

# Self-Adaptive System Architecture Modeling Based on DoDAF

Guanyue Gao and Feng Liu

**Abstract**—Now the system can be called as system of system. The system consists of several sub systems. At the same time, the system has ability that can change their behavior according to the operational environment or their running state. The DoDAF is an architecture framework for United States Department of Defense. It is a challenge for extending the DoDAF and design the self-adaptive system architecture. This paper focuses on the self-adaptive theory and self-adaptive system architecture modeling.

**Index Terms**—Self-adaptive, modeling, system architecture, operational environment

## I. INTRODUCTION

Now, more and more system is constituted by many small systems. These systems communicate each other by network. The department of defense information enterprise system, intelligent transportation systems and large software systems are some of networked systems that we are observing in governments and commercial enterprises. Inevitably we are increasing dependent on these networks. And these systems do not only depend on the network, but also depend on the environment that they are staying. At the same time, these systems are running for a long time. Maybe environment is changed, but we hope, our system is also running and not crush. These systems have an ability the can adjust themselves and adapt to the environment.

We can call these systems as self-adaptive system. The self-adaptive system can modify its own behavior in response to changes in its operating environment or accept the new sub-system because the old sub-system is replaced by the new. When the old system is replaced by the new, we need confirm that the system can adapt to the new, and do not be disrupted.

When system's operating environment changes, the system must adapt itself to complete mission. For example, A fleet of unmanned air vehicles undertakes a mission to disable an enemy airfield. Pre-mission intelligence indicates that the airfield is not defended, and mission planning proceeds accordingly. While the UAVs are en route to the target, new intelligence indicates that a mobile surface-to-air missile launcher now guards the airfield. The UAVs autonomously re-plan their mission, dividing into two

groups—a SAM-suppression unit and an airfield-suppression unit—and proceed to accomplish their objectives. During the flight, specialized algorithms for detecting and recognizing SAM launchers automatically upload and are integrated into the SAM-suppression unit's software [1].

We can model the system by the tool System Architecture based on the DoDAF (the Department of Defense Architecture Framework). DoDAF is an architecture framework for the United States Department of Defense, which provides structure for a specific stakeholder concern through viewpoints organized by various views.

But for the self-adaptive system, the DoDAF doesn't define how we can describe the self-adaptive section and does not give a clear approach to model the self-adaptive system architecture. For the self-adaptive system, we can extend the DoDAF, and use the Model-Driven theory to support the self-adaptive system modeling. Section 2 discusses the DoDAF and system architecture modeling. Section 3 discusses the model-driven theory and how to extend the DoDAF to support the self-adaptive system modeling. Section 4 provides a small example to illustrate how to support the self-adaptive system modeling.

## II. DoDAF AND SYSTEM ARCHITECTURE MODELING

The Department of Defense Architecture Framework (DoDAF) provides a foundational framework for developing and representing architecture descriptions that ensure a common denominator for understanding, comparing, and integrating architectures across organizational, Joint, and multinational boundaries. It establishes data element definitions, rules, and relationships and a baseline set of products for consistent development of systems, integrated, or federated architectures. These architecture descriptions may include Families of Systems (FoS), Systems of Systems (SoS), and net-centric capabilities for interoperating and interacting in the NCE [2].

Note, where the diagram states TBD, the DoDAF V2.0 was promulgated on May 28, 2009. The first version of the development DoDAF was developed in the 1990s under the name C4ISR architectural Architecture Framework. C4ISR stand for The Command, Control, Communications, Computers, and Intelligence, Surveillance, and Reconnaissance. In the same period the reference model TAFIM, which was initiated in 1986, was further developed. The first C4ISR Architecture Framework v1.0, released 7 June 1996, was created in response to the passage of the Clinger-Cohen Act. It addressed the 1995 Deputy Secretary of Defense directive that a DoD-wide effort be undertaken to define and develop a better means and process for ensuring

Manuscript received September 20, 2012; revised December 16, 2012. This work was supported in part by the National Natural Science Foundation of China No.90718018, 863 subjects (2007AA010302).

Guanyue Gao is with the Beihang University, Beijing. He is now with the School of Computer Science and Engineering (e-mail: gaoguan Yue@gmail.com).

Feng Liu was with the Beihang University, Beijing (e-mail: StevenLiu100@163.com).

that C4ISR capabilities were interoperable and met the needs of the warfighter. Continued development effort resulted in December 1997 in the second version, C4ISR Architecture Framework v2.0.

