Websites:
http://www.sciencepub.net
http://www.sciencepub.net/researcher

Emails:
editor@sciencepub.net
marslandresearcher@gmail.com

Researcher

MARSLAND PRESS
Multidisciplinary Academic Journal Publisher

XMI-based Integration Model of Heterogeneous Formal Method in Embedded Software

Haiyang Xu[1, 2], Yi Zhuang [1]

[1.] Department of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China
[2.] Department of Science and Information Science, Qingdao Agricultural University, Qingdao, China
xhy@nuaa.edu.cn

Abstract: Formal model is important during the software life cycle. It uses formal language to precisely describe model specification, and can be used to model reasoning and verification. MARTE is an UML profile for real-time embedded software, and it can model the non-function properties. But it is lack of precise semantic, and cannot analyze and verify software model with effect. Combining the advantages of Object-Z and PTA in formal modeling, we propose an integration model of heterogeneous formal method, called PTA-OZ. This model can be used to analyze and verify the static structure and dynamic sematic of embedded software at the same time. Since formal method is dull and is not convenient for software developer, we design an executable transformation between MARTE model and integration model under MDA, and also base on XMI to realize the transformation.
[Xu H, Zhuang Y. **XMI-based Integration Model of Heterogeneous Formal Method in Embedded Software**. *Researcher* 2021;13(9):66-78] ISSN 1553-9865 (print); ISSN 2163-8950 (online). http://www.sciencepub.net/researcher. 8. doi:10.7537/marsrsj130921.08.

Keywords: Software life cycle; integration formal; heterogeneous method; XMI; meta model

1. Introduction

In the software engineering field, software life cycle is used to describe the whole process from software design to complete. This process can be represented by software development model, while formal model is one of the important software development model.

With the developments of computer technology, embedded systems have been widely used to control and manage of military equipment, such as unmanned aerial vehicles, warships or airplane. Embedded software is an important part of embedded system, and software quality can directly affect the performance of the embedded system. How to improve the trust of embedded software has become the important subject in the development of embedded software. MARTE (Modeling and Analysis of Real Time and Embedded systems) is a UML profile, which is released by OMG in 2007. It contains some modeling elements of non-function properties, such as real time, source. MARTE extends modeling ability of UML for modeling and analyzing real-time embedded software by adopting stereotypes, tagged values and constraints(Mallet et al, 2009, André et al,2009). Many researchers and research institutions base on MARTE specification to model, analysis and verify real time and embedded software. To efficiently design real time and embedded software, they present a MOPCOM design methodology (Vidal et al,2009, Lecomte et al,2011), which combine the advantages of MARTE with UML. Researchers use MARTE to describe and model real time properties of embedded software, and design a set of transformation rules that could be used to transform model into VHDL code. Yu et. Al (Yu et al,2008) adopts MARTE to model high-performance embedded software, and make use of formal model checker to verify correctness of design model. Most of those methods use MARTE to build embedded software model, then present some transformation rules to transform model into formal method or generate software code. So model transformation plays an important role in MARTE model and formal verification.

Our motivation is how to reduce the risk of software development at the early stage of software life cycle, and how to adequately analysis and verify MARTE model. All of the existing researches are to transform MARTE models into one formal method, which only verify one aspect of MARTE model. Combining the advantages of Object-Z and PTA, we present an integrated heterogeneous formal model, which is named PTA-OZ. This model makes use of the advantages of Object-Z in structure verification, and use PTA to verify behavior semantic. Under MDA, we present an executable transformation model from MARTE model to PTA-OZ, and realize the model transformation in XMI.

The rest of the paper is organized as follows. The basic information of XML and XMI is

introduced in Section 2. The integration model is given in Section 3. In Section 4 we present how to implement the transformation from MARTE model to integration model using eclipse modeling tools. Section 5 gives a case study to illustrate the transformation framework, and gives some part of the transforming result. Concluding remarks are gives in Section 6.

2. Preliminary

XML(Extensible Markup Language) is one of the W3C standards, it provides the general format of structural data and file interchange for data exchange, web service, and content management. XML/XSL have been used to develop a web environment of Object-Z, and they realize the transformation from Object-Z to UML(Sun et al, 2001). In the next generation network, XML is used to express the data of different ontology, and Z can be used to describe the behavior of semantic web, then Z schema can be transformed into OWL(Web Ontology Language) (Khan et al,2012).

Although XML defines the structural elements and properties, XML does not support object-oriented feature and object-oriented schema application (Grose et al,2002). XMI (XML-based Metadata Interchange) is released by OMG, which is a standard method of meta model exchange and bases on XML. It provides a standard to express object data, and it can make up the problems existed in the transformation from object to XML. XMI provides a standard method to map object of UML type to XML, and XMI 2.0 also provides the mapping between UML model and XML schema.

XMI does not define a XML document, but define an algorithm to generate XML document from meta model. It can describe how to generate XML schema from model, then we can use XML schema to verify XML document. XMI also can be used to describe how to generate model form XML document, DTDs and schema. In the paper, we call the XML document which is generated by XMI method XMI document.

XSL(Extensible Stylesheet Language) is specially made for XML by W3C in 1998, and it is mainly used to transform XML document to arbitrarily structural target document, such as XML document. XSLT(Extensible Stylesheet Language Transformations) is an important part of XSL, it defines the transfer language of XML document to describe how to realize the transformation of a XML document. In the practical application, we choose XSLT as the style sheet to control the display of XML.

XSLT Stylesheet consists of a set of template rules. Template rules are used to describe the location and context of node of source document in target document tree. Each template rule contains pattern part and template part. Pattern part is used to match source node in source document, while template part defines the method to handle source nod and to define target node, it contains some element instructions, the copied data from XML source document or new data. The key of XSLT transformation process lies in whether could design a well form XSLT style sheet.

