Systematic Transformation Method from UML to Event-B

Xue Geng, Sheng-rong Zou*, and Ju-yi Yao
College of Information Engineering, Yangzhou University, Yangzhou, China
1964405987@qq.com, srzou@qq.com, 1524396696@qq.com
*corresponding author

***Abstract*—In object-oriented software development, UML has become a de facto modeling standard. However, although UML is easy to understand and apply, it has inaccurate semantics, and UML is a semi-formal modeling language, which cannot be formally verified. Event-B is a formal method based on a large number of mathematical predicate logic, which is precise but difficult to understand and apply. Therefore, how to combine the advantages of UML diagram and Event-B method is the focus of the research. The previous transformation methods are based on the transformation from UML scatter diagram to Event-B, which is prone to conflict and inconsistency. Therefore, we propose a systematic transformation method that can realize the corresponding unification of elements in UML and those in Event-B. The general software system is a medium-sized system. We believe that the medium-sized system can be clearly expressed by using use case diagram, class diagram, state diagram and sequence diagram. In this paper, the transformation methods from these four diagrams to Event-B are given respectively. The transformation method of the system is applied to the elevator control system which requires high safety and reliability. The system transformation method from UML to Event-B not only improves the accuracy of UML and is easy for software practitioners to use, but also enhances the comprehensibility of formal methods and is conducive to the promotion and application of formal methods.**

Keywords- UML, formal method, Event-B, Elevator, model

I. INTRODUCTION

UML is widely used in the software modeling industry to help developers understand the requirements of the system intuitively in a graphical way. In object-oriented software development, UML has become the de facto modeling standard. However, in fact, UML is a semi-formal modeling language, which is inaccurate and cannot be formally verified. Formal method is a special technology based on mathematics, which can be verified in systems with high security requirements. There are many researches on formal methods. For example, Z method, B method and Event-B method. B method is a popular and easy to use formal method. Event-B is the latest B method with accurate semantics. However, the Event-B formal method is based on a large number of mathematical logic predicates, which is difficult for software requirements analysts to understand and apply. Since UML is not accurate and Event-B method is not easy to understand, the combination of UML and Event-B has been a research topic.

II. REVIEW OF EXSITING RESEARCH

There have been some studies on the combination of UML and formal methods. For example, the paper evaluates the understandability of B model and UML-B model, UML-B model and Event-B model. The evaluation results show that the combination of semi-formal and formal methods promotes the participants' understanding of the model problem domain. However, the transformation methods mentioned above are based on the transformation from scattered individual diagrams of UML14 diagrams to Event-B methods, and there is no systematic transformation method. When transforming UML to Event-B methods using these scattered transformation methods, there may be inconsistent transformation problems and transformation conflicts, which are difficult to apply in actual software development process. Therefore, it is necessary to systematically study the transformation from UML to Event-B formal methods.

III. ABSTRACT TRANSFORMATION METHOD

The general software system is a medium-sized system with a code volume of 5000 to 50000 lines. This medium-sized system can be well expressed only by using UML use case diagram, class diagram, sequence diagram and state diagram. With the above four diagrams, the requirements acquisition, requirements analysis, design and detailed design of the software life cycle can be fully expressed. In order to systematically transform UML into Event-B method, we propose a transformation method from UML to Event-B based on the four UML diagrams mentioned above. In this paper, the transformation methods from these four diagrams to Event-B are given respectively.

A. Use Case Diagram

Some transformation methods from use case diagram to Event-B method have been proposed. Based on the above transformation methods, we improved the transformation method from use case diagram to Event-B method from the perspective of relationship, and realized a more comprehensive translation of the elements in the use case diagram. The actors and use cases in the use case diagram are transformed into constants in Event-B. The relationships in the use case diagram are transformed into events. In particular, generalization relationships between actors can be transformed into axioms in context. The transformation rules from use case diagram to Event-B are given in table I.

TABLE I. USECASE DIAGRAM TRANSFORMATION RULES

Use case diagram	Event-B	Part of Event-B
actor	constant	set
use case	constant	set
generalization	event/axiom	machine/context
association	event	machine
extend	event	machine
include	event	machine

B. *Class Diagram*

iUML-B is a graphical front end similar to UML, which is used to model the object-oriented concept of Event-B method, and supports the modeling of class diagram. Classes, properties, and relationships in class diagrams are associated with constants, variables, and other elements in Event-B. Methods in the classes are transformed into events in Event-B, and statements can be added to the action module of the event to represent the specific execution process of the method. The transformation rules from class diagram to Event-B are given in table II.

