An UML-Based Meta-modeling Method of Building Architecture Product

Jiong Fu, Aimin Luo, Xueshan Luo

An UML-Based Meta-modeling Method of Building Architecture Product

Jiong Fu [1], Aimin Luo [1], Xueshan Luo [1]

[1] Science and Technology on Information Systems Engineering Laboratory
of National University of Defense Technology, Changsha, P.R.China
fu9jiong2@163.com, {amluo, xsluo}@nudt.edu.cn

Abstract. Architecture product modeling is an important means to collect enterprise-wide decision-making data. DoDAF2.0 proposes meta-model theory to facilitate organizing and collecting data, but still no clear concrete modeling methods for collecting data are available. This paper presents an UML-based meta-modeling method of building architecture product. This method contributes by solving the existing problem within product modeling, and by using the method we can get the modeling tool easily.

Keywords: Architecture, Meta-model, Meta-modeling, UML.

1 Introduction

Information system architecture is defined as the structure of components consisting of its system, relationships, and the disciplines and guidelines governing their design and evolution over time [1]. At present, much research on information system architecture is generally conducted based on the research of DoDAF, and thus research around modeling of architecture product is an important part. Though DoDAF2.0 [2] provides the criterion and method for data organizing and collecting, the method and process for product modeling is left absent. As a result, there is a lack of maneuverable guidance for product modeling, and this lack will affect data collecting and moreover affect decision-making. Presently, Object Management Group(OMG) recommended MOF as the standard of meta-meta-model, but still not accepted widely [3]. Lan [4] introduced the basic building process of executable meta-model in MDA domain. However, the research on architecture product modeling has little progress.

In order to settle the problem, refer to the meta-modeling method and technologies in MDA domain, this paper proposes a meta-modeling method framework and elaborates the basic process of information system architecture product modeling. Further, this paper studies three key technologies of the meta-modeling of architecture product, i.e., (I) meta-meta-model, (II) meta-model and (III) meta-modeling tool. By using the method of meta-modeling of architecture product, we can solve the problem of product modeling, and it can provide the modeling tool easily.

This paper is organized as follows. In section 2, we present the architecture of the product element modeling framework. Section 3 discusses the method and basic

process of modeling meta-meta-model. In section 4, we propose a construction method based on UML for architecture product meta-modeling. In Section 5 we take Operational Resource Flow Description (OV-2) as an example to illustrate our method and the meta-model tool. Section 6 concludes the paper by noting the future works.

2 The Meta-modeling Method Framework of Architecture Product

No unified and standardized definition of architecture meta-model is available now. Based on its application, the architecture meta-model can be categorized into two categories, i.e., data meta-model and product meta-model. Data meta-model focuses on the criterions of architecture data, it can better support us to collect and store data. This part (Data meta-model) is the core of DoDAF2.0; while product meta-model focuses on the direct guidance for various architecture products modeling. It (product meta-model) is very important in process of architecture development and design. This paper mainly focuses on latter one, i.e., product meta-model.

The meta-modeling of architecture product is defined as the process of characterizing the meta-model of the modeling language of architecture product, customizing the modeling language of view product and configuring the modeling tool which supports the modeling language. The process of meta-modeling involves several important elements, including meta-meta-model, meta-model, model, meta-modeling tool and modeling tool. Different from the methods used to build DoDAF Meta Model(DM2), the product meta-modeling approach is mainly to solve the problem as to how to build meta-meta-model, product meta-model and to customize modeling tool.

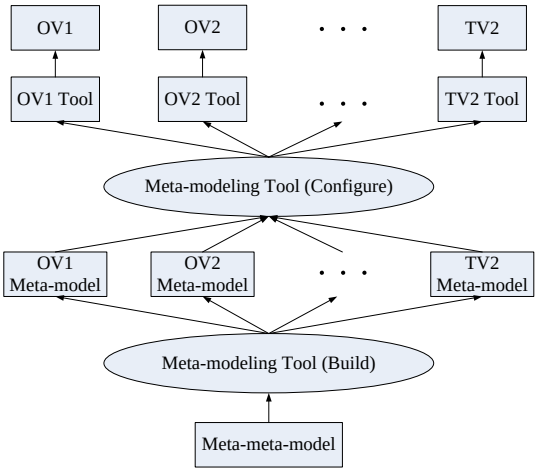


Fig. 1 The meta-modeling method framework of architecture product

For the problem of the lack of guidance and progress of the architecture product meta-modeling, refer to the meta-modeling method and technologies in MDA domain, this section presented a meta-modeling method framework of architecture

product and analyzed basic processes and main technologies of meta-modeling. Through the research, we improve the modeling framework of architecture product under the meta-model theory, and provide a method for architecture product modeling.

