A Formal Transformation Approach for Embedded Software Modeling

Haiyang Xu
Department of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing
Department of Science and Information Science, Qingdao Agricultural University, Qingdao, China
Email: xhy@nuaa.edu.cn

Yi Zhuang
Department of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China
Email: zhuangyi@nuaa.edu.cn

***Abstract*—Formal specification can enhance the reliability of the embedded system and verify the system properties at the design stage. This paper presents a formal transformation approach for MARTE(Modeling and Analysis of Real Time and Embedded systems) model based on MDA(Model Driven Architecture), and defines the transformation rules of static and dynamic semantic between MARTE model and Object-Z model in term of the formal meta-model. The approach can produce a precise specification and verify the correctness of the system properties before implementing. The paper reports a case study to illustrate formal transformation of MARTE model. It demonstrates that the approach improves the accuracy of system model by transforming it into formal specification and enhances the reliability of software system.**

***Index Terms*—Reliability, Meta-Model, Formal modeling, Formal transformation rules**

I. Introduction

With the development of electronic technology, embedded system has been widely used in control, communication, aerospace and other fields. As the increasing heterogeneity and complexity of system components, the development of embedded system was confronted with major difficulties[1]. In order to enhance the system reliability, we need to develop an effective design method to provide language and tool support for embedded system. So we could analyze and predict the reliability and safety performance of system in the early stage of system design.

UML is a standard object-oriented modeling language, it uses graphical notion to model the static and dynamic features of the system[2]. The Object Management Group (OMG) proposes to build the domain-specific modeling language by refining existing modeling language. MARTE is a new UML profile [3], it supports real-time and embedded systems specification. In the functional design, it adds model packages to describe the hardware and software resources, and defines a specific set of attributes for designer, which could be used to analyze the time and energy performance in the early design stage. For the non-functional requirement, it adds the definition of clock. But MARTE is short of precise semantic, and cannot describe and analyze in the strict formal way. So it cannot be used to automatic reasoning the properties of the system.

Formal specification lay a solid foundation for the syntax and semantic of UML/MARTE modeling language, and provide a precise reasoning mechanism[4]. It can transform design model of system into reliability analytical model by formal method. So it can find out the design fault in the design stage, and facilitate the system reliability.

The paper advances an Object-Z meta-model by analyzing MARTE and Object-Z, which is the foundation of model transformation. And we define a transformation framework from MARTE to Object-Z based on MDA. In the framework, we propose the transformation rules of the static and dynamic semantic between MARTE and Object-Z. In particular, according to the Object-Z specification and its reasoning technology, in the design stage, we could formally verify the system models which are built using MARTE. This method provides a general transformation framework, which can transform MARTE model into Object-Z model. For the characteristic of embedded system, the paper focuses mainly on reliability. The paper reports a case study involving the formal specification of Josefil, and illustrates the transformation from MARTE model to Object-Z model.

The rest of this paper is organized as follows. Section II outlines related work and associated problems. Section III describes the transformation framework based on MDA. Section IV gives the detailed semantic transformation rules of our approach at meta-model level. Section V reports a case study of the formal specification. Finally, Section VI concludes the paper.

II. Related Work

In the model-driven framework, there are two questions needed to be solved. The first is modeling the software model correctly; the second is the valid transformation between heterogeneous models. For the first question, it uses UML and its extensions as modeling

doi:10.4304/jsw.9.4.807-813

language. For the transformation, it is still being researched.

Kim Soon-Kyeong et al. [6,7] present a formal description for UML class constructs, and a formal meta-modeling approach to transform UML to Object-Z. They utilize MDA for automated transformations between models defined in different languages at different abstraction levels [8].

Ahmed M.Mostafa et al. [9] use Z specification to increase the safety and reliability of the system, and present a set of formalism models for Use Case diagram, Class diagram, and State Machine diagram, which transforms a UML model to Z specifications. In the integrated framework, it can achieve a model that is unambiguous, verifiable and traceable.

Michael Möller et al. [10] describe a formal method CSP-OZ, which combines the process algebra CSP with the specification language Object-Z. It can verify properties of models in the early design phases and check adherence of implementations to models.

