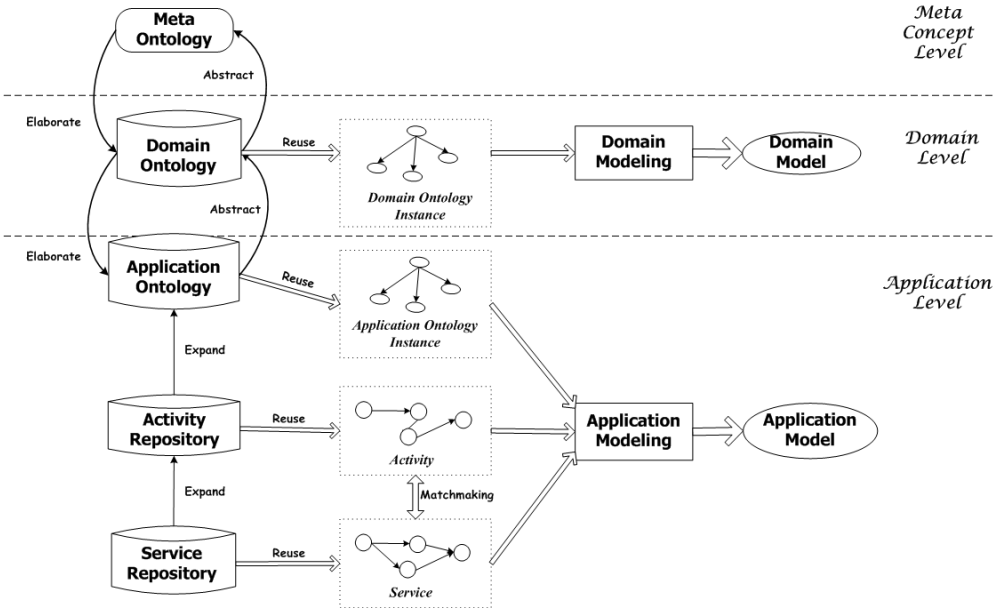

Figure 2. The hierarchy modeling framework for service-oriented SoS

B. Multiple ontologies definition

Definition 1 Meta Ontology

Meta Ontology = <MetaConcept, MetaRelation, MetaRule >

where:

$MetaConcept=\{mc_1,mc_2,...,mc_n\}$, mc_i, $(1 \le i \le n)$ is a meta concept.

$MetaRelation=\{MR_1,MR_2,...,MR_m\}$, MR_j, $(1 \le j \le m)$ is a meta relation. For given $mc_1 \in MetaConcept$, $mc_2 \in MetaConcept$ and $MR \in MetaRelation$, expression $MR(mc_1, mc_2)$ describes there is an association MR between concepts mc_1 and mc_2.

$MetaRule = \{MU_1, MU_2, ..., MU_t\}$, MR_k, $(1 \le k \le t)$ is a meta rule that defines the constraint necessarily held by all concepts and relations in the models.

The meta ontology defines the essential concepts, relations and rules, but are still too abstract to provide domain information. Thus we define domain ontology to capture domain specific concepts and relations as follows.

Definition 2 Domain Ontology

Domain Ontology =< DomConcept, DomRelation, DomRule, DomRef >

where:

$DomConcept = \{dc_1, dc_2, ..., dc_n\}$, $dc_i, (1 \le i \le n)$ is a domain specific concept.

$DomRelation = \{DR_1, DR_2, ..., DR_m\}$, $DR_j, (1 \le j \le m)$ is a domain specific relation. For given $dc_1 \in DomConcept$, $dc_2 \in DomConcept$ and $DR \in DomRelation$, $DR(dc_1, dc_2)$ expresses there is an association DR between domain concepts dc_1 and dc_2.

$DomRule = \{DU_1, DU_2, ..., DU_t\}$, $DU_k, (1 \le k \le t)$ is a domain specific rule that specifies necessary constraint of all concepts and relations in domain models. It keeps consistent with the rules in *MetaRule*.

$DomRef \subseteq (DomConcept \times MetaConcept) \cup (DomRelation \times MetaRelation)$ is a mapping function. For given $dc \in DomConcept$ and $mc \in MetaConcept$, the mapping from dc to mc can be described as $dref(dc) = mc$, $dref \in DomRef$, which means the meta concept of dc is mc. Similarly, *DomRef* also can specify the mapping from *DomRelation* to *MetaRelation*.