3. Integration Model

3.1 MARTE meta model

MARTE follows the conception of UML, and it uses different package to classify the conception. The conceptions in MARTE consist of foundation, modeling and analyzing(Faugère et al,2007). While foundation model package defines the foundational conceptions used to model-based modeling and analyzing for real time and embedded systems. Modeling parts support the requirement of specification and the detailed design. Analyzing parts define the abstract and specific elements to annotate models, and support the analysis of performance and dependability.

We focus on the high-level modeling concepts of MARTE for real-time and embedded system, especially the definition of RtUnit and PpUnit. The design models of MARTE encapsulate in the HLAM (High-Level Application Modeling) package. It provides the model element at abstraction level to model the real-time active object and behavior, such as the independent control module, the protection module for limited resources and the dynamical schedulable resources(Quadri et al,2012).

Figure 1 shows the basic design elements of HLAM model from the perspective of domain conception. RtUnit and PpUnit are used to describe the active objects, and mainly to model the real time behavior and service of systems.

Figure 2 is the definition of RtUnit and PpUnit as the stereotype in MARTE profile. The definition of domain conception is different from the definition of stereotype, they focus on different fields. Domain conception emphasizes the conception of special field, and it directly reflects the conception model. While stereotype focuses on application object of special model in UML (Zhang et al,2009).

The class is modeled by meta classes, which inherit from the abstract meta classifier. RtUnit and PpUnit stereotypes are applied to meta class *BehavioredClassifier* through the extension relationship. When instancing the RtUnit and PpUnit stereotypes, they will be bound to an instance of given meta model as attached characters.