Huascar Espinoza et al. [11] use UML as the base language, and combine the SysML and MARTE profiles in a common modeling framework. The framework can specify embedded systems at different abstraction levels, and provide a convergence and alignment program for the two respective technologies. The integration strategy offers a better understanding of their conceptual domains, and helps avoiding semantic and syntactical mismatches.

Simona Bernardi et al. [12] propose a dependability analysis and modeling profile for MARTE models to reuse proposals that derive dependability models from UML annotated specifications and to be compliant with the recently adopted MARTE profile. They also propose to add to MARTE profile for dependability analysis and modeling (DAM), and use MARTE-DAM profile to derive a stochastic Petri net model for performance and dependability assessment [13].

Zhang Tian et al. [5] present a representative case study of bridging MARTE to FIACRE. They develop the semantic mapping rules to implement the transformation between meta-models and build the concrete syntax rules to generate the textual programs.

Ling Yin et al. [14] propose a formal state-based interpretation of fundamental subset of the Clock Constraint Specification Language (CCSL) clock constraints to verify CCSL specifications. Through the formalism, they translate a CCSL specification into a Promela model and check the correctness of model by model checker SPIN.

Some recent research efforts have adapted formal method for UML, and also transformed MARTE into other embedded system modeling languages, or transformed some parts of or the whole MARTE into a formal model. So we can validate the correctness of MARTE model using existing checker tools for formal model. But there is little research on transforming MARTE model into Z or Object-Z specification. In our approach, we define a general transformation framework, and transform the static and dynamic sematic of MARTE into Object-Z formal model.

III. THE MDA-BASED TRANSFORMATION FRAMEWORK

There are two core questions that need to be solved: formal modeling and formal model transformation. For formal modeling, we define modeling notation based on Object-Z meta-model. For the latter, we need to understand the abstract syntax and semantic between source model and target model. We define formal functions between MARTE and Object-Z, and present transformation rules from informal modeling language to formal ones to implement the heterogeneous models transformation. All the work is under the MDA framework.

A. MDA

MDA was launched by the Object Management Group (OMG) in 2001. It bases on models and the automatic mappings or transformations between model and code. It is a new software design approach, and changes the traditional code-centric software development method. There are two abstract models in MDA, PIM and PSM. Basic MDA model include defining system functionality using a PIM, and automatic mapping from PIM to one or more PSMs.

The MDA approach has several advantages. Firstly, the separation from domain-specific language to concrete implementation improves the portability of the model. Secondly, the automatic mapping between PIM and PSM increases the productivity. Thirdly, the proved well-formed patterns are used to model, and improve the quality of model. Fourthly, the sufficiently independent relationship between PIM and PSM enhances the maintainability of the software system. Finally, the consistency and the traceability are also defined and realized.

Model transformation is an important part of MDA. If the MDA model is related to OMG QVT standards, it can increase the application experience of model transformation in different fields. Lots of tools have been developed to interpret, verify, and transform models or meta-models, like Compuware OptimalJ, Borland Together, AndroMDA.

B. Model Transformation Framework

Object-Z model is at the model level (M1-level). While the definition of the formal model is the abstract of itself, formal model needs to obey the definition to describe. So the definition is at the meta-model level (M2-level). Essentially the formal modeling process is the instantiation process of formal meta-model at the M2-level, that is, the formal method is used to concrete PIM modeling [15]. The foundation of formal definition is mathematical model, so it must obey to mathematic axioms. We manage to identify the interoperability between formal model and MARTE, and build them into the same architecture. The case is feasible and brings certain benefits.

The meta-model of MARTE formal model is at the M2-level of MOF's [16] hierarchy and it is the pattern specification which implements MARTE formal modeling and transforming.

Fig.1 shows the model transformation process between MARTE model and Object-Z model under the meta-to-meta model architecture. And the processes are:

–We need to build the meta-model of Object-Z, and make sure it can be operated in meta-to-meta model architecture.

–It is necessary to define the mapping function from MARTE meta-model to Object-Z meta-model.

–We implement the transformation at model level using the mapping rules defined at the meta-model level.