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

Definition 3 Application Ontology

Application Ontology = < AppConcept, AppRelation, AppRule, AppRef >

where:

$AppConcept = \{ac_1, ac_2, ..., ac_n\}$, $ac_i, (1 \le i \le n)$ is an application specific concept in a domain;

$AppRelation = \{AR_1, AR_2, ..., AR_m\}$, $AR_j, (1 \le j \le m)$ is an application specific relation in a domain. For give $ac_1 \in AppConcept$, $ac_2 \in AppConcept$, $AR \in AppRelation$, $AR(ac_1, ac_2)$ expresses there is an

© 2012 ACADEMY PUBLISHER

association *AR* between application concepts ac_1 and ac_2;

$AppRule = \{AU_1, AU_2, ..., AU_t\}$, $AU_k, (1 \le k \le t)$ is an application specific rule that constraints application concepts and relations according to the rules of *DomRule* or *MetaRule*.

$AppRef \subseteq (AppConcept \times (DomConcept \cup MetaConcept)) \cup (AppRelation \times (DomRelation \cup MetaRelation))$, is a mapping function. For given $ac \in AppConcept$, $dmc \in DomConcept \cup MetaConcept$, the mapping from ac to dmc can be described as $aref(ac) = dmc$, $dref \in AppRef$, which means the meta concept of ac is dmc. Similarly, *AppRef* also can specify the mapping from *AppRelation* to $DomRelation \cup MetaRelation$.

Activity is important role in application level. We formally define activity as follows.

Definition 4 Formal Definition of Activity

Activity = < *ActivityName, ActivityProcess, ActivityRelation, ActivityFunction, ActivityRef*>

where:

ActivityName is the name of an activity.

$ActivityProcess = AtomActivity \cup ActStructure$ represents the typical implementation process of an activity. $AtomActivity = \{ata_1, ata_2, ..., ata_n\}$, in which $ata_i, (1 \le i \le n)$ is an atomic activity that is the basic unit of activity process;

$ActStructure = \{sequence, split, split + join, Unordered, Choice, If - Then - Else, Iterate, Repeat - Until\}$ is the control structure set represents the relationships between atomic activities

$ActivityRelation = \{HasProcess, ReqSerOf\}$, in which *HasProcess* is used to represent the relation between an activity and its implementation process, while *ReqSerOf* is represented the relation between an atomic activity and services contribute to realize it.

$ActivityFunction = Input \cup Output \cup Precondition \cup Effect \cup Constraint$ is functional description of an activity. *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. *Constraint* is a finite set of activity attributes that is usually used to specify some perform demand of an activity, such as the perform time, etc.

AppRef is a mapping function, specifying the mapping from *ActivityName* to a unique *activity* concept of *AppConcept* in application ontology. Introduction of mapping function can matching the activity concept between *activity* and *application ontology*.

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

Definition 5 Formal Definition of Service

Service = < *ServiceName, ServiceFunction, ServiceRef* >

where:

ServiceName is the name of a service.

$ServiceFunction = Input \cup Output \cup Precondition \cup Effect \cup Constraint$ is functional description of a 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. *Constraint* is a finite set of service attributes, generally used to describe the QoS properties of service. For C4ISR system, the main considerations of QoS are reliability, availability, safety, accuracy and so on.

ServiceRef is a mapping function, specifying the mapping from *ServiceName* to *ActivityName* or *activity* concept of *AppConcept* in application ontology, or from *ServiceFuntion* to *ActivityFunction.* Introduction of this mapping function can match the services with its name and function between *service* and *activity*.

A simple example of *SchemeCreating* service is shown in Fig. 3.

```
<Service>
  <ServiceName> SchemeCreating </ServiceName>
  <ServiceFunction>
    <Input>Type of Vehicle</Input>
    <Input>Dispatch Destination</Input>
    <Input> Dispatch Number </Input>
    <Output>Dispatch Scheme</Output>
    <Precondition>Receive Dispatching Command</Precondition>
    <Effect>Creating Accurate Dispatch Scheme </Effect>
    <Constraint>
      <SafetyGrade>Secret Grade</SafetyGrade>
    </Constraint>
  </ServiceFunction>
</Service>
```

Figure 3. The *SchemeCreating* service

IV. SERVICE-ORIENTED MODELING PROCESS

The reusable method for service-oriented SoS modeling consists of four phases. The whole process involves four repositories constituted the domain knowledge for modeling, including domain ontology repository, application ontology repository, activity repository and service repository as shown in Fig. 2.