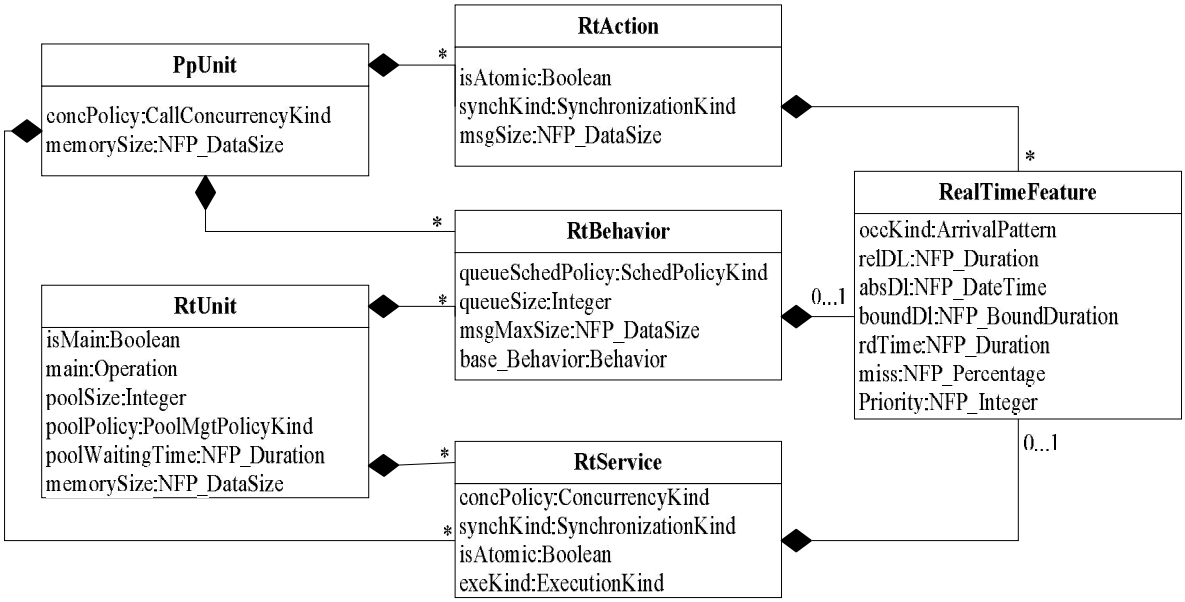

Figure 1. The HLAM model of MARTE

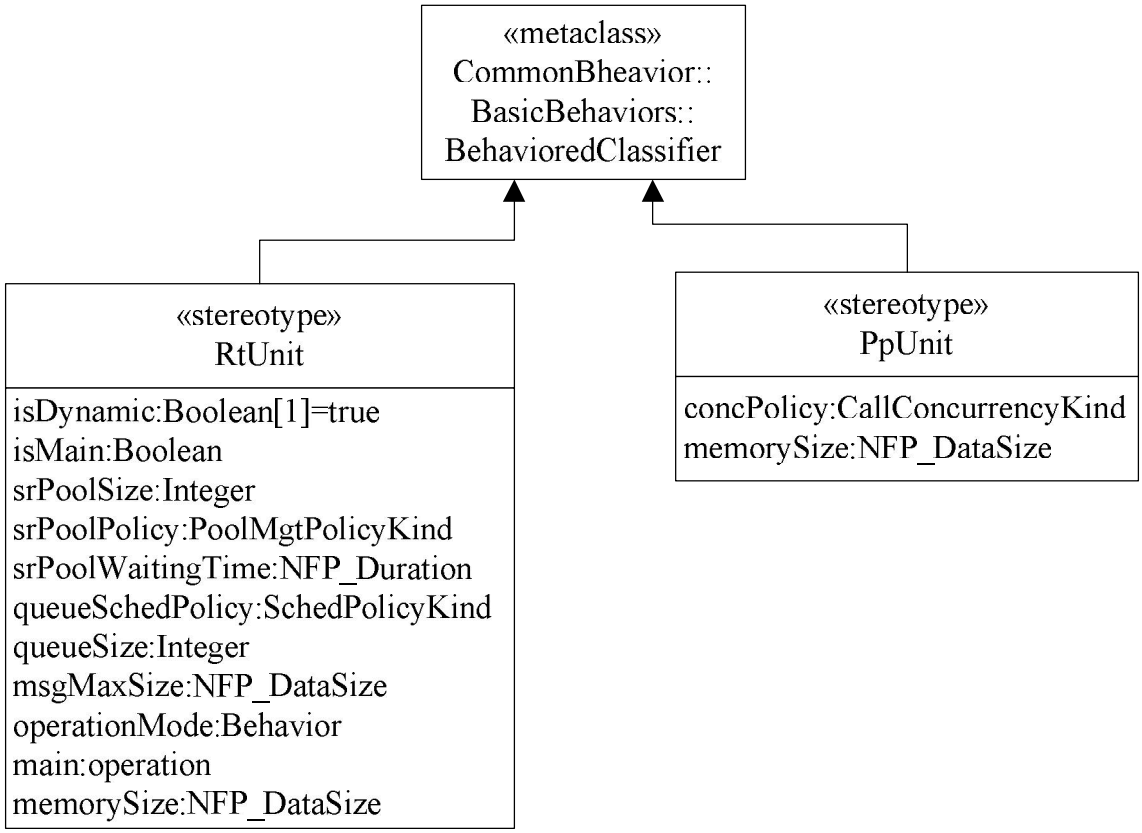

Figure 2. Meta model of RtUnit and PpUnit

3.2 The meta model of integration model

There are different kinds of diagram and chart in MARTE, so MARTE can describe and model the software system from different view. While MARTE is inaccuracy, the existing formal method only model software from one point of view. It is necessary for the developer to propose a formal method which can describe the software system from any point of view(Möller et al,2008).

Object-Z is a state-based formal language, and PTA is a behavior-oriented formal language. Combining the advantages of them, we propose an integration method of heterogeneous formal methods, which is called PTA-OZ.

Definition 1 A PTA-OZ is a 5 tuple PO=(*A*, *I*, *L*, *P*, *OZ*), where

- *A* is a ancestors and is used to inheritance, and it lists all the superclass using *inherit* clause;
- *I* consists of interface declaration, which is provided and used by class;
- *L* is a local channels, and it cannot be accessed outside;
- *P* is a probability timed automata, and it comprises all of the possible sequence invoked by method using PTA process notation. PTA part consists of some automata defined by PTA process equations;
- *OZ* is originated from Object-Z, and it contains a state schema, an initial schema and several operations.

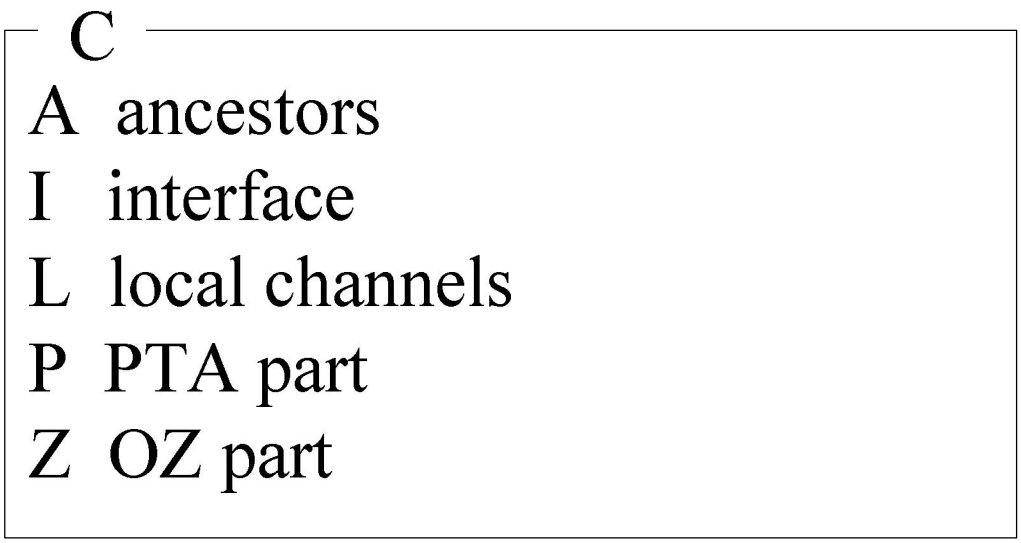

Figure 3. Class of PTA-OZ

Figure 3 is a PTA-OZ class called *C*. In our integration class schema, PTA part denotes the transitions between states in the form of behavior. OZ part is mainly used to express the state schema and operation schema in the form of object-oriented. We distinguish different type of declarations by the keywords *method*, *chan* and *local_chan*. *I* is the interface declaration, which extends from Object-Z visibility list. But it is different from Object-Z, the attributes of *I* are invisible, and they are private property of PTA-OZ class.

In order to facilitate the encoding of the software life cycle, the OZ part of PTA-OZ is different from Object-Z. We define two schemas called *enable_op* and *effect_op* for the operations in OZ part. The *enable_op* schema is used to specify the *guard* of the operation in term of states, input and the invoked simple parameters. The *effect_op* schema could specify the desired state transitions which are related to the states, the transition result state and all of the parameters.

4. Implementation framework

In the integration model of heterogeneous formalism method, OZ part and PTA part describe the static semantic and behavior semantic of the class respectively. In this paper, we only study how to realize the transformation from MARTE model to PTA-OZ model.