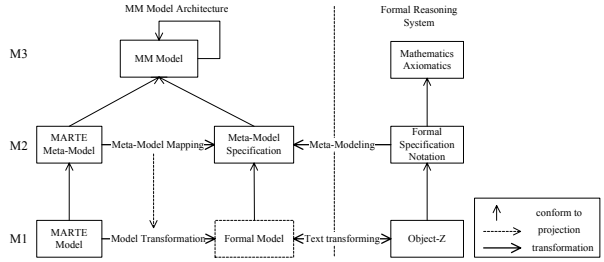


Figure.1 Transformation relationship

C. Object-Z Meta-model

Object-Z uses the Object-Oriented (OO) paradigm, that is similar to UML, and their underlying concepts are also the same. For example, an Object-Z class can encapsulates its attributes and operations. Using the OO formal specification has fewer complex problems than using the non-OO formal specification in the transformation process. Semantically, a class maps to a set of object identifiers [17].

We use class diagram to describe the structural features of Object-Z core model elements[7], as show in Fig.2. OZModelElement is the most top meta-class, and it can be instantiated into concrete classes. OZClass, OZParameter, OZSpecification, OZPredicate and OZFeature inherit from OZModelElement. The Object-Z class OZClass composes of OZAttribute, OZOperation and Invariant et al.

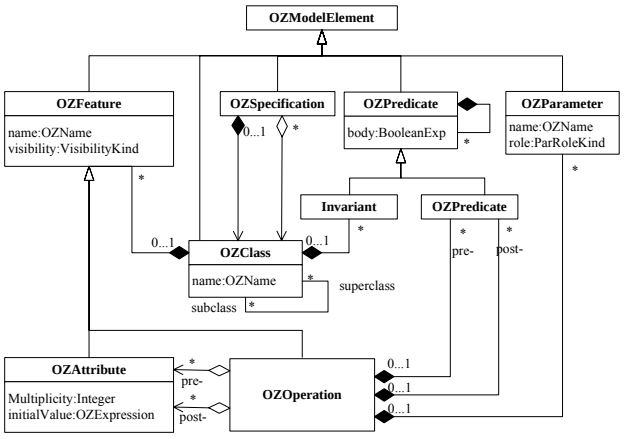


Figure.2 Object-Z core model elements

Meta-Class OZPredicate is used to define the condition of class or operation. The condition defined in class is invariant. The condition defined in operation is either the pre-condition, or the post-condition. OZAttribute and OZOperation define static and dynamic behaviors of the class instances. The semantic details of Object-Z can refer to literature [12].

IV. The Semantic Transformation at Meta-Model Level

A. Abstract Syntax of MARTE

We first define the domain of models in an abstract syntax using Graphical Extension of Backus Normal Form (GEBNF)[18], and then use it to define some transformation rules of class and association. A MARTE class is composed of name, attributes, and operations. An attribute has name, type, multiplicity and other tags. An operation has name, parameters, and other tags.

Definition 1. (MARTE class's abstract syntax)
Class::=name:String,[attributes:Property*],[operations: Operation*]
Property::=name: String, type:Type, visibility: [VisibilityKind], multiplicity:[Multiplicity]
Operation::=name:String,parameters:[Parameter*], visibility:[VisibilityKind],isAbstract:[Bool], isQuery:[Bool],isLeaf:[Bool],isNew:[Bool]
Parameter::=name:[String],type:[Type],direction: [ParaDirKind],multiplicity:[Multiplicity]
VisibilityKind::=private|public|protected
Multiplicity::=lower:[Natural],upper:[Natural|*]
ParaDirKind::=in|inout|out|return
Definition 2. (Association's abstract syntax)
MARTEAssociation::=name:String, vars:AssociationEnd^{+}
AssociationEnd::= name:String, attachedClass: Class^{+}, multiplicity:[Multiplicity], aggregation: [AggregationKind],navigability:Boolean
AggregationKind::=none|aggregate|composite
Definition 3. (State's abstract syntax)
State::=name:String, attributes:Property^{+}, associations: MARTEAssociation*
Definition 4. (Actions's abstract syntax)
Actions::=name:String, operations:Operation Kind, isAsnchronous:Boolean
OperationKind::=entry|exit|doActivity|effect:Operation
Definition 5. (Event's abstract syntax)
Event::=name:String, attributes:Property*,
Definition 6. (Transition's abstract syntax)
Transition::=name:String,source: State,target: State, trigger:Event,guard: State,effect:Action
Definition 7. (StateMachine's abstract syntax)
StateMachine::=states:State,transitions:Transition, $state_{top}$:State