The meta-modeling method framework of architecture product is presented here, as shown in Figure 1. The meta-modeling method framework includes two main processes. One is the process of product meta-model built by meta-meta-model, and the other is the process of configuring modeling tool by product meta-model. Two processes are all based on the architecture meta-modeling tool.

3 Architecture Meta-meta-model

Architecture meta-meta-model is the basis for building architecture product meta-model, the contents of meta-meta-model directly impact on the contents of the product meta-model. In this section, we build a set of meta-meta-model that can be used to construct and describe architecture product meta-model.

The basic method of establishing architecture meta-meta-model is to extract all the essential attributes that describe the architecture, under the deep analysis of the 52 View product models in DoDAF2.0. And then extend specific attributes in architecture domain, refer to Ecore model [5] in MDA. Specific steps can be divided into the followings:

Step 1: Classify the 52 products of DoDAF2.0 by different visual styles, shown in Table 1.

Table 1 Classification of Architecture Products by Visualization Style

Visualization Style	Architecture Products
Digraph/Graph	OV-1、OV-2、SV-1、SV-2、CV-1、CV-2、SvcV-1、SvcV-2、SvcV-4、DIV-1
Table/Matrix	OV-3、SV-3、SV-5a、SV-5b、SV-6、SV-7、SV-9、AV-1、AV-2、CV-4、CV-5、CV-6、CV-7、SvcV-3a、SvcV-3b、SvcV-5、SvcV-6、SvcV-7、SvcV-9、StdV-1、StdV-2、PV-1、PV-3
Hierarchy Diagram	OV-4、OV-5、SV-4、CV-2
State Transition Diagram	OV-6b、SV-10b、SvcV-10b
Timing Diagram	OV-6c、SV-10c、SvcV-10c
Timeline Diagram	SV-8、CV-3、SvcV-8、PV-2

Step 2: For each visual style, analyze and extract essential elements.

The visualization styles in step 1 can be roughly divided into two categories: "Node + Connection" type , including digraph/graph, hierarchy diagram, state transition diagram, timing diagram; and "Table" type, including table and matrix , timeline diagram.

For the "Node + Connection" type, entity and relationship are the core data elements. There are some other data elements, e.g. property and type. Entity can be in place of icon, vertex and node in products, also refers to specific activities or

functions. Relationship can be in place of connection, arc, input connection or output connection. Entity and relationship can have a number of categories and attributes, relationship can be subdivided into binary relationship, inheritance relationship, association, aggregation relationship, etc.

For the "Table" type, the row, column, or cell of the table can be seen as an entity, while other data elements include property, type and comment.

Therefore, summarizing the above analyses, the essential elements of different visualization style are shown in Table 2. The essential data elements of architecture products include: Entity Relationship (binary relations, inheritance, association and aggregation), property, type and comment.

Table 2 Essential Elements of Different Visual Style

Type	Visual Style	Core Data Elements	Other Data Elements
"Node + Connection" Type	Digraph/Graph	Entity 、 Relationship	Property、Type
	Hierarchy Diagram		
	State Transition Diagram		
	Timing Diagram		
"Table" Type	Table/Matrix	Entity	Property 、 Type 、 Comment
	Timeline Diagram		

Step 3: Refer to Ecore model, extend specific attributes in architecture domain, and we get architecture meta-meta-model, as shown in Fig. 2.