In August 2003 the DoDAF v1.0 was released, which restructured the C4ISR Framework v2.0 to offer guidance, product descriptions, and supplementary information in two volumes and a Desk Book. It broadened the applicability of architecture tenets and practices to all Mission Areas rather than just the C4ISR community. This document addressed usage, integrated architectures, DoD and Federal policies, value of architectures, architecture measures, DoD decision support processes, development techniques, analytical techniques, and the CADM v1.01, and moved towards a repository-based approach by placing emphasis on architecture data elements that comprise architecture products [3].

In February 2004 the documentation of Version 1.0 was released with volume "I: Definitions and Guidelines", "II: Product Descriptions" and a "Deskbook". In April 2007 the Version 1.5 was released with a documentation of "Definitions and Guidelines", "Product Descriptions" and "Architecture Data Description".

On May 28, 2009 DoDAF v2.0 was approved by the Department of Defense. DoDAF V2.0 is published on a public website.

Other derivative frameworks based on DoDAF include the NATO Architecture Framework (NAF) and Ministry of Defense (United Kingdom) Architecture Framework (MODAF). Like other EA approaches, for example The Open Group Architecture Framework (TOGAF), DoDAF is organized around a shared repository to hold work products. The repository is defined by the Core Architecture Data Model 2.0 (CADM -- essentially a common database schema) and the DoD Architecture Registry System (DARS). A key feature of DoDAF is interoperability, which is organized as a series of levels, called Levels of Information System Interoperability (LISI). The developing system must not only meet its internal data needs but also those of the operational framework into which it is set.

The DoDAF V1.5 defines a set of products, a view model, that act as mechanisms for visualizing, understanding, and assimilating the broad scope and complexities of an architecture description through graphic, tabular, or textual means. These products are organized under four views:

- All View (AV)
- Operational View (OV)
- System View (SV)
- Technical Standards View (TV)

Each view depicts certain perspectives of architecture as described below. Only a subset of the full DoDAF view set is usually created for each system development. The Fig represents the information that links the operational view, systems and services view, and technical standards view. The three views and their interrelationships – driven by common architecture data elements – provide the basis for deriving measures such as interoperability or performance, and for measuring the impact of the values of these metrics on operational mission and task effectiveness.

The representation of DoDAF products may be drawn

from many diagramming techniques, including IDEF, UML, SysML etc. But for the self-adaptive system modeling, we had better use the meta-model theory and extend the DoDAF framework to support the self-adaptive system modeling [3]

TABLE I: AV VIEW

Product	Description
AV-1	Overview and Summary Information
AV-2	Integrated Dictionary

TABLE II: OV VIEW

Product	Description
OV-1	High Level Operational Concept Graphic
OV-2	Operational Node Connectivity Description
OV-3	Operational Information Exchange Matrix
OV-4	Organizational Relationships Chart
OV-5	Operational Activity Model
OV-6a	Operational Rules Model
OV-6b	Operational State Transition Description
OV-6c	Operational Event-Trace Description
OV-7	Logical Data Model

TABLE III: SV VIEW

Product	Description
SV-1	Systems/Services Interface Description
SV-2	Systems/Services Communications Description
SV-3	Systems-Systems, Services-Systems, Services-Services Matrices
SV-4a/b	Systems/Services Functionality Description
SV-5a/b/c	Operational Activity to Systems Function, Operational Activity to Systems and Services Traceability Matrices
SV-6	Systems/Services Data Exchange Matrix
SV-7	Systems/Services Performance Parameters Matrix
SV-8	Systems/Services Evolution Description

TABLE IV: TV VIEW

Product	Description
TV-1	Technical Standards Profile
TV-2	Technical Standards Forecast

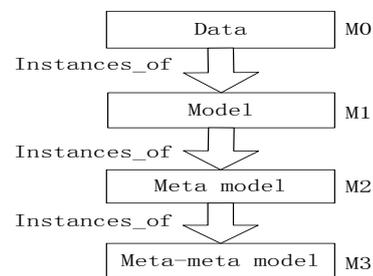


Fig. 1. Four-layer meta-model architecture.

### III. MODEL-DRIVEN AND META-MODEL DESIGN

Meta-modeling, or meta-modeling in software engineering and systems engineering among other disciplines, is the analysis, construction and development of the frames, rules, constraints, models and theories applicable and useful for

modeling a predefined class of problems. As its name implies, this concept applies the notions of meta and modeling [4].