B. Static Semantic Transforming

In MDA, the semantic and abstract syntax of a language are defined in terms of its meta-model [5]. According to the meta-model of MARTE and Object-Z, we propose transformation rules in the context of class and class association to show the static and dynamic sematic relationships between heterogeneous model built by its respective meta-model[7, 10].

Semantically, MARTE class is an abstract type, and it can be instantiated to all possible objects. So every MARTE class will be transformed into an Object-Z class.

We define a function *mapMARTEClassToOZ* to formally describe the transformation rules from a

MARTE class to an Object-Z class. This function takes a MARTE class and returns a corresponding Object-Z class

Rule 1. (Class transformation rule) The MARTE class name is used as the corresponding Object-Z class name. Each attribute in the MARTE class is declared as an attribute in Object-Z class. Each operation is transformed into an operation schema:

$mapMARTEClassToOZ$: $MARTEClass \rightarrow \mathbb{P} OZClass$

$\forall mc$: $MARTEClass \bullet mapMARTEClassToOZ(mc) =$

$\{oc:OZClass \mid mc.name = oc.name \wedge$

$\forall ma:mc.attributes \bullet \exists oa:oc.attributes \bullet$

$oa.name = ma.name \wedge oa.multiplicity = ma.multiplicity \wedge$
$oa.type = convType(ma.type) \wedge oa.visibility = ma.visibility \wedge$
$oa.relationship = relNone \wedge oa.navigability = navNone$

$\forall mo:mc.operations \bullet \exists oo:oc.operations \bullet$

$oo.name = mo.name \wedge oo.visibility = uo.visibility \wedge$

$\forall mp:mo.parameters \bullet \exists op:oo.parameters \bullet op.name = mp.name \wedge op.type = convType(mp.type)\}$

The attributes types and operation parameters are all language-dependent specification in MARTE, but those in Object-Z are language-independent. So we utilize an abstract function *convType* that builds the mapping relationship between MARTE type and Object-Z type [7].

Objects of the class can communicate with each other via the link provided by an association. There are three types of association: one-to-one, one-to-many, many-to-many. In Object-Z, instantiation mechanism is used to build the communication between objects. So no matter which kind of association, it can be instantiated within other classes as an attribute.

Rule 2. (Association transformation rule) For every association, if one association end is navigable to the opposite end, then the opposite MARTE class transforms into an Object-Z class, and the association is instantiated as an attribute within this Object-Z class.

$mapAssocToOZ$:$MARTEAssociation \rightarrow \mathbb{P}(OZClass \times OZClass)$

$\forall ma:MARTEAssociation \bullet mapAssocToOZ(ma) =$
$\{ soc,toc:OZClass \mid$

$soc \in mapMARTEClassToOZ(ma.e1.attachedClass) \wedge$

$toc \in mapMARTEClassToOZ(ma.e2.attachedClass) \wedge$

$ma.e1.navigability = true \Rightarrow \exists ta:toc.attributes \bullet$

$ta.name = ma.e1.name \wedge ta.type = powerT(soc) \wedge$

$ta.multiplicity = ma.e1.multiplicity \wedge ta.navigability =$

$convNavigability(ma.e2.navigability) \wedge ta.relationship =$
$convRelationship(ma.e2.aggregation, ma.e1.aggregation)$

$ma.e2.navigability = true \Rightarrow \exists sa:soc.attributes \bullet$

$sa.name = ma.e2.name \wedge sa.type = powerT(toc) \wedge$

$sa.multiplicity = ma.e2.multiplicity \wedge sa.navigability =$

$convNavigability(ma.e1.navigability) \wedge sa.relationship =$
$convRelationship(ma.e1.aggregation, ma.e2.aggregation)$

C. Dynamic Semantic Transforming

The value of attribute represents the object state, and it can be changed by its operation. It defines the object behavior in term of the attributes and operations. The state machine is similar to the object behavior in Object-Z semantically, so it is feasible to use a state machine to represent the object behavior [7, 8]. We define the mapping rules between MARTE state machine and Object-Z as follow.