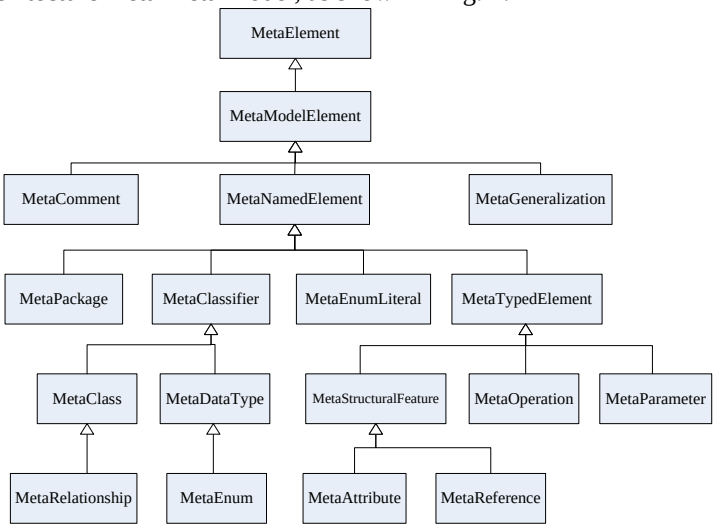


Fig. 2 Architecture meta-meta-model

The elements used for meta-modeling in architecture meta-meta-model shown in Fig. 2 include: MetaClass, MetaRelationship, MetaEnum, MetaPackage, MetaComment, MetaReference, MetaGeneralization and MetaAttribute.

MetaClass is used to define building blocks of the modeling language, for example, when building meta-model of Operational Resource Flow Description (OV-2), the MetaClass is used to describe operational nodes and operational activities. MetaRelationship is used to define binary relationship in modeling language, such as association, generalization, realization, etc. MetaEnum is used to define enumeration

types of modeling language. MetaPackage is used to manage and organize the blocks defined. MetaComment is used to define the annotation to aid in understanding the modeling language. MetaReference is used to define association and aggregation relationship between blocks. MetaGeneralization defines inheritance relationship between blocks. MetaAttribute defines the properties of elements defined.

4 The Architecture Meta-model

The meta-model of architecture product includes abstract syntax model, surface syntax model, semantic model. This section uses UML-based approach to build meta-model of architecture product. The idea is shown in Fig. 3.

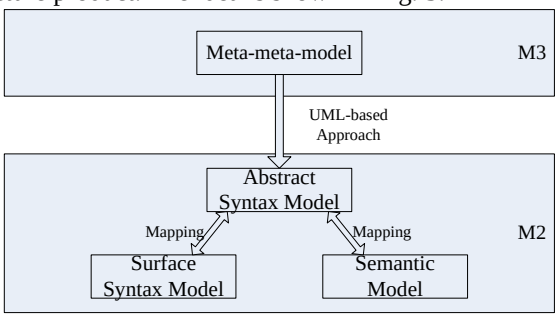


Fig. 3 UML-based Approach to Build Meta-model

In Fig. 3, the UML-based approach to build meta-model of architecture product contains the following three aspects: the abstract syntax model built by meta-meta-model, the surface syntax model and its mapping with abstract syntax model, the semantic model and its mapping with abstract syntax model.

4.1 Abstract Syntax Model

Abstract syntax model mainly describes concepts, the relationships between concepts and constraint rules. The build process of abstract syntax model can be broken down into several stages, namely concepts identifying, concepts modeling, building formal rules between concepts, add necessary operations, model validating and model testing [6]. Fig. 4 shows the abstract syntax model building process and the corresponding relevant contents.

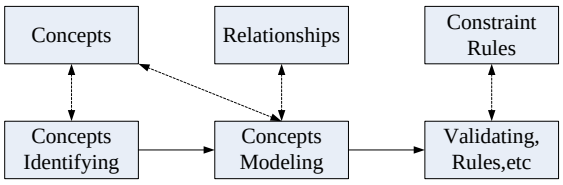


Fig. 4 The Abstract Syntax Model Building Process

Concepts modeling is the core stage of building the abstract syntax, it can be divided into the following two steps:

Step1: Classify the identified concepts.

Here to sort out which concepts are MetaClass, which concepts are MetaRelationship, which concepts are MetaAttribute. Typically, the concept which means entity can use MetaClass to describe, the relationship between entities can use MetaRelationship to describe, and the attributes of entity or relationship can use MetaAttribute to describe.

Step2: Use UML-based approach for modeling.

After classifying the identified concepts, we can use UML-based approach to model the concepts and relationships of abstract syntax model.