Phase 1: capture domain knowledge, i.e. domain conceptual modeling. If the domain ontology repository has been established, modeler could reuse the existing domain ontology instance for domain modeling. Otherwise, domain experts and modelers should work together to construct domain model. The way is defining domain concepts, relations and rules by specializing meta ontology according to specific domain features, and specifying the corresponding relation between domain concepts/relations/rules and meta concepts/relations/rules. Domain ontology defines all concepts and relations which are general and reusable for a specific domain. The new domain model should be restored in domain ontology repository after modeled in order to future reuse.

Phase 2: construct the initial application model. The first step is looking up and retrieving eligible application

© 2012 ACADEMY PUBLISHER

model in the application ontology repository. If there is no reusable application model in current repository, it is need to define application ontology and establish application model under the guide and constraint of domain ontology. Application ontology is a projection of domain ontology in a concrete application which describes concepts and relationships for specific application in domain. It is the static reusable asset of domain knowledge. Therefore, the application model should be stored in the repository after built. In the future modeling process, modeler should find whether there is available application model in domain knowledge repository or not at first, which could improve the modeling efficiency by reuse.

Phase 3: for each activity established in the initial application model, analyze and model activity process. Activities are divided into two kinds: simple activity and complex activity. The former is performed by an atomic activity while the latter is performed by several atomic activities. The activity repository restored both of the two kinds of activities. For each activity, if the activity is included in the activity repository, modeler could reuse the activity process directly. Otherwise, the modeler could modify the most similar existing activity process according to specific application requirements then reuse, or model a new activity with new relations, processes, attributes, rules of the activity to supplement and improve the repository.

Phase 4: create the final application model by matchmaking the atomic activities and services. An activity is implemented by activity process which consists of atomic activities. And an atomic activity is realized by service which could be one service or services composition. Because the concepts and relations of *activity* and *service* are all formally descript in the same domain knowledge, we could regard atomic activity as service requester. That means the *ActivityFunction* of atomic activity could be regarded as service request description, and the *ServiceFunction* of service could be regarded as service provider advertisement. For each atomic activity, use the **ALGORITHM 1** (as shown in Fig. 4) to find the matched service set which could implement it. If the matchmaking is fail, then clue the modeler either modify the inputs and outputs then matching one more time or define a new service. Finally, the final application model is constructed by supplement the initial application model with the matched services.

ALGORITHM 1 the matchmaking algorithm between atomic activity and service
INPUT: the atomic activity set *APSET* which is obtained by previous phase. Service repository *SS*, time limit T_{out}
OUTPUT: the atomic activity implementation set *MatchServiceListSet* which is consist of services
INITIALIZATION: $MatchServiceListSet = NULL$;
ALGORITHM STEPS:

Step1: Select the first atomic activity in *APSet* as the current activity, obtain its output-set $O_{act} = \{O_1, O_2, \cdots, O_n\}$ and input-set $I_{act} = \{I_1, I_2, \cdots, I_m\}$.

Step2: $O_{new} = O_{act}$, $I_{new} = I_{act}$, $SStemp = SS$, $ServiceList.Ser = NULL$, $ServiceList.Ctr = NULL$.

Step3: For each service Ser_j in $SStemp$, compute the intersection set $O_{iset_j} = O_{new} \cap O_{ser_j}$ and $|O_{iset_j}|$, where $O_{ser_j} = \{O_1, O_2, \cdots, O_l\}$ is the output-set of Ser_j . The result may be four mutually exclusive cases, as follow:

① For each O_{iset_j} is satisfied $O_{iset_j} = \varnothing$, which means there is at least one element (output) of O_{new} can not be provided by services in current service repository, matching failed, $ServiceList.Ser = NULL$, $ServiceList.Ctr = NULL$, do Step5.

② There is at least one O_{iset_j} satisfying $O_{iset_j} \supseteq O_{new}$. This means there is at least one service could match all output concepts needed by O_{new} directly. Add Ser_j to $ServiceList.Ser$, $ServiceList.Ctr = Choice$, do Step4.

