UML 到 Event-B 的系统化转换方法

耿 雪,邹盛荣*,刘晓莹,姚聚义
(扬州大学 信息工程学院,江苏 扬州 225009)

摘 要:在面向对象的软件开发中,UML 已经成为事实上的建模标准。然而,UML 虽然直观容易理解和应用,却存在着不精确的语义,而且 UML 是一种半形式化的建模语言,无法进行形式化的验证。Event-B 是一种基于大量数学谓词逻辑的形式化方法,虽然精确却难以理解和应用。因此,如何结合 UML 图和 Event-B 方法的优点是研究的重点,以往的方法都是基于 UML 零散图到 Event-B 的转换,缺乏系统的转换方法。系统性的转换方法可以实现 UML 中的元素与 Event-B 中的元素相对应统一。一般的软件系统是中型系统,中型系统采用用例图、类图、状态图和顺序图这四种图就可以很好地表达清楚,有了上述的四种图,软件生命周期的需求获取、分析、设计、详细设计就可以完全表达清楚。文章中分别给出了这四种图到 Event-B 的转换方法,并将该系统的转换方法应用到对安全性和可靠性要求较高的电梯控制系统中。基于该实例的研究,验证了 UML 到 Event-B 系统性转换方法的可行性和有效性。UML 到 Event-B 的系统转换方法不仅有利于 UML 的精确化和软件从业人员的使用,而且增强了形式化方法的可理解性,有利于形式化方法的推广和应用。

关键词:统一建模语言;形式化方法;Event-B;抽象转换;模型
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2023)12-0113-08
doi:10.3969/j.issn.1673-629X.2023.12.016

Systematic Transformation Method from UML to Event-B

GENG Xue, ZOU Sheng-rong*, LIU Xiao-ying, YAO Ju-yi
(School of Information Engineering, Yangzhou University, Yangzhou 225009, China)

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, , and lack of systematic transformation methods. The systematic transformation method can realize the unity of elements in UML and elements in Event-B. Medium-sized system can be clearly expressed by using use case diagram, class diagram, state diagram and sequence diagram. With the above four diagrams, the requirements acquisition, requirement analysis, design and detailed design of the software life cycle can be fully expressed. Based on the elevator case study, the feasibility and effectiveness of the system transformation method from UML to Event-B proposed are verified. 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.

Key words: unified modeling language (UML); formal method; Event-B; abstract transformation; model

0 引 言

UML 在软件建模行业应用普及[1],以图形化的方式帮助开发人员直观地理解系统的需求。在面向对象的软件开发中,UML 已经成为事实上的建模标准[2]。但是,实际上 UML 是一种半形式化的建模语言,存在不精确性[3],无法进行形式化的验证[4]。而形式化方法是一种严格基于数学的特种技术,可以在安全性要求较高的系统中进行验证。形式化方法的研究有很多。例如 Z 方法、B 方法、Event-B 方法等[5]。B 方法是比较热门和易于使用的形式化方法[6],Event-B 是最新的 B 方法[7],具有精确的语义[8]。但是 Event-B 形式化方法基于大量的数学逻辑谓词,对于软件需求

收稿日期:2023-02-15 **修回日期**:2023-06-16
基金项目:国家自然科学基金(61972335)
作者简介:耿 雪(1999-),女,硕士研究生,研究方向为形式化方法、高级软件工程;通信作者:邹盛荣(1968-),男,副教授,研究方向为形式化方法、高级软件工程。

分析人员来说难以理解和应用。鉴于 UML 不精确而 Event-B 方法虽精确不太易懂,将 UML 与 Event-B 相结合是一直以来研究的课题[9]。

1 研究现状

UML 与形式化方法的结合已有一些研究。例如,文献[10]中评估了 B 模型和 UML-B 模型,UML-B 模型与 Event-B 模型的可理解性,评估结果表明半形式化和形式化方法的结合促进了参与者对于模型问题域的理解。Event-B 形式化方法到 UML 的一些转换方法已经被提出[11]。UML 到 Event-B 形式化方法的转换方法也有一些被提出。例如,文献[12]提出了在元模型层自动地将 UML 图转换成 Event-B 形式化方法。文献[13]提出了活动图到 Event-B 形式化方法的转换。根据类图到 Event-B 形式化方法的转换,该方法被应用在了欧洲铁路信号系统中[14]。文献[15-16]提出了顺序图到 Event-B 形式化方法的转换。文献[17]提出了协作图、状态图到 B 形式化方法的转换。但是上述所提到的转换方法都是基于 UML 14 种图的零散个别图到 Event-B 形式化方法的转换,没有形成系统的转换方法。基于上述 UML 14 种图的零散的转换方法,在转换的过程中,可能会存在转换的不一致问题[18]和转换冲突问题,难以应用在实际的软件开发过程中。因此,对于 UML 到 Event-B 形式化方法的转换有必要加以系统化的研究。笔者认为系统化地将 UML 转换成 Event-B 形式化方法存在许多优点:一方面,软件开发人员不仅可以依据图形化的方式直观理解系统的需求,而且可以使得 UML 精确化,易于软件从业人员的使用;另一方面,将 UML 转换成 Event-B 形式化方法,可以在软件设计的早期进行形式化的验证,提高软件设计的可靠性,降低在软件开发后期因解决缺陷和错误所需付出的高额代价和成本。同时,也有利于形式化方法的普及和应用。

2 抽象转换方法

一般的软件系统是中型系统,代码量在 5 000 行到 5 万行之间,这种中型系统完全可以只选择 UML 的用例图、类图、顺序图和状态图这四种图就能够很好地表达出来[19]。在软件的需求分析阶段,用例图抽象地描述了系统的功能,对系统中的哪些用户实现系统中的哪些功能进行建模,即为软件产品本身和软件产品的使用者之间建模。类图是使用最广泛的 UML 图,可以应用于软件开发的各个阶段,在每个阶段的抽象程度和详细程度不同。类图为系统的静态结构进行建模,描述了系统中的元素以及这些元素之间的关系。UML 中的状态图不仅可以详细描述实体对象和整个系统的状态,同时也可以描述状态转换过程中触发状态转换的事件,以及系统中的实体对象在每个状态中表现出的行为。UML 中的顺序图描述了系统中实体对象之间的交互过程。可以对用例图中用例的详细执行过程进行描述,即对系统中的复杂功能模块的具体交互实现过程进行详细的展示。有了上述的四种图,软件生命周期的需求获取、分析、设计、详细设计就完全表达清楚。为了系统地将 UML 转换成 Event-B 形式化方法,基于上述所提到的四种 UML 图,该文提出了一种 UML 到 Event-B 形式化方法的转换方法。以表格的形式展示了 UML 图中的各元素与 Event-B 中的各元素之间的对应关系,并给出了 UML 到 Event-B 形式化方法的转换步骤。最后通过将该系统化的转换方法应用到电梯控制系统中,实现了电梯控制系统的 UML 图到 Event-B 形式化方法的转换,并基于 Rodin 平台为电梯控制系统建模,对于模型中产生的证明义务进行解除。使用 ProB 提供的证明器对所创建的模型进行检验,确保没有死锁、不变量违规等问题。基于该电梯系统的实