Not only in software engineering, but also in system engineering, the use of models is more and more recommended. This should be contrasted with the classical code-based development techniques. A model always conforms to a unique meta-model. One of the currently most active branches of Model Driven Engineering is the approach named model-driven architecture proposed by OMG [4]. This approach is based on the utilization of a language to write meta-models called the Meta Object Facility or MOF. Typical meta-models proposed by OMG are UML, SysML, SPEM or CWM. ISO has also published the standard meta-model ISO/IEC 24744. All the languages presented in Fig. 1 could be defined as MOF meta-models [5].

The meta-meta model is the infrastructure for a meta-modeling architecture and defines the language for specifying meta-models. The meta-model is an instance of a meta-meta model. The model is an instance of a meta-model and defines a language to describe an information domain. The data is an instance of a model, and defines a specific information domain.

For support the self-adaptive system architecture modeling, we first design the meta-model according the DoDAF framework, and at the same time, need to add some views to describe the self-adaptive view. Second, we need to develop the modeling tools to modeling a system. At the last, we need to do an example to validate. But, there are a lot of products in each view. I just pick the two products in OV(OV-2 and OV-5) and give an approach how we can give the self-adaptive modeling.

OV-2:

Product Definition. The OV-2 graphically depicts the operational nodes (or organizations) with needlines between those nodes that indicate a need to exchange information. The graphic includes internal operational nodes (internal to the architecture) as well as external nodes.

Product Purpose: OV-2 is intended to track the need to exchange information from specific operational nodes (that play a key role in the architecture) to others. OV-2 does not depict the connectivity between the nodes.

Product Detailed Description. The main features of this product are the operational nodes and the needlines between them that indicate a need to exchange information. The product indicates the key players and the interactions necessary to conduct the corresponding operational activities of OV-5.

Operational Nodes. An operational node is an element of the operational architecture that produces, consumes, or processes information. What constitutes an operational node can vary among architectures, including, but not limited to, representing an operational/human role (e.g., Air Operations Commander), an organization (e.g., Office of the Secretary of Defense (OSD)) or organization type, i.e., a logical or functional grouping (e.g., Logistics Node, Intelligence Node), and so on. The operational node will also vary depending on the level of detail addressed by the architecture effort [1].

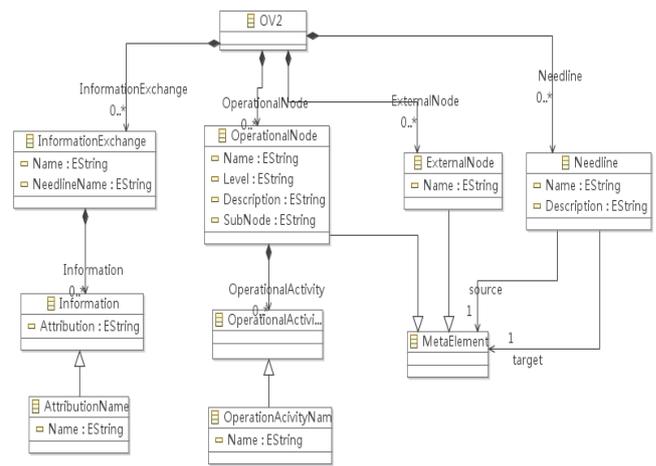


Fig. 2. OV-2 meta-model

TABLE V: OV-2 ELEMENT

Element name	Attribution	Description
OV2	root	Define the OV-2 view
OperationNode	Operational Node	Operational Node in OV2
OperationalActivity	Operation Activity	The attribution of operational node
ExternalNode	External node	The external node in OV2
Needline	Need line	Define the information exchange
InformationExchange	Information exchange	The information that needline need
MetaElement	Temporary element	Not be used in OV-2, just simplify the design of meta model

According to the description about OV2, OV-2 Meta Model is designed in Fig. 2, and the Table V gives the detail information about the ov2's elements:

OV-5:

Product Definition: The OV-5 describes the operations that are normally conducted in the course of achieving a mission or a business capability. It describes capabilities, operational activities (or tasks), input and output (I/O) flows between activities, and I/O flows to/from activities that are outside the scope of the architecture. High-level operational activities should trace to (and are decompositions of) a Business Area, an Internal Line of Business, and/or a Business Sub-Function as published in OMB's Business Reference Model [OMB, 2003].