Rule 3. (State transformation rule) A state is central to the description of a system, and represents as an assignment to each variable at a certain moment during the period of lifetime. To provide the syntactical mapping between the two models, we define a mapping rule from a state to the attributes of Object-Z class. Detailed transformation rule is implemented through function *mapStateToOZ*, which takes a state and returns attributes of Object-Z class.

$mapStateToOZ$:$State \rightarrow \mathbb{P}\ OZAttribute$

$\forall ms:State \bullet mapStateToOZ\ (ms) = \{oa:OZAttribute \mid$

$\exists oc:OZClass \bullet oa \in oc.attributes \wedge ms.name = oa.name \wedge$

$ms.type = oa.type \wedge ms.visibility = oa.visibility\}$

Rule 4. (Event transformation rule) A event can be classified as internal event and external event, examples include receipting a request or invoking an operation. So we map every event of state machine to an Object-Z operation, each parameters of the event maps to the parameters of the corresponding operation. The function *mapEventToOZ* is used to describe this formal mapping rule.

$mapEventToOZ$:$Event \rightarrow OZOperation$

$\forall me:Event \bullet mapEventToOZ(me) = \{oo:OZOperation \mid$

$\exists oc:OZClass \bullet oo \in oc.operations \wedge$

$(\forall mp:me.parameters \bullet \exists op:oo.parameters \bullet$

$op.name = mp.name \wedge op.type = convType(mp.type))\}$

Rule 5. (Actions transformation rule) Action is a named element which represents a single atomic step within activity, and is the fundamental unit of executable processing or behavior. So we define a transformation rule that maps an action to an operation in Object-Z. The function *mapActionToOZ* takes an action and returns an operation.

$mapActionToOZ$: $Actions \rightarrow OZOperation$

$\forall ma:Action \bullet mapActionToOZ(ma) = \{oo:OZOperation \mid$

$\exists oc:OZClass \bullet oo \in oc.operations \wedge ma.name = oo.name \wedge$
$ma.visibility = oo.visibility$

Rule 6. (Transition transformation rule) A transition takes operation from source state to target state, and represents the response to a particular event. It can connect states with transitions and create internal transitions within states. So we transform transition into corresponding Object-Z operation, and the states of two end map to the attributes of Object-Z class. The rule is defined in the function *mapTransitionToOZ*.

$mapTransitionToOZ$:$Transition \rightarrow \mathbb{P}\ OZOperation$

$\forall mt:Transition \bullet mapTransitionToOZ(mt) =$

$\{oo:OZOperation \mid$

$(\exists oc:OZClass \bullet oo \in oc.operations \wedge mt.name = oo.name)$

$(\exists source,target,attrs:oc.attributes \bullet source = mapStateToOZ(mt.source) \wedge target = mapStateToOZ(mt.target) \wedge attrs = mapStateToOZ(mt.guard))$

$(\exists op1,op2:oc.operation \bullet op1 = mapEventToOZ(mt.trigger) \wedge op2 = mapActionToOZ(mt.effect))$

Rule 7. (State Machine transformation rule) In term of the defined transformation rules, we can map state machine to Object-Z class. That is, we transform all of the states, events, actions, transitions into the attributes

and operations of corresponding Object-Z class. The function *mapStateMachineToOZ* formalizes the detailed transformation rule.