4.2 Surface Syntax Model

Surface syntax model describes the details of the abstract syntax model. Surface syntax model divides into two parts: text surface syntax and graphical surface syntax, this paper only discusses its graphical surface syntax, specifically including four aspects, namely, graphical elements, combination layout, location and mapping between the abstract syntax model and graphical surface syntax model [7].

Graphical elements are the basic structural elements of graphic symbols, such as line, rectangle, ellipse, and so on. In follow of the principles of usability, completeness, conciseness, we get the basic graphical elements, including rectangle, round rectangle, diamond, triangle, polygon, ellipse, circle, line, polygonal line, arc, text object, image object and angular rectangle.

Combination layout is a combination of several elements combined in a certain way to become a complex graphical symbol. Combination layout reflects the internal structure of the surface syntax model. Combination layout divides into two types: one is nested combination between block graphical elements, while the other is the combination of linear graphical elements combined with other graphical elements.

Unlike combination layout, location reflects possible positional relationship of graphic symbols. There are many kinds of positional relationships between graphic symbols, and we summarize as five types: nested type, connected type, block-block attached type, block-line attached type and line-line attached type.

After defining combination layout and basic positional relationships between graphical symbols, the mapping between the abstract syntax model and graphical surface syntax model is also very important. The mapping can be divided into element mapping and positional relationship mapping. The element mapping is to attach graphical symbols to modeling elements, while positional relationship mapping is to attach location to the relationships between modeling elements.

4.3 Semantic Model

Abstract syntax model describes the structure of modeling language, and surface syntax model describes the manifestation of modeling language. In contrast, semantic model describes the meaning of the concept of modeling language.

Abstract syntax model describes the structure of modeling language, and surface syntax model describes the manifestation of modeling language. In contrast, semantic model describes the meaning of the concept of modeling language.

In this paper, we use UML-based extensional semantics [8] approach to describe the semantic model. The so-called UML-based extensional semantics approach is to extend the existing UML semantics specification to adapt the semantic description of architecture model language. The way of extension is by the way of object-oriented inheriting with precise semantics of the UML model elements, thus reusing existing language semantics.

The core elements of UML models with precise semantics include Class, Package, Association and Property, while the sorts of the core elements of the abstract syntax model of the architecture modeling language include MetaClass, MetaPackage, MetaRelationship and MetaAttribute. Let them inherit from Class, Package, Association and Property of the UML language, as shown in Fig. 5, so that they have semantics when building abstract syntax model.

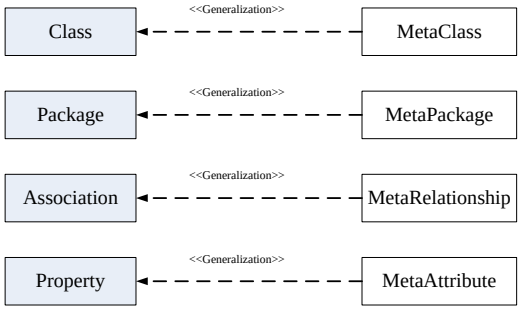


Fig. 5 UML-based Extensional Semantics

5 The Architecture Meta-modeling Tool and Case Study

Architecture meta-modeling tool is the carrier of meta-modeling process, as well as the manifestation of the meta-modeling method of architecture product. Architecture meta-modeling tool should be able to construct product meta-model, and also can customize the corresponding modeling tool. So compared with the traditional architecture modeling tool, meta-modeling tool has better expansibility, and meta-modeling can greatly reduce workload of tool development.

By developing our architecture meta-modeling tool prototype, we use Java language which is platform-independent to code, and use integrated development platform Eclipse to develop the tool. Besides we use Graphical Editing Framework (GEF) to achieve the graphical representation of modeling elements. The visual interface of the tool includes six parts, followed by toolbar, palette, editing area, navigation area, property area and outline area, as shown in Fig. 6.

We take Operational Resource Flow Description (OV-2) as an instance to illustrate the meta-modeling method and the tool introduced above.

Operational Resource Flow Description (OV-2) describes the resource information exchanged between operational activities. Now assume a simple scenario: we assume

that a combat unit has a simple process including intelligence gathering, information processing, combat command ordering and combat performing. Table 3 lists all the scenarios including operational activity, operational information and operational node.