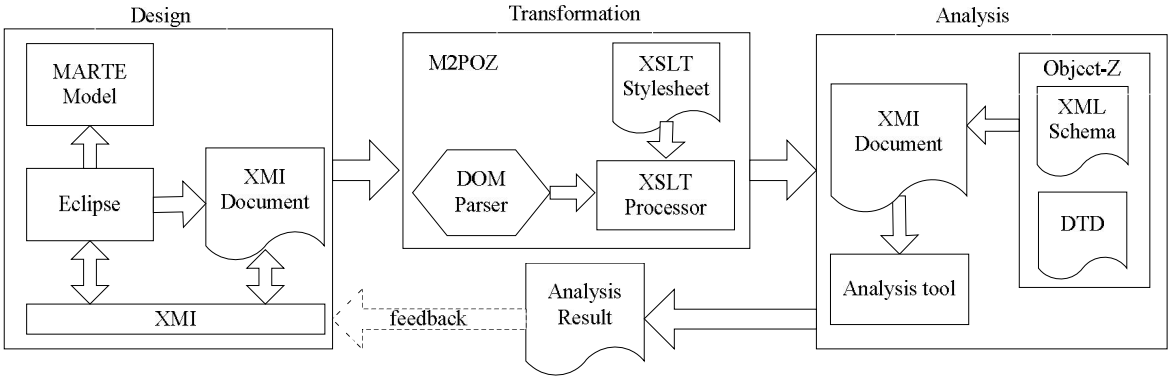

Figure 4. The framework of model verification

Figure 4 shows the transformation framework which transforms the MARTE model into OZ part of PTA-OZ model, and verification processes. The framework mainly contains the following steps:

- System design. We use the modeling elements of Papyrus platform to build the MARTE model of software system, and transform model in XMI document.
- Document transformation. We adopt DOM Parser to parse document, and store the parsing result into DOM tree structure that we construct. XSLT Stylesheet is used to control the transformation relationship from XMI document to result document. XSLT Processor transforms and parses XMI source document in term of the template rules defined in XSLT Stylesheet.
- Document analysis. According to the previous definition of XMI schema of Object-Z and DTD entity, we can achieve the Object-Z document, and analyze it in Object-Z type checker, such as CZT(Malik et al,2005).
- Result feedback. In term of the analysis results, we feedback them to the design model, so we can utilize the results to modify the error information in software model.

4.1 Modeling the real time software

There are lots of tools supporting the MARTE modeling, such as Papyrus, OpenEmbeDD. Papyrus is an open source environment which can be integrated into Eclipse, and it targets to provide a user-consumable platform for editing EMF (Eclipse Modeling Framework) model. Papyrus has a favorable graphical modeling interface to support diagram editors, and it supports several embedded real time software modeling language and specification, such as UML2, MARTE, SysML, EAST-ADL.

We download Eclipse modeling tools, and install Papyrus on the platform. So we can use the modeling elements of MARTE to construct software model on the Eclipse platform. As XMI is used as the definition specification by EMF, we can export XMI document for MARTE models which are constructed by Eclipse modeling tools.

The model data of Papyrus is stored in two XML document, *.uml and *.umldi. The *.uml document is used to store the structure information of model, such as the class name, parameters and operation information. The *.umldi document is used to store the expression information of model, such as the size of elements, the location and the font information. As the construction of MARTE model only need the structure information, we focus on the research of the definition of *.uml document.

The head part of the *.uml document is xml declaration, it consists of the version number, the encoding type and the description of namespace. The root node of the document is <*uml*:*Model*>, all of the model elements are recorded in its descendant nodes. The format is as below:

<uml:Model xmi:id="_gTnLY" name="model">

All the elements of MARTE model are respectively recorded in the sub-nodes of *packagedElement* node. The name of element is stored in the *name* attribute. The type of element is stored in the *xmi*:*type* attribute. The attribute value of class element is *uml*:*Class*. The attribute value of interface element is *uml*:*Interface*. The attribute value of dependency element is *uml*: *Dependency*. The attribute value of realization element is *uml*: *Realization*.

For the class and the interface, we also need to store their owned information among attributes and operations. Those information is recorded under class element *packagedElement* in the form of *ownedAttribute* and *ownedOperation*. The name of attributes or operations is as the *name* value of nodes. The type of attributes is stored in the *type* sub-nodes of the *ownedAttribute* node, and each parameter of class operations is showed in the *ownedParameter* sub-nodes of the *ownedOperation* node. The type of parameter is also stored in the *type* sub-nodes of the parameter nodes. The basic format are

<ownedAttribute xmi:id="_BdFzE" name="Attitude" visibility="public">
<ownedOperation xmi:id="_p43tE" name="ready"/>

XSLT parser would deal with the XMI document in the form of structure tree. So *.xml document just is a structure tree from the view of XSLT.

Figure 5 shows an example of structure tree of *uml*:*Class* type.

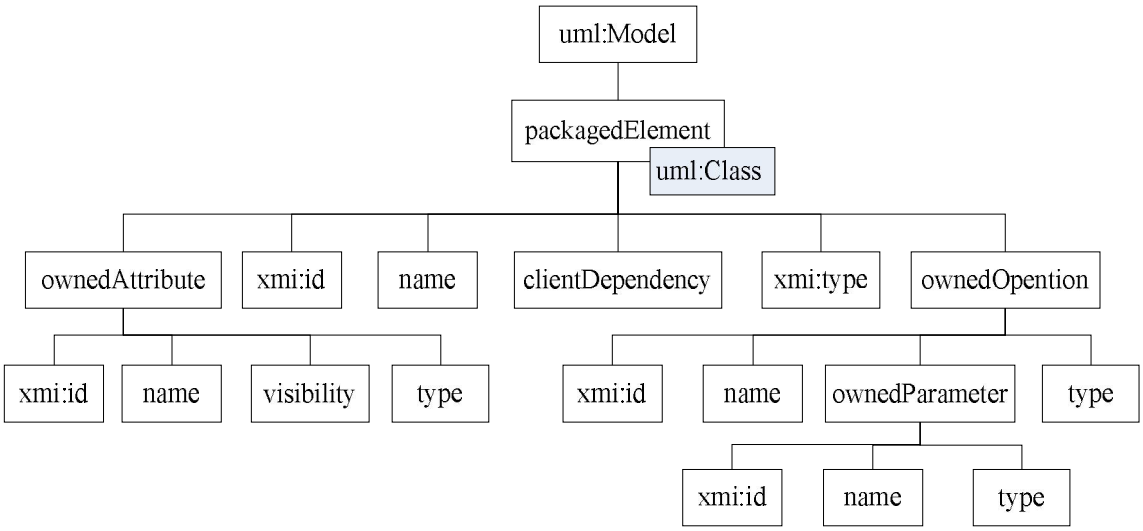

Figure 5. The structure tree of uml:Class type

4.2 The Transformation based on XMI and XSLT

There are several documents in our paper, including XMI document of MARTE model, XMI document of Object-Z model, XML schema document of Object-Z meta model. Now we give the relationships among them in MDA-based four layers meta model, as showed in

Figure 6. M1 is the model layer, and M2 denotes the meta model layer.