③ This case is means that although there is no single service could provide all output concepts in current service repository, there are some services can provide the concepts corporately, i.e. the service composition could provide all output concepts requested by O_{new} . Therefore, choice the Ser_j which has the biggest $|O_{iset_j}|$, add Ser_j to $ServiceList.Ser$, and let $ServiceList.Ctr = Split \ \& \ Join$, $O_{new} = O_{new} - O_{iset_j}$, $SStemp = SStemp - Ser_j$, do Step3.

④ The searching time is out of time limit, matching failed, $ServiceList.Ser = NULL$, $ServiceList.Ctr = NULL$, do Step5.

Step4: For each service Ser_j in $ServiceList.Ser$, compute $I_{dset_j} = I_{new} - I_{ser_j}$, where $I_{ser_j} = \{I_1, I_2, \cdots, I_l\}$ is the input-set of Ser_j . And the result is divided into two situations:

① For each Ser_j , the I_{dset_j} is satisfied $I_{dset_j} = \varnothing$, which means that all input concepts of Ser_j have already included by service request, matching is successful, do Step5.

② If Ser_j has $I_{dset_j} \neq \varnothing$, which means there are at least one input concept of Ser_j could not be provided by current service request. Therefore, it is need to find the predecessor services of current service. For each Ser_j , let $O_{new} = I_{dset_j}$, $SStemp = SStemp - Ser_j$, $ServiceList.Ctr = Sequence$, do Step3 again.

Step5: Add $ServiceList$ to $MatchServiceListSet$, complete the process of current activity. Select the next activity in *APSet* as current activity, while get its input and output concept sets as O_{act} and I_{act} , do Step2, until process all atomic activities in *APSet* .

Step6: Return $MatchServiceListSet$

Figure 4. The matchmaking algorithm between atomic activity and service

All concepts used during the whole modeling process should be included in domain knowledge. If there is necessary concept has not been defined, it is need to add the concept into domain knowledge. That means all newly-built domain model and application model including activity process and service should be added and stored in repositories to improve and enrich the domain knowledge for modeling.

In addition, because the modeling process involves multi-tiers and multi-models, the correctness and

© 2012 ACADEMY PUBLISHER

consistency between models can be ensured by model verification. An effective method is transforming the models to OWL DL[25] ontology, then accomplish the model verification of correctness and consistency by ontology reasoning technology based on Description Logics (DLs)[25-26].

V. CASE STUDY

In this section, we introduce a case study of C4ISR system to illustrate the applicable and available of the method proposed in this paper. The instance is belonged to military armament supply domain, and the specific application is dispatch vehicle LAV-I.

According to the Meta-Model Data Groups in DoDAF 2.0[7], the meta ontology of C4ISR capability analysis is shown in Fig. 5. Meta concepts define all basic concepts for C4ISR capability analysis such as *Capability*, *Activity*, *Performer*, *Resource*, etc. Meta relations define all associations between meta concept, such as *RuleConstrainsActivity*, *ActivityPerformedByPerformer*, etc. Both meta concepts and meta relation keep the same semantics as those in DoDAF 2.0[7].

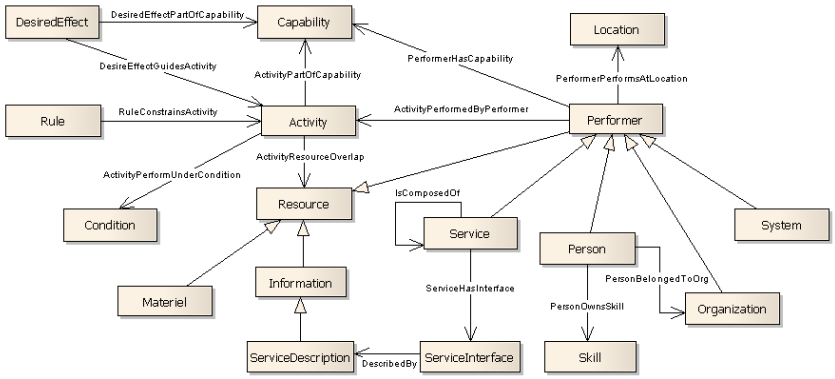

Figure 5. An example of C4ISR capability meta ontology

Firstly, we define domain ontology of military logistics by inheriting and extending meta ontology according to domain requirements and features. The logistics domain model (fragment) is shown in Fig. 6, we use UML stereotype to describe the mapping between meta ontology and domain ontology. For example, domain concept *VehicleSupply* is specialized from meta concept *Capability*, and domain concept *DispatchVehicle* is specialized from meta concept *Activity*, etc.