mapStateMachineToOZ:*StateMachine*→*OZClass*
∀*sm*:*StateMachine*•*mapStateMachineToOZ*(*sm*)={
∃*oc*:*OZClass*|(∀*st*:*sm.states*|*st*∈↓*State*•
(∃*ba*:*oc.attributes*|*ba*∈*OZAttribute*•*mapStateToOZ*(*st*)=*ba*))∧
(∃*ep*:*oc.operations*|*ep*∈↓*OZOperation*•{*ep*}={*e*:*st.entry*•*mapActionToOZ*(*e*)})∧
(∃*xp*:*oc.operations*|*xp*∈↓*OZOperation*•{*xp*}={*e*:*st.exit*•*mapActionToOZ* (*e*)})∧
(∃*ap*:*oc.operations*|*ap*∈↓*OZOperation*•{*ap*}={*e*:*st.doActivity*•*mapActionToOZ*(*e*)})
(∀*t*:*sm.transition*•(∃*tp*:*oc.operations*|*tp*∈*OZOperation*•*tp*=*mapTransitionToOZ*(*t*)∧
tp.source∈ {*ba*:*oc.attributes*|*ba*∈*OZAttribute*}∧
tp.target⊆{*ba*:*oc.attributes*|*ba*∈*OZAttribute*}∧
tp.stateEntry⊆{*ap*:*oc.operations*|*ap*∈↓*OZOperation*}∧
tp.stateExit⊆{*ap*:*oc.operations*|*ap*∈↓*OZOperation*}∧
tp.stateActivity⊆{*ap*:*oc.operations*|*ap*∈↓*OZOperation*}∧
(∀*e*:*t.effect*•*mapActionToOZ*(*e*)∈↓*OZOperation*)∧
(∀*e*:*t.trigger*•(∃*ep*:*oc.operations*|*ep*∈↓*OZOperation*•*ep*=*mapEventToOZ*(*e*)))

V. CASE STUDY

The ability to formally specify the system model is the main goal of this paper. We take a case study of Josefil to implement the transformation from MARTE model to Object-Z model. The Josefil case study is proposed by Ensieta, and it consists in developing a robotics system following model-driven engineering practices. Its goal is to provide a framework for testing the MDA approaches in case of specification changes to distributed real-time and embedded systems [19].

The case study introduces a control system for an exploration robot. It consists of a robot and a remote control. The robot communicates with a remote control station, when it receives a position to reach transmitted by the control station, it moves to this position, and waits for a new route.

A.Semantic Transformation

To enhance the reliability of the embedded system, we translate MARTE model into formal model in the design stage. For the sake of brevity, we choose two logical entities of the Josefil case study: the system supervision (*Supervisor*) and the management of communications with the control station (*CommunicationLink*) (As show in Fig.3).

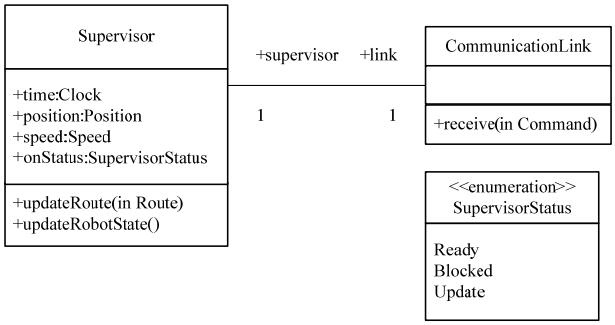


Figure.3 The class diagram of system supervision

Firstly, we translate the static semantic of the class diagram. As discussed in Section IV. We translate the Supervisor class into an Object-Z class, the class name, attributes, and operations of MARTE model are mapped to the class name, attribute, and operations of the Object-Z class. As class Supervisor is associated with class CommunicationLink, which is transformed as an attribute in the corresponding Object-Z class. Detailed result is below(Fig.4).

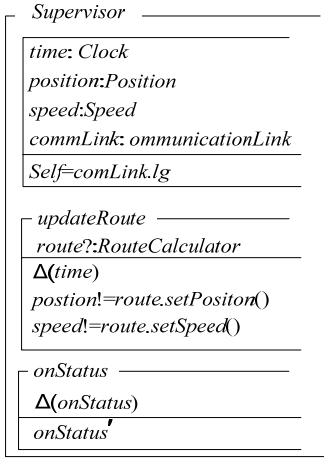


Figure.4 Static transformation results

Secondly, we deal with the dynamic semantic. According to states and transitions between states, state machine can be used to describe the behavior of a system. In state machine, transition is triggered by time. We define the duration of a behavior using stereotype «timedProcessing», and apply it on the updateTimeOut activity. The duration of this time processing is (50, ms).