Product Purpose. OV-5 can be used to:

- Clearly delineate lines of responsibility for activities when coupled with OV-2.
- Uncover unnecessary operational activity redundancy.
- Make decisions about streamlining, combining, or omitting activities.
- Define or flag issues, opportunities, or operational activities and their interactions (information flows among the activities) that need to be scrutinized further.
- Provide a necessary foundation for depicting activity sequencing and timing in OV-6a, OV-6b, and OV-6c.
- Identify critical mission threads and operational information exchanges by annotating which activities are critical (i.e., identify the activities in the model that are critical).

Product Detailed Description. OV-5 describes capabilities, operational activities (or tasks), I/O flows between activities, and I/O flows to/from activities that are outside the scope of the architecture.

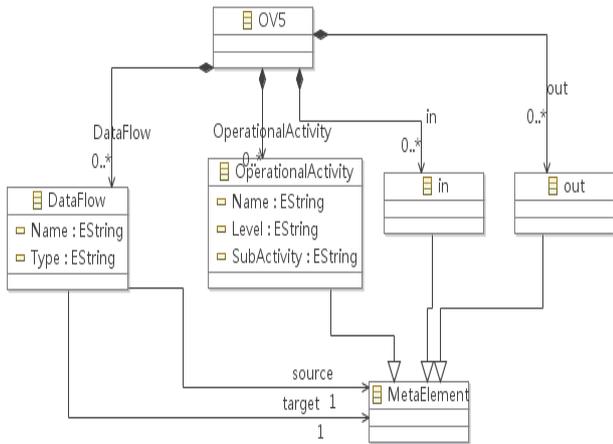


Fig. 3. OV-5 meta-model

TABLE VI: OV-6 ELEMENT

Element name	Attribution	Description
OV5	Root	Define the OV-5 view
OperationalActivity	Operation activity	Complete the operation mission activity
In	Input	External input
Out	Output	Output to external
DataFlow	Data flow	Information input/output
MetaElement	Temporary element	Not be used in OV-5. Just simplify the design of meta-model

I/Os of operational activities relate to information elements of OV-3 and are further characterized by the information exchange attributes described in OV-3. I/Os that are produced or consumed by leaf operational activities that cross operational node boundaries are carried by needlines of OV-2. Operational activities can be annotated (e.g., via the mechanism arrow in an IDEF0 diagram) with the corresponding operational node from OV-2.

According to the description about OV-5, The OV-5 meta-model is designed in Fig. 3 and the Table VI gives the detail information of OV-5's elements:

To describe the self-adaptive system, need to define the different state. Using the different state to represent the different scene, and represent system architecture response to the scene. And we must define a language to auto-generate next system architecture according to one architecture.

We can call the state view as OV-8:

Production Definition: The OV-8 describes the states that represent the different scene and include all views that is influenced by different scene. That says a view will be changed in different scene.

Product Purpose: OV-8 intends to track the different state about system architecture and to give a clear description of system architecture. OV-8 does not have the system element that is composed by system architecture.

According to the description about the OV8, the OV-8

meta-model is designed in Fig. 4:

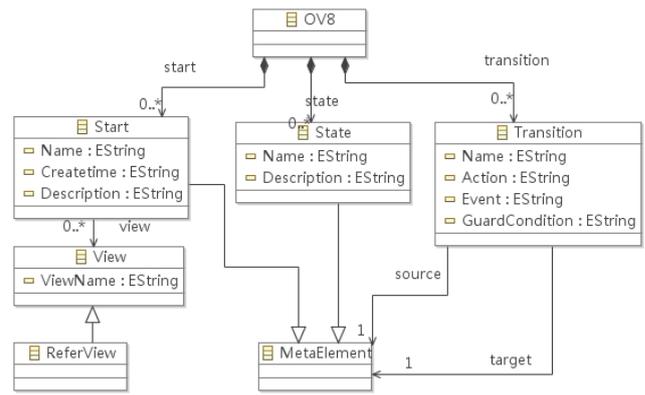


Fig. 4. OV-8 meta-model.

And the language is needed to define how to modify the the architecture to represent the different scene. The language is defined below:

Operator:

Load:

Load the initial system architecture for the system

Syntax:

Load filename

Example:

Load example.ov2

Modify:

Modify a system element according to the system design.

Syntax:

Modify element [add or delete]

Example:

Modify operationalactivity add

If else:

Judgement statement.

Syntax:

If ( something is true ) do

Else do

Example:

If (true) modify operationalactivity add

Now we have design the basic element to support the self-adaptive system.