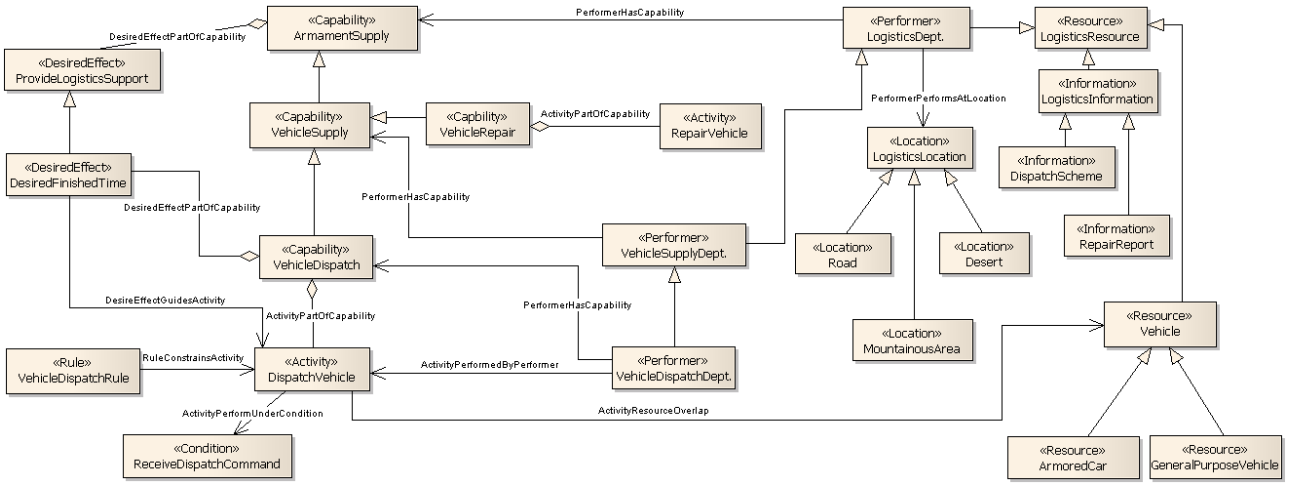

Figure 6. The vehicle supply domain model (fragment)

Secondly, we suppose that there is no reusable instance in application ontology repository currently. Therefore, it is need to define application ontology in application level by instantiating domain ontology, and establishing the initial application model. As shown in Fig. 7, application concept *DispatchLAV-I* is an instance of domain concept *DispatchVehicle*, and application concept *LAVDispatchCenter* is an instance of domain concept *VehicleDispatchDept.*, etc. The initial application model should be added to application ontology repository which could be reuse in future.

© 2012 ACADEMY PUBLISHER

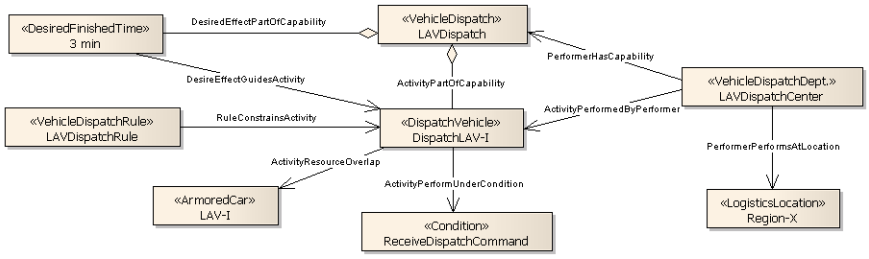

Figure 7. The initial application model of dispatch LAV-I

The next phase is analyzing and modeling each activity established in the initial application model with its implement process. We suppose that the activity process named *DispatchProcess* has stored in activity repository and reuse it directly. Then matchmaking each atomic activity of *DispatchProcess* with services by using the **ALGORITHM 1** in the service repository. Finally, composite all the services into the initial application model, and obtain the final application model with service as the basic granularity, as shown in Fig. 8. Both the *RepairProcess* and services *CommandReceiving*, *SchemeCreating*, *SchemeSending* may also be used in modeling other application such as dispatch MBT, dispatch APC and so on.