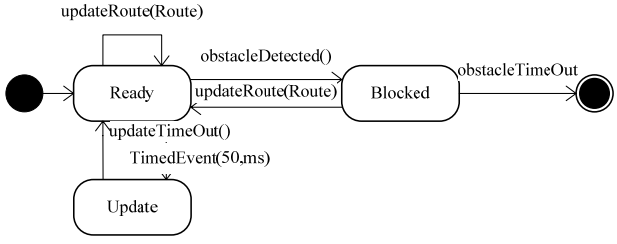


Figure.5 State diagram of Supervisor

Fig.5. shows a state machine of Supervisor class, and it provides a simple view to describe the behavior of this class. The Supervisor state machine has three states: Ready, Update, Blocked. Ready state periodically enters the Update state for updating the robot sensors values. This process has to be done within a period of 50 ms.

Then it returns to a Ready state. If a sensor detects an obstacle, the Supervisor enters the Blocked state. From there, the operator has 50 seconds to enter a new route. If no route is received within this time frame, a time out is triggered [19].

According to the defined dynamic semantic transformation rules, the states, events, actions, and transitions can be transformed into the attributes and operations of corresponding Object-Z class. Thus, we have the following schema(Fig.6).

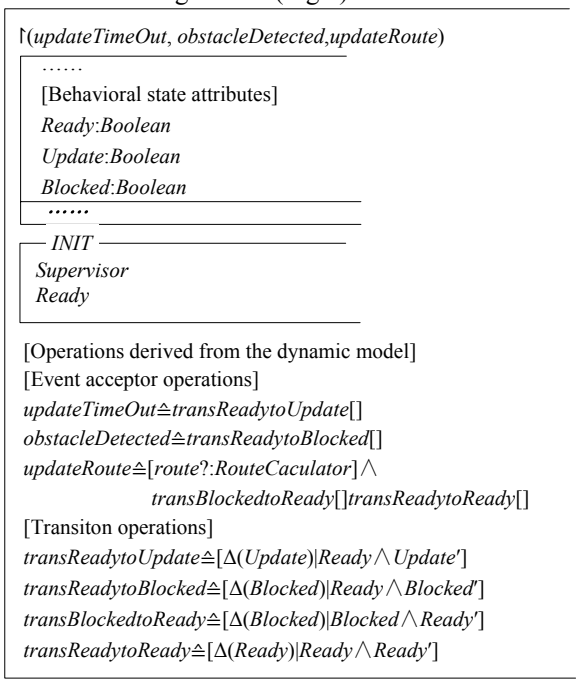


Figure.6 dynamic transformation results

B. Analysis

We focus on two logical entities of the Josefil case study, and present how to formally transform the static and dynamic semantic of a model. The transformation from a MARTE model to an Object-Z model enables the target model to be more precise and to be strictly analyzed. So we can use the existing Object-Z analysis tools to verify the correctness of the model, and make sure the consistency of the model and specification. Finally, it can improve the software reliability of the system.

VI. CONCLUSIONS

Comparing with the traditional computer system, the reliability is more important for embedded system. This motivates researchers to develop well-formed methods and tools to model and analyze the embedded system. Formal specification is defined by strict mathematical notation and can specify the system properties. It enables us to find the inconsistency in the design stage. In the paper, we provide a set of formal transformation rules for MARTE profile. Our main contributions here are: (a) the MDA-based transformation framework between MARTE model and Object-Z model, (b) the static semantic transformation rules, (c) the dynamic semantic transformation rules.

The MARTE profile also adds constructs to provide the Non-Functional Properties (NFPs) modeling framework (e.g., throughputs, delays, memory usage). For the future work, we will further develop the NFPs formal transformation.

ACKNOWLEDGMENT

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

REFERENCES

[1] Yingjie Song, Rong Chen, Yaqing Liu. A Non-Standard Approach for the OWL Ontologies Checking and Reasoning. Journal of Computers. vol. 7, no.10, pp.2454-2461,2012.

[2] Chunyu Miao. Dynamic Slicing Research of UML Statechart Specifications. Journal of Computers. vol. 6, no.4, pp. 792-798,2011.