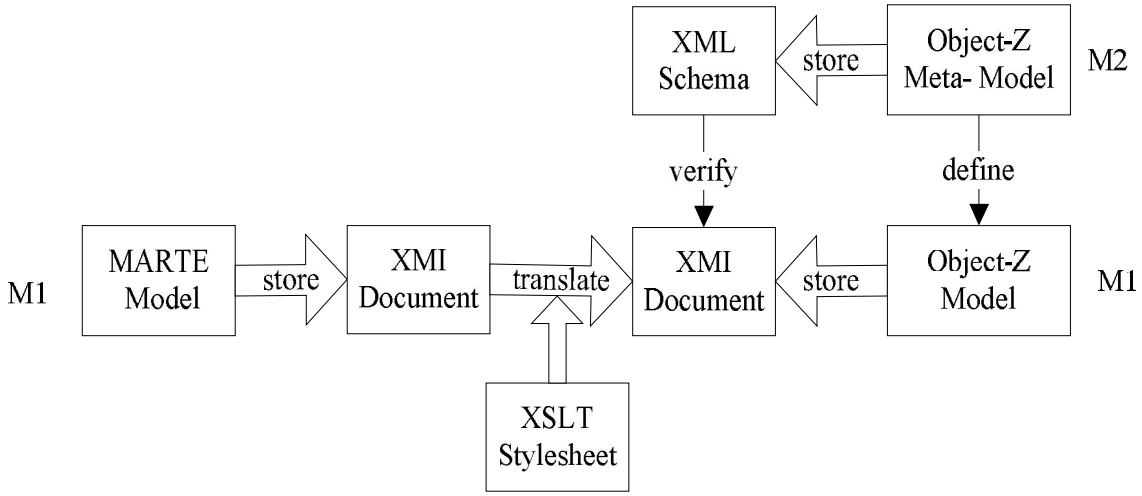

Figure 6. The relationships among models and XMI documents

For the XMI document derived from MARTE model, we need to transform it into XMI document of Object-Z. In the process, the key is that whether could design a good XSLT Stylesheet. The XSLT Stylesheet is used to construct the one-to-one mapping relationship between XMI document of MARTE model and XMI document of Object-Z, and the XMI document of MARTE model is automatically generated by Papyrus. So before designing the XSLT Stylesheet, we need to define the construction of XMI document of Object-Z in term of Object-Z meta model at M2 layer. That is, the XML schema document at upper layer can be used to define and verify the XMI document at lower layer.

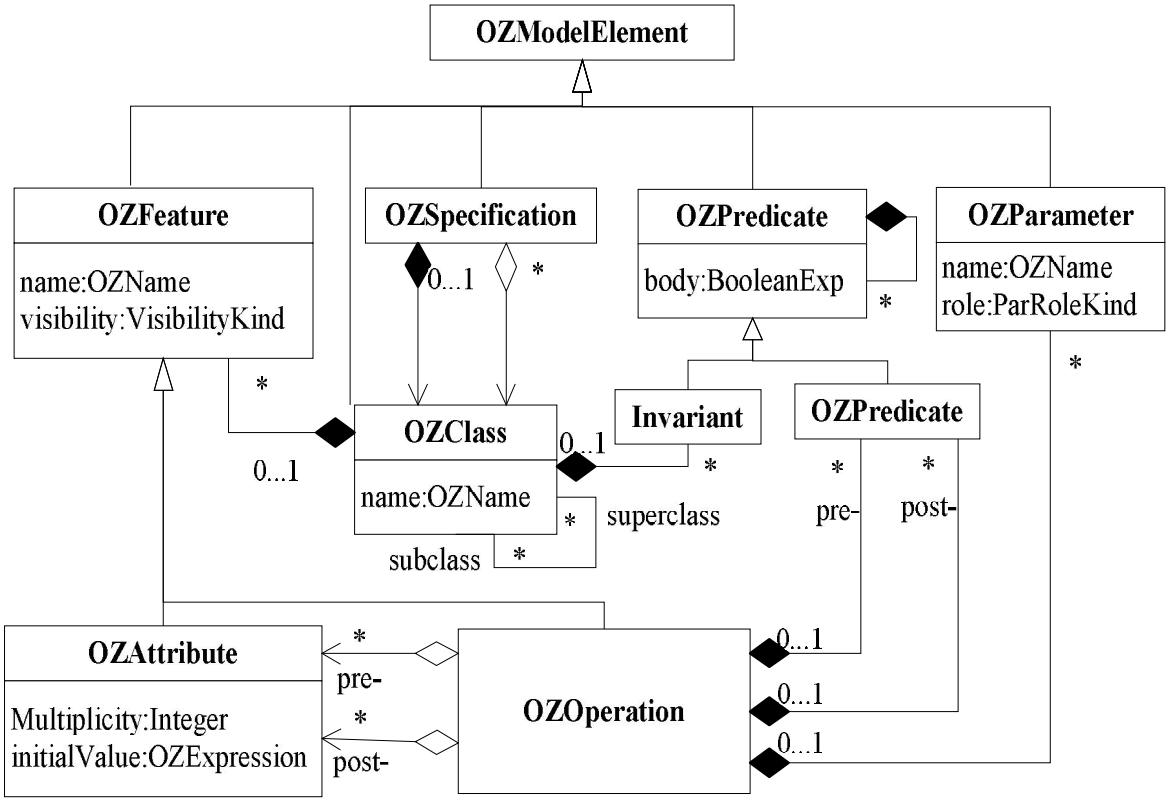

Figure 7. The core meta model of Object-Z

Object-Z uses object-oriented paradigm, and its basic concepts are similar to UML.When realizing the transformation of object-oriented model, Object-Z can reduce the complex compared with non-object-oriented formal specification(Kim et al,2002), such as Z.