#### IV. UAV EXAMPLE

In the chapter, there is an example to describe the example mentioned in the article beginning.

Following is the OV-2 view that describe what is the UAV system:

But there is a situation that we mentioned above. It is the uav must adjust the fight flow to complete. So we can define the language to support the change.:

Load ov5

Modify needline add fight to UAV control

Then you can auto-generate the next scene system architecture bellow:

Here we just provide the OV-5 for simplify, and the other we can get the same result. As we have seen, the method is a good way to represent the self-adaptive system.

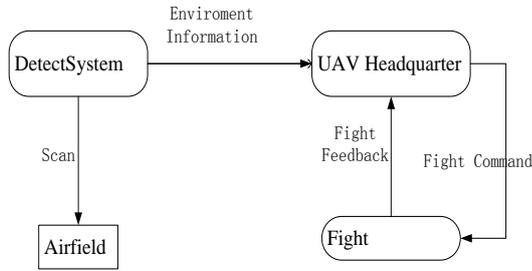


Fig. 4. OV-2 view

OV-5 describes how the mission is completed by UAV:

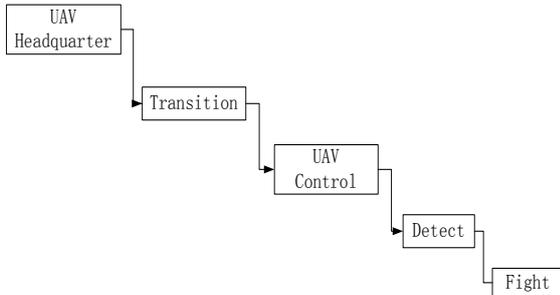


Fig. 5. OV-5 view

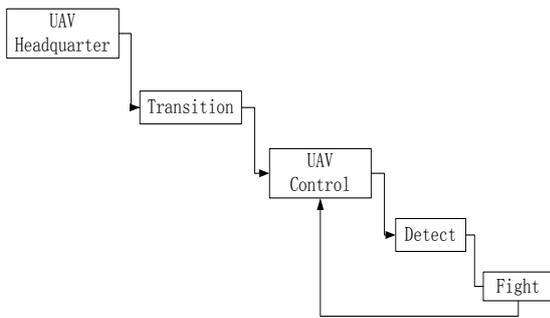


Fig. 6. OV-5 changed View

### V. CONCLUSION

Based on the research included in this, we can find this is a good way to develop the self-adaptive system architecture. The experiment results that the self-adaptive system modeling method can reduce the time consumed to design the different system architecture. And this is a method to how to extend the DoDAF.

### ACKNOWLEDGMENT

This work was partly supported by the National Natural Science Foundation of China No.90718018, 863 subjects(2007AA010302). The inspiration emerged from previous work about the system architecture modeling.

### REFERENCES

- [1] P. Oreizy, M. Gorlick, R. Taylor, D. Heimbigner, G. Johnson, N. Medvidovic, A. Quilici, D. Rosenblum, and A. Wolf, "An architecture-based approach to self-adaptive software," *IEEE Intell. Syst.* vol. 14, no. 3, pp. 54-62, 1999.
- [2] J. Dowling and V. Cahill, "The K-component architecture meta-model for self-adaptive software," in *Proceedings of Reflection 2001. Lecture Notes in Computer Science*. Springer-Verlag, Berlin, vol. 2192, pp. 81-88, 2001.
- [3] N. Kilicay and C. Dagli, "Methodologies for understanding behavior of system of systems," in *Proc. of Conference on System Engineering Research*, Hoboken, New Jersey. March 14-16, 2007.
- [4] T. Li, "Model-based self-adaptive embedded programs with temporal logic specifications," in *Proc.6th International Conference on Quality Software*, 2006.
- [5] F. Oquendo, "π-Method: A model-driven formal method for architecture-centric software engineering," *ACM SIGSOFT Software Engineering Notes* vol. 31, no. 3, pp.1-13, 2006.



**Guanyue Gao** was born in Henan Province in 1988. Between 2006.9 and 2010.7, he was a student of ZhengZhou University, and he got the bachelor degree in 2010.7. Now he is a student of Beihang University, and major in system architecture modeling research as a master.



**Feng Liu** was born in 1978. Between 2007.9 and 2010.6, he was a student of Beihang University, and he got the master degree in 2010.6. he majored in the software architecture design.