[3] UML Profile for MARTE: Modeling and Analysis of Real-time Embedded Systems. 2011 http://www.omg.org/spec/MARTE/1.1.

[4] Lina Chen. Automatic test cases generation for statechart specifications from semantics to algorithm. Journal of Computers. vol. 6, no.4, pp. 769-775,2011.

[5] Zhang Tian, Jouault F,et al. “MDE-Based Model Transformation: From MARTE Model to FIACRE Model”. Journal of Software, vol. 20, no.2, pp.214-233,2009.

[6] Kim S-K, Carrington D. “A Formal Mapping between UML Models and Object-Z Specifications”, In Proceeding(s) of LNCS 1878, pp. 2-21,2000.

[7] Kim S-K, Carrington D. “A formal metamodeling approach to a transformation between the UML state machine and object-Z”, In Proceeding(s) of ICFEM 2002, LNCS 2495,pp.548-560,2002.

[8] Kim S-K, Burger D, Carrington D. “An MDA Approach towards Integrating Formal and Informal Modeling Languages”, Formal Method, LNCS 3582,pp.448-464, 2005.

[9] M.Mostafa A, Ismail MA, EL-Bolok H et al. “Toward a formalization of uml2. 0 metamodel using z specifications”, In Proceeding(s) of 8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, pp.694-701,2007.

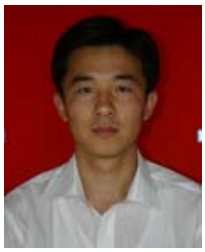
[10] Möller M, Olderog E-R, Rasch H et al. “Integrating a formal method into a software engineering process with UML and Java”, Formal Aspects of Computing,vol. 20,no.2,pp.161-204,2008.

[11] Espinoza H, Cancila D, Selic B et al. “Challenges in Combining SysML and MARTE for Model-Based Design of Embedded Systems”, In Proceeding(s) of the 5th European Conference on Model Driven Architecture - Foundations and Applications, LNCS 5562,pp.98-113, 2009.

[12] Bernardi S, Merseguer Je, Petriu DC. “Adding Dependability Analysis Capabilities to the MARTE Profile” , In Proceeding(s) of MoDELS 2008, LNCS 5301,pp.736-750,2008.

[13] Bernardi S, Merseguer J, Petriu DC. “A dependability profile within MARTE”,Software and Systems Modeling, no.10,pp. 313-336, 2011.

[14] Yin L, Mallet F. "Verification of MARTE/CCSL Time Requirements in Promela/SPIN", In Proceeding(s) of ICECCS,pp.65-74,2011.
[15] Liu Yaping, Huang Zhiqiu, Zhu Yi. "Research on Model Transformation Method of Real-time System Based on Metamodeling", Journal of Chinese Computer Systems, vol.31,no.11,pp.2146-2153, 2010.
[16] Meta Object Facility Specification, Version 2.0. http://www.omg.org/spec/MOF/2.4/Beta2/PDF/
[17] Kim S-K, Carrington D. "Formalizing the UML Class Diagram Using Object-Z", In Proceeding(s) of UML'99, LNCS 1723,1999.
[18] Bayley I, Zhu H. "Formal specification of the variants and behavioural features of design patterns", The Journal of Systems and Software, no.83,pp.209-221,2010.
[19] Demathieu S, Thomas F, André C et al. "First experiments using the UML profile for MARTE", In Proceeding(s) of 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing, pp.50-57,2008.



Haiyang Xu, was born in WeiHai, Shandong Province, China in 1981. He received the M.S. degree in Computer Application Technology from Capital Normal University in 2006, and he is currently a Ph. D. candidate of Nanjing University of Aeronautics and Astronautics.

He has been a faculty member of Software Engineering at Qingdao Agriculture University, Qingdao, China, since 2006, where he is currently a lecturer. His research interests include trusted computing, model transformation, model checking.

Yi Zhuang, was born in Juangsu Province, China in 1957. She is a professor and Ph. D. supervisor of Nanjing University of Aeronautics and Astronautics. Her research interests include distributed computation, information security, trusted computing and model migration et al.