Figure 7 is the structure diagram, and it shows the core component of meta model of Object-Z. *OZModelElement* is the topmost meta class, all the type of Object-Z meta model are derive from it. Now we define the XML schema of Object-Z in term of Object-Z meta model.

Figure 8 is part of XML schema of Object-Z meta model. Through the definition of XML schema document, we can define the syntax structure of XMI document of Object-Z model, which is used to verify the syntax validity of transformed XMI document.

```
<?xml version="1.0" encoding=" UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="classdef ">
      <xs:complexType>
        <xs:sequence>
                <xs:element ref="inherit" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element ref="state" minOccurs="0" maxOccurs="1"/>
                <xs:element ref="init" minOccurs="0" maxOccurs="1"/>
                <xs:element ref="op" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="state">
      <xs:complexType>
        <xs:sequence>
                <xs:element ref="decl" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element ref="st" minOccurs="0" maxOccurs="1"/>
                <xs:element ref="predicate" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

Figure 8. XML schema document of Object-Z meta model

For the mathematical notations in Object-Z, we can realize them by defining entity in DTD document(Sun et al,2001). The entity declaration in DTD document is as showed in Figure 9.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ENTITY delta "&#x0394;">
<!ENTITY integers "&#x2124;">
<!ENTITY and "&#x2227;">
<!ENTITY or "&#x2228;">
<!ENTITY forall"&#x2200;">
```

Figure 9. DTD document of Object-Z

In term of the definition of XMI schema document of Object-Z meta model, we will generate the XSLT Stylesheet and set up the relationship with XMI document of MARTE. In the XSLT Stylesheet, if the type of *packagedElemant* element in *UML*:*Model* is *uml*:*Class*, the *name* attribute is translated into the class name of Object-Z. For the sub-elements of *packagedElemant* element, the transformation rules are as follows:

- The *ownedAttribute* element is translated into the *state schema* of Object-Z;
- The *init* operation in *ownedOperation* element is translated into the *init* operation of Object-Z;
- The *ownedOperation* element is translated into the *operation* of Object-Z;
- The basic data type of MARTE is translated into the XMI owned data type or simple type;
- The stereotype of MARTE is translated into the complex type of XMI;
- For the *generalization* connection class and interface of MARTE class, we translate its elements into the class schema and operation of Object-Z:
  - ◊The *ownedAttribute* sub-element is translated into the *state schema* of Object-Z;
  - ◊The *init* operation is translated into the *init* operation of Object-Z;
  - ◊The *ownedOperation* sub-element is translated into the *operation* of Object-Z.