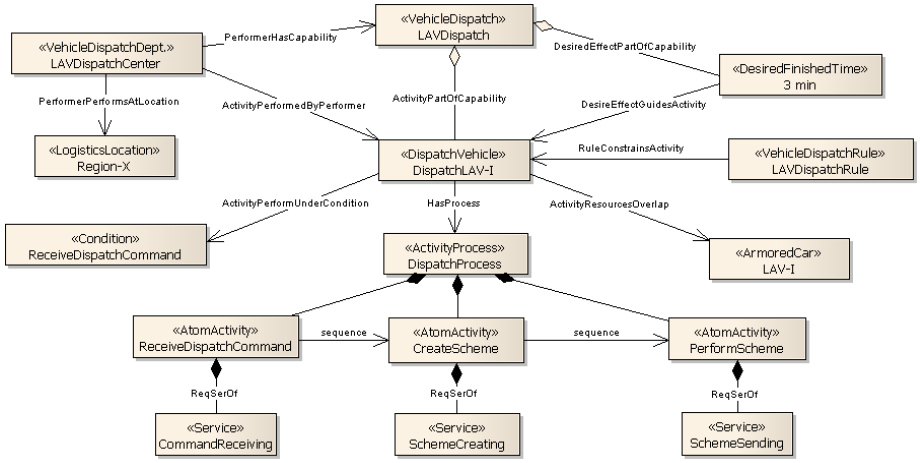

Figure 8. The final application model of dispatch LAV-I

After these phases, the higher complex C4ISR requirements had been mapped to the application model with service as basic granularity. The final application model provides a blueprint for service-oriented C4ISR system integration. The successional phase could assemble and deploy services by using the existing SOA tools according to the blueprint.

VI. Conclusions And Future Works

This paper studies how to apply SOC with its reusable, flexible, loose couple features to SoS requirement analysis such as C4ISR system. Firstly, a service-oriented conceptual model for C4ISR is presented. Based on the conceptual model, a novelty modeling method with three layer and multiple ontologies is proposed for service-oriented C4ISR capability analysis. The features represent of our work is summarized as follows:

(1) This method drives service modeling by activity analysis, and provides C4ISR capability by invoked and integrated services. It resolve the alignment of C4ISR requirements analysis and system design by taking service as the basic granularity, which will enable the high layer requirements description to be mapped to system design architecture.

(2) This method is a hybrid modeling method. On the one hand, this method could increase the modeling efficiency by reuse domain knowledge, i.e. the assets restored in the repositories. On the other hand, it also supports dynamic modeling by service matchmaking and composition.

(3) This method is flexible because it focuses on the requirements modeling of problem domain rather than the technical details of application implementations. Therefore, if the higher requirements change, only the abstract solution specification needs to be updated to reflect the new requirements, modeler can reconstruction the application model through invoking and integrating services flexibly and fast. Correspondingly, if the implementation technology changes, it should be possible to reuse the same abstract solution specification defined by the domain experts. This will reduce the cost and decrease the time to implement a change. This method also supports updating services that the new and better services could replace the old ones without influencing the high-level model.

© 2012 ACADEMY PUBLISHER

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

ACKNOWLEDMENT

This research is supported by the National High-Tech Research and Development Program of China under the project number 2007AA01Z126. The authors are grateful for the anonymous reviewers who made constructive comments.