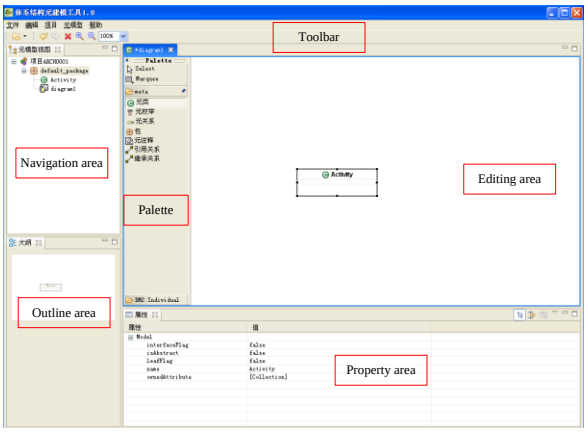


Fig. 6 The Visual Interface of The Tool

Table 3 Activities, Information and Node

Operational Activity	Operational Information	Operational Node
Intelligence gathering	Intelligence information	Accused node
Information processing	Processed information	Processing node
Combat command ordering	Combat command	Firepower node
Combat performing		

(1) Build meta-model of OV-2

Firstly, we need to build OV-2 abstract syntax model. The concepts meaning entity include Activity, OperationalNode, Information, Organization, Condition, and they can be created by MetaClass. NeedLine is a kind of relationship and can be created by MetaRelationship, while ActivityConsumesResource, ActivityProducesResource, ActivityPerformedUnderCondition and ActivityPerformedByPerformer are detailed description of relationships, and they can be created by MetaReference. So OV-2 abstract syntax model is shown in Fig. 7.

Fig. 7 shows that OperationalNode contains Activity, while Activity contains Condition and Organization. NeedLine contains Information, and both source end and target end of NeedLine are Activity.

Then, build OV-2 surface syntax model. Since Organization and Condition can be regard as attributes of Activity, they don't need graphical symbols, the same to Information. Only OperationalNode, Activity and NeedLine need to define graphical symbols. The OV-2 surface syntax model is shown in Fig. 8.

Fig. 8(a) shows that rectangle represents OperationalNode, and round rectangle represents Activity, while the combined symbol of a solid line with an arrow represents NeedLine. The location of OperationalNode and Activity is belong to block-block attached type, while the location of Activity and NeedLine is belong to connected type. Fig. 8(b) shows the modeling interface of surface syntax model.

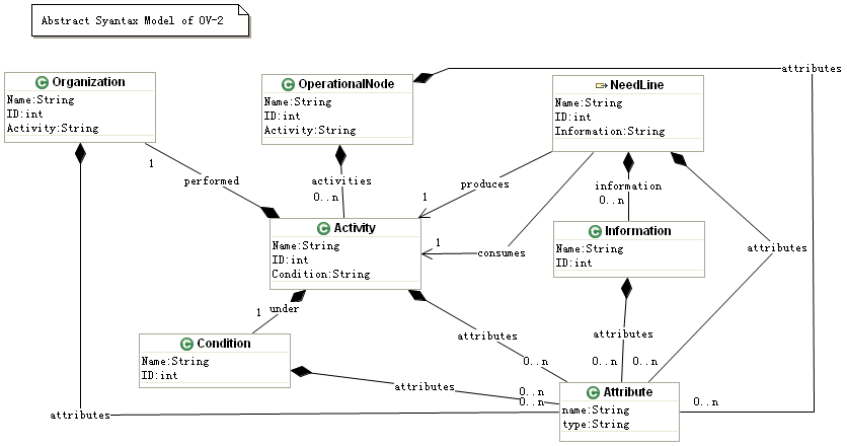
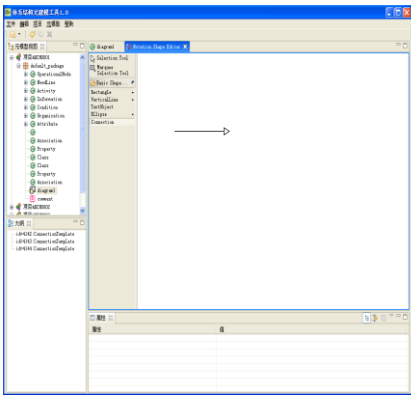


Fig. 7 OV-2 Abstract Syntax Model

Concepts/Relationships	Graphical Symbol/Location	Type
OperationalNode		Element Mapping
Activity		
NeedLine		
OperationalNode & Activity		Positional Relationship Mapping
NeedLine & Activity		

(a)



(b)

Fig. 8 OV-2 Surface Syntax Model

Besides, by analyzing OV-2 abstract syntax model, we can find that Activity, OperationalNode, Information, Organization, Condition are instances of MetaClass, so they inherit semantics from Class. NeedLine is an instance of MetaRelationship, and it inherits semantics from Association. Attribute is an instance of MetaAttribute, and therefore it inherits semantics from property.