The structure information of MARTE model built by Papyrus can be stored in the form of XMI document, and then it can be denoted as a tree structure. The advantages of tree structure are legible layer and high readability. The XSLT-based transformation aims at the tree structure of document, so it is easy to analyze and extract transformation rules using XSLT technology. We can design a well-form style sheet for transforming, as showed in Figure 10.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="text" encoding="UTF-8"/>
<xsl:template match="/">
  <?xml-stylesheet type="text/xsl" href="OZSchema.xslt">
  <xsl:apply-templates select=" UML:Model/packagedElement "/>
</xsl:template>
<xsl:template match="packagedElement [@xmi:type=uml:Class]">
     <xsl:value of select="@name"/ >
     <xsl:apply-templates  select="generalization/ownedAttribute"/>
     <xsl:apply-templates  select="ownedAttribute"/>
     <xsl:apply-templates  select="ownedOperation[@name=init]|generalization/init"/>
     <xsl:apply-templates  select="generalization/ownedOperatio"/>
     <xsl:apply-templates  select="ownedOperation"/>
</xsl:template>
……
<xsl:template match="ownedOperation |generalization/ ownedOperation ">
     <xsl:for-each select= "ownedOperation" />
          <xsl:value-of select="@name"/> (
          <xsl:for-each select= "ownedParameter " />
               <xsl:apply-templates  select="ownedParameter"/>
          </xsl:for-each>)
    </xsl:for-each>
</xsl:template>
……
</xsl:stylesheet>
```

Figure 10. XSLT Stylesheet

4.3 Analysis of transformed result

The document transformation is to transform MARTE model into an XMI document of Object-Z, so we can verify the syntax validity of transformed XMI document using XMI schema of Object-Z. The result is a precise formal model, and its feature is precision.

In addition, the existing open tool CZT can edit and check the standard Z specification and Object-Z specification, and it can be used to parse, type check and transform the Z and Object-Z specification in the form of LATEX, Unicode and XML. CZT tool is open source and has high expansibility, and it can be integrated with other languages or systems. It is feasible to support the Z extension language. In term of the proposed framework, we transform MARTE model into XMI document which is fit of CZT form by using XSLT Stylesheet, and then analyze and verify the XMI document by CZT tool. At the end, in term of the verified XMI document, we will modify the original MARTE model.

5. Case study

We have performed our experiments on TCAS (Traffic Collision Avoidance System), to show how to generate XMI document from MARTE model, and to generate XMI document of Object-Z schema in term of the transformation framework proposed in Section 4.

5.1 Software modeling

TCAS is a set of embedded software system in aircraft to avoid mid-air collisions. TCAS II consists of four sub systems, and they are monitoring system, collision system, display & control system, monitor system. In accordance with the software engineering method, we firstly model the software system using MARTE. We adopt RtUnit and PpUnit to denote and model the active objects in embedded software.

RtUnit is a concurrent unit which contains the object and its process schema, and it supports asynchronous processing of messages to realize the control of concurrent. PpUnit is a protected passive unit, and it specifies concurrent access policy for shared data. PpUnit can model the protected source in system, and it is owned and controlled by RtUnit. The *Aircraft* class is an RtUnit unit, it realizes the *AircraftService* interface, and depends on the operations provided by *ConflictResolution* interface. It shares the data through *Signal* RtUnit unit. The relationships of them are as showed as Figure 11.

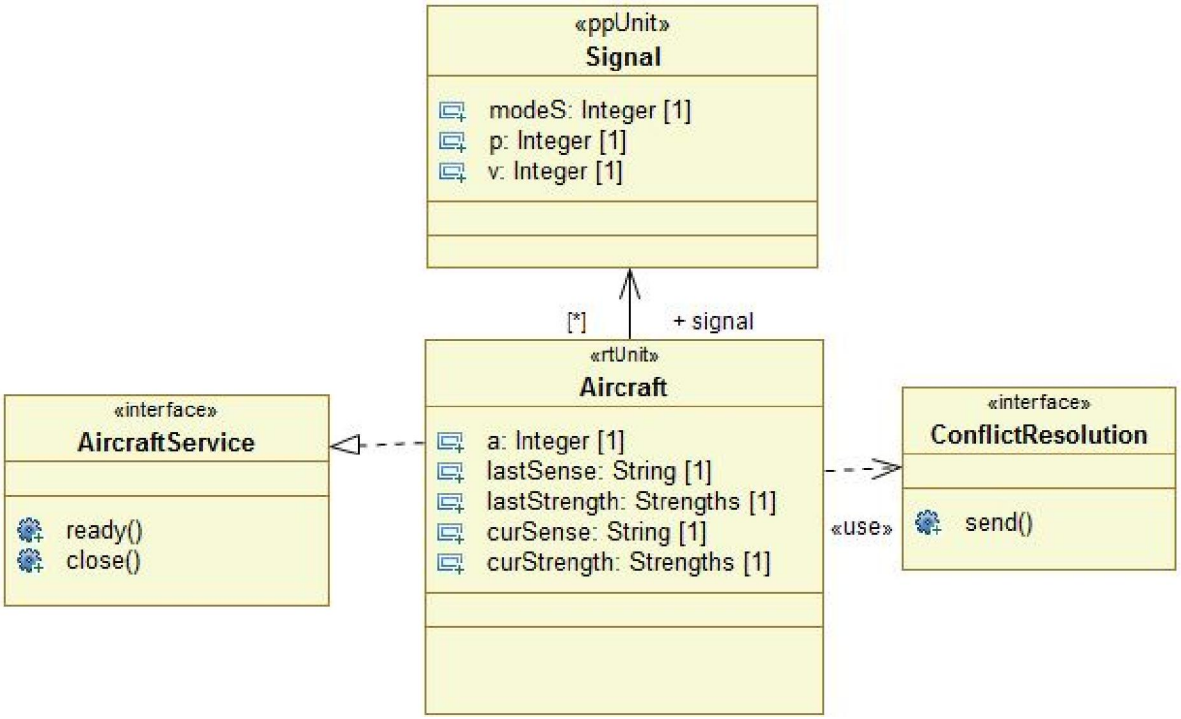

Figure 11. The MARTE class diagram of Aircraft

There is an *Association* relationship between the *Aircraft* RtUnit class and the *Signal* PpUnit class, which is model in Papyrus. The part XMI document of the class diagram is specified as follow:

<packagedElement xmi:type="uml:Class" xmi:id="_classsignal" name="Signal">

<packagedElement xmi:type="uml:Class" xmi:id="_classaircraft" name="Aircraft" clientDependency="_relationRealization _relationUsage _intReal ">

<ownedAttribute xmi:id="_j-C0EB" name="signal" type="_classSignal" isUnique="false" association="_assocaAS">

<packagedElement xmi:type="uml:Association" xmi:id="_assocaAS" name="A_Aircraft_Signal" >

From the XMI document, it shows that the class name and the relationship name are recorded in the *name* attribute. The element type of the *Aircraft* RtUnit class and the *Signal* PpUnit class is *uml*:*Class*. The element type of dependence relationship is *uml*:*Association*. The *Aircraft* class shares the information provided by *Signal* class through *A_Aircraft_Signal* association class, so the *Signal* class becomes an attribute of the *Aircraft* class that is realized by *_assocaAS* association class. Where its value is *signal*, and its type is *_classSignal*.

As Papyrus tool supports to derive model diagram in the form of XMI, we can analyze and transform class diagram in XMI form after building MARTE model.

Figure 12 shows the part of XMI document in Figure 11 class diagram.

```
<?xml version="1.0" encoding="UTF-8"?>
<xmi:XMI xmi:version="2.1"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:HLAM="http://MARTE.MARTE_DesignModel/schemas/HLAM/_A4jPMJ1HEd6zgsjJ8hkrLg/13"
>
 <uml:Model xmi:id="_v3Amw" name="marteModel">
   <packagedElement xmi:type="uml:Interface" xmi:id="_interAirser" name="AircraftService">
       <ownedOperation xmi:id="_Bs8gM" name="ready"/>
       <ownedOperation xmi:id="_FR08M" name="close"/>
  </packagedElement>
  <packagedElement xmi:type="uml:Class" xmi:id="_classaircraft" name="Aircraft"
        clientDependency=" _relationRealization _relationUsage _intReal ">
      <generalization xmi:id="_3Qlws" general="_interAirser"/>
      <ownedAttribute xmi:id="_ssvfA" name="a" isUnique="false">
          <type xmi:type="uml:PrimitiveType" href="pathmap:#Integer"/>
      </ownedAttribute>
          ……
      <interfaceRealization xmi:id="_intReal" name="Realization of AircraftService"
            supplier="_interAirser" client="_classaircraft" contract="_interAirser"/>
  </packagedElement>
 </uml:Model>
   ……
 <HLAM:RtUnit xmi:id="_h0olc" base_BehavioredClassifier="_classaircraft"/>
   ……
</xmi:XMI>
```

Figure 12. XMI document of Model class diagram

5.2 Realizing transformation

In term of the proposed transformation framework, we base on XMI and XSLT to transform XMI document of MARTE into XMI document of Object-Z schema. The result is showed as Figure 13.