REFERENCES

[1] Krygiel, *Behind the Wizard's Curtain*, CCRP Publication Series, July, 1999, pp. 33.
[2] WANG Yuan-fang, ZHOU Hong-ren, JING Zhong-liang, "System-of-Systems, an Overview", *Journal of System Simulation*, 19(6), pp. 1182-1185, 2007 (in Chinese with English abstract).
[3] IFAC-IFIP Task Force on Architecture for Enterprise Integration, *GERAM: Generalized Enterprise Reference Architecture and Methodology* (Version 1.6.3). 1999.
[4] Department of the Treasury CIO Council, *Treasury Enterprise Architecture Framework* (Version l), 2000.
[5] Zachman J, *The Zachman framework: a primer for enterprise engineering and manufacturing*, 2003.
[6] Open Group, *TOGAF: The Open Group Architecture Framework* (Version 9), Document Number: G091, 2009.
[7] US Department of Defense, *DoD Architecture Framework* (Version 2.0), 2009.
[8] LUO Aimin, HUANG Li, LUO Xueshan, "Research on activity-centric architecture methodology", *Systems Engineering and Electronics*, 30(3), pp. 499-502, 2008 (in Chinese with English abstract).
[9] Cuenca L., Ortiz A., Boza A, "Business and IS/IT Strategic Alignment Framework", In: Camarinha-Matos L., Pereira P., Ribeiro L. (Eds.), *Emerging Trends in Technological Innovation, IFIP Advances in Information and Communication Technology*, vol. 314, pp. 24-31, 2010.
[10] Chen D., Doumeingts G., Vernadat F, "Architectures for enterprise integration and interoperability: Past, present and future", *Computers in Industry*, 59(7), pp. 647-659, 2008.
[11] Schorling, S., Rine, D, "A methodology for designing toolkits for specification level verification of interval-contained information systems requirements", *Information and Software Technology*, 44 (2), pp. 77-90, 2002.
[12] Liao, S., Sun, B., Wang, R. "A knowledge-based architecture for planning military intelligence, surveillance, and reconnaissance", *Space Policy*, 19(3), pp. 191-202, 2003.
[13] UK Ministry Of Defence. *MOD Architecture Framework* (Version 1.2.004), 2010.
[14] SUN Yan, DAI Hao, "Brief Introduction of Military Requirement Method Based on Capability", *Science Technology And Engineering*, 9(7):2170-2176, 2007 in Chinese with English abstract).
[15] Lam S, Lemieux F, Lalancette C, et al. "Toward a Capability Engineering Process", *Report No. ADA432358*, Canada Valcartier(Quebec): Defence Research and Development, 2004.
[16] Lam S, Mokhtari M, Lizotte M, et al. "CapDEM - Toward a Capability Engineering Process: A Discussion Paper", *Report No. ADA440003*, Canada Valcartier(Quebec): Defence Research and Development, 2005.
[17] Joint Chief of CHIEFOFSTAFF. CJCSI 3170.01F Joint Capabilities, *Integration and Development System*, 2007.
[18] Arsanjani A, "Service-Oriented Modeling and Architecture", *IBM Technical Report*, 2004.
[19] Erl Thomas, Service-Oriented *Architecture (SOA):Concepts, Technology, and Design*, Pearson Education, Inc, 2005.
[20] Huhns Michael N, *Service-oriented computing semantics, processes, agents*, Chichester : John Wiley & Sons, Ltd, 2006.
[21] Budan Wu, Zhi Jin, "Service-Oriented Modeling: An Extensive Reuse Method", *Proc. of Annual IEEE International Computer Software and Applications Conference*, 2008.
[22] Duncan Russell, Nik Looker, Lu Liu , Jie Xu, "Service-Oriented Integration of Systems for Military Capability", *Proc. of 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing (ISORC)*, 2008.
[23] ZHANG Ying, WANG Zhixue, LIU Xiaoming, et al. "C4ISR Capability Analysis based on Service-Oriented Architecture", *Proc. of 5th IEEE international symposium on Service-oriented system engineering (SOSE)*, 2010.
[24] Stanislav Vassilev Pokraev, *Model-driven Semantic Integration of Service-Oriented Applications*, Enschede, The Netherlands, 2009.
[25] Franz Baader, Diego Calvanese, Deborah L. Mcguinness, et al. *The Description Handbook*. Cambridge University Press, 2003.
[26] DONG Qingchao, WANG Zhixue, ZHU Wei-xing, etal. "Domain-specific language for C4ISR capability analysis", *Systems Engineering—Theory&Practice*. 31(3), pp. 552-560, 2011 (in Chinese with English abstract).

Ying ZHANG was born in 1982. She is a candidate for doctor in Institute of Command Automation, PLA University of Science & Technology. Her main research is software engineering, requirement engineering, SOA; Email: zhywl66@163.com

Xiaoming LIU was born in in 1956. He is a professor for doctor in Institute of Command Automation, PLA University of Science & Technology. His main research is system engineering, computer simulation.

Zhixue WANG was born in 1961. He is a professor for doctor in Institute of Command Automation, PLA University of Science & Technology. His main research is requirement engineering, system engineering, SOA.

Li CHEN was born in 1982. He received the Bachelor of Engineering and Master of Engineering degrees from Institute of Meteorology, PLA University of Science and Technology in 2005 and 2008, respectively. He began his PhD study in Institute of Command Automation, PLA University of Science and Technology since 2008. His research interests include semantic Web, data mining, and Web services.

© 2012 ACADEMY PUBLISHER