TABLE II. CLASS DIAGRAM TRANSFORMATION RULES

Class diagram	Event-B	Part of Event-B
class	set/constant/variable	context/set/machine
attribute	constant/variable	set/machine
method	event	machine
inheritance	constant/variable	set/machine
association	constant/variable	set/machine
aggregation	constant/variable	set/machine
composition	constant/variable	set/machine
dependency	constant/variable	set/machine
realization	constant/variable	set/machine

C. *Sequence Diagram*

The message and communication objects in the sequence diagram are transformed into constants, and the message delivery order in the sequence diagram is controlled by variables in Event-B machine. The specific transmission process of messages is transformed into events in Event-B. Table III shows the transformation rules from sequence diagram to Event-B method.

TABLE III. SEQUENCE DIAGRAM TRANSFORMATION RULES

Sequence diagram	Event-B	Part of Event-B
object	constant	set
message	constant	set
activation	action	event
lifeline	variable	event

D. *State Machine Diagram*

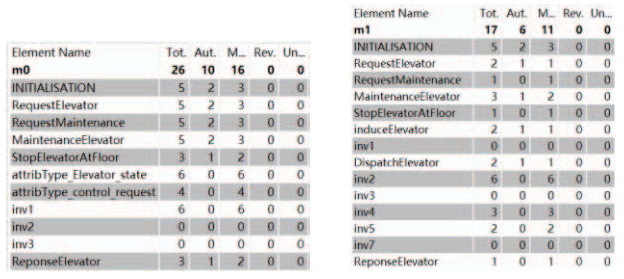
The state in the state diagram is transformed into a constant, the action, fork and join in the state diagram are transformed into actions, and the activity is transformed into an action module. Events that trigger state transitions in the state diagram are transformed into events in Event-B, transition and junction are transformed into guards. Table IV shows the transformation rules from state diagram to Event-B.

TABLE IV. STATE DIAGRAM TRANSFORMATION RULES

State diagram	Event-B	Part of Event-B
state machine	machine	machine
submachine state	constant	set
composite state	constant	set
activity	action module	event
action	action	event
event	event	machine
transition	guard & action	event
junction	guard	event
fork	action	event
join	action	event

IV. APPLICATION IN ELEVATOR SYSTEM

Elevator is a real-time hardware system that requires high reliability and security. Elevator is the most common example in the field of security [1]. A large number of scholars in the field of safety control often take elevator control system as a research example, and elevator is a medium-sized system, which is suitable for our proposed systematic transformation method [2]. The proof obligations generated in the elevator model are shown in Figure 1. Figure 1 (a) shows that the abstract machine m0 has generated 21 proof obligations, and Figure 1 (b) shows that the refined machine m1 has generated 22 proof obligations. These proof obligations have been discharged through automatic proof and interactive proof.



Element Name	Tot.	Aut.	M...	Rev.	Un...
m0	**26**	**10**	**16**	**0**	**0**
INITIALISATION	5	2	3	0	0
RequestElevator	5	2	3	0	0
RequestMaintenance	5	2	3	0	0
MaintenanceElevator	5	2	3	0	0
StopElevatorAtFloor	3	1	2	0	0
attribType_Elevator_state	6	0	6	0	0
attribType_control_request	4	0	4	0	0
inv1	6	0	6	0	0
inv2	0	0	0	0	0
inv3	0	0	0	0	0
ReponseElevator	3	1	2	0	0

(a)

Element Name	Tot.	Aut.	M...	Rev.	Un...
m1	**17**	**6**	**11**	**0**	**0**
INITIALISATION	5	2	3	0	0
RequestElevator	2	1	1	0	0
RequestMaintenance	1	0	1	0	0
MaintenanceElevator	3	1	2	0	0
StopElevatorAtFloor	1	0	1	0	0
induceElevator	2	1	1	0	0
inv1	0	0	0	0	0
DispatchElevator	2	1	1	0	0
inv2	6	0	6	0	0
inv3	0	0	0	0	0
inv4	3	0	3	0	0
inv5	2	0	2	0	0
inv7	0	0	0	0	0
ReponseElevator	1	0	1	0	0

(b)

Figure 1. (a) Proof obligations of abstract machine m0, (b)Proof obligations of refined machine m1

V. CONCLUSION

The system transformation method in this paper systematically transforms UML into Event-B formal method. The system transformation method from UML to Event-B not only improves the accuracy of UML and is easy for software practitioners to use, but also enhances the comprehensibility of formal methods and is conducive to the promotion and application of formal methods.

REFERENCES

[1] XL. Wang, et al.“Research on the Risk Points of Elevator Braking System,” CA: 2020 International conference on Artifical Intelligence and computer engineering, 2020.

[2] WX. Chen, et al. “A Real-Time Matrix Iterative Optimization Algorithm of Booking Elevator Group and Numerical Simulation Formed by Multi-Sensor Combination,” ELECTRONICS, vol. 10, no. 24, 2021.