```
<?xml version="1.0" encoding="UTF-8"?>
<classdef  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="marteOZSchema.xsd">
      <name>Aircraft</name>
      <inhertit> AircraftService</inhertit>
      <state>
         <name>signal_modeS</name>:<type>Integer</type>
         <name> signal_p</name>:<type>Integer</type>
         <name> signal_v</name>:<type>Integer</type>
         <name>a</name>:<type>Integer</type>
         <name>lastSense</name>:<type>String</type>
         <name>lastStrengh</name>:<type>String</type>
         ……
      </state>
      <init> </init>
      <op>
         <type>Boolean</type>
         <name>ready</name>
         <parameter>
             <name> </name>
             <type> </type>
         </parameter>
         ……
      </op>
</classdef>
```

Figure 13. Transformed result

After transforming the different XMI document, we can utilize Object-Z schema to check out the syntax of the transformed structure, and also use the existing verification tool to analyze and verify the XMI document of Object-Z(Derrick et al,2008). In term of the analysis result, we can feedback it to design model and modify MARTE model at the graphical interface. The verification process would not be discussed in this paper, but the verification result is very useful for development engineers to amend the fault of design model at the design stage of software life cycle.

6. Conclusion

At the different stages of the software life cycle, the targets of software model are different. At the design stage, MARTE model is used to informal communicate among software development engineers, the advantage is visibility. At the transformation stage, we use parser and processor to transform MARTE model into formal model, such as Objcet-Z. The advantage is precision. At the analysis stage, we can accurately and formally verify Object-Z model by verification tool, the advantage is automaticity. The verification result can be feedback and be used to guide the software engineers to perfect the design model.

We incorporate the advantages of MARTE model with formal model, and propose an integration formal model which combines Object-Z with PTA. This method can transform MARTE model into formal PTA-OZ model, so we can use the verification tools to analyze and verify the structure semantic and behavior semantic of the MARTE model at the design stage. How to realize the transformation is the focus of our research. Under MDA architecture, we

use XMI document of Object-Z model and XML schema document of Object-Z meta model to achieve the executable transformation framework from MARTE model to PTA-OZ model. So at the design stage, we can utilize different model to show and provide variable information for software development engineers.

Acknowledgements:

This work was supported by Funding of Jiangsu Innovation Program for Graduate Education and the Fundamental Re-search Funds for the Central Universities under Grant No.CXLX12_0161.

Corresponding Author:
Haiyang Xu
PO BOX 296
Nanjing University of Aeronautics and Astronautics
No.29 Yudao Street, Qinhuai District,
Nanjing City 210016
Jiangsu Province, China
E-mail: xhy@nuaa.edu.cn

References

[1]. Mallet, F., André, C. On the semantics of UML/MARTE clock constraints. IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing. 2009; 305-312.

[2]. André, C., Mallet, F. Specification and verification of time requirements with CCSL and Esterel. ACM Sigplan Notices. 2009;167-176.

[3]. Vidal, J., de Lamotte, F., Gogniat, G. et al. A co-design approach for embedded system modeling and code generation with UML and MARTE. 2009 Design, Automation & Test in Europe Conference & Exhibition. 2009;226-231.

[4]. Lecomte, S., Guillouard, S., Moy, C. et al. A co-design methodology based on model driven architecture for real time embedded systems. Mathematical and Computer Modelling. 2011; 53(3): 471-484.

[5]. Yu, H., Gamatié, A., Rutten, É. et al. Safe design of high-performance embedded systems in an MDE framework. Innovations in Systems and Software Engineering. 2008; 4(3): 215-222.

[6]. Sun, J., Dong, J. S., Liu, J. et al. Object-Z web environment and projections to UML. Proceedings of the 10th international conference on World Wide Web. 2001; 725-734.

[7]. Khan, S. A., Hashmi, A. A., Alhumaidan, F. et al. Semantic Web Specification using Z-Notation. Life Science Journal. 2012; 9(4): 994-1000.

[8]. Grose, T. J., Doney, G. C., Brodsky, S. A. Mastering Xmi: Java Programming with Xmi, XML and UML. John Wiley and Sons; 2002.

[9]. Faugère, M., Bourbeau, T., Simone, R. D. et al. MARTE Also an UML Profile for Modeling AADL Applications. ICECCS. 2007; 359-364.

[10]. Quadri, I. R., Gamatié, A., Boulet, P. et al. Expressing embedded systems configurations at high abstraction levels with UML MARTE profile: Advantages, limitations and alternatives. Journal of Systems Architecture. 2012; 58(5): 178-194.

[11]. Zhang, t., JOUAULT, F., ATTIOGBÉ, C. et al. MDE-Based Mode Transformation:From MARTE model to FIACRE Model. Journal of Software. 2009; 20(2): 214-233.

[12]. Möller, M., Olderog, E.-R., Rasch, H. et al. Integrating a formal method into a software engineering process with UML and Java. Formal Aspects of Computing. 2008; 20(2): 161-204.

[13]. Malik, P., Utting, M. CZT: A framework for Z tools. Formal Specification and Development in Z and B. 2005; 65-84.

[14]. Kim, S.-K., Carrington, D. A formal metamodeling approach to a transformation between the UML state machine and object-Z. ICFEM 2002, LNCS 2495. 2002;548-560.

[15]. Derrick, J., North, S., Simons, A. J. Z2SAL-building a model checker for Z. Abstract State Machines, B and Z. 2008;280-293.

2/25/2021