(2) Configure to generate OV-2 modeling tool

Architecture meta-modeling tool is able to analyze the product meta-model we build, and customize the modeling tool which supports the product. Fig. 9 shows the modeling interface of OV-2 modeling tool, which is created by the meta-modeling tool after analyzing OV-2 meta-model.

Fig. 9 shows that yellow round rectangle represents Activity, and rectangle represents OperationalNode, and the straight line with arrow represents Needline. When modeling, Activity can only attach to the boundary of OperationalNode, while Needline can only be created between activities.

From the content and visual style of the OV-2 product we built, we can find they correctly reflect the meaning of OV-2. Thus, the architecture meta-modeling tool has validated our meta-modeling method of architecture product.

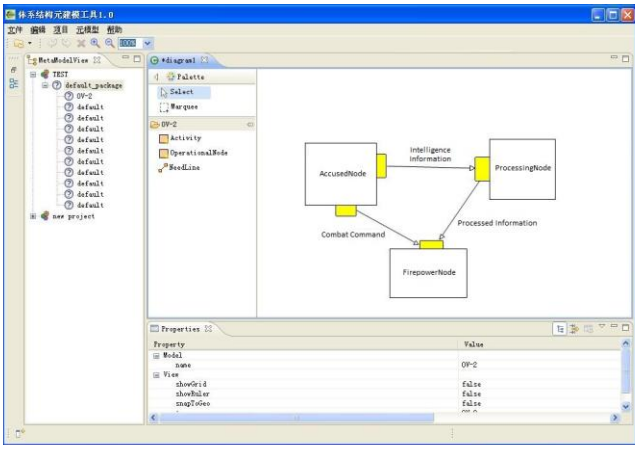


Fig. 9 The Modeling Interface of OV-2 Modeling Tool

6 Conclusions

This paper presents a meta-modeling method of architecture product. We analyze the main technical issues of meta-modeling, including meta-meta-model, product meta-model and meta-modeling tool. Based on the tool, we analyze and validate the meta-modeling method of architecture product via a case. This research provides a new method for architecture product modeling with the theory of meta-modeling. This new method enables enterprise modeling with meta-models. Future work will be conducted with a focus on integrating other modeling methods into the as is tool.

References

1. IEEE Architecture Working Group, IEEE Recommended Practice for Architecture Description, IEEE Std 1471, Draft Version 3.0, USA, 1998.
2. DoD Architecture Framework Working Group. DoD Architecture Framework Version 2.0: Architect's Guide-Architecural Data and Models [R].U.S.: Department of Defense, 2009.
3. LIU Hui, MA Zhi-Yi, SHAO Wei-Zhong. Progress of Research on Metamodeling[J]. Journal of Software, Vol.19, No.6, June 2008, pp1317-1327.
4. Lan Qingguo. Research on the Key Technology of Executable Metamodel[D]. Jilin University, 2006.
5. Ecore. The Core of EMF, Package org.eclipse.emf.core
6. JIANG Zhi-ping, HE Ming, WANG Zhi-xue, QIU Hang-ping. The Research on the Data-centered C4ISR System Architecture Development Method [J]. Fire Control & Command Control, Vol.34, No.1, January, 2009.
7. HE Xiao, MA Zhi-Yi, SHAO Wei-Zhong. A Metamodel for the Notation of Graphical Modeling Languages [J]. Journal of Software, Vol.19, No.8, August, 2008.
8. Tony Clark, Andy Evans, Paul Sammut, James Willans, Applied Metamodeling A Foundation for Language Driven Development Version 0.1[EB/OL]. http://albini.xactium.com/content/index.php?option=com_remository&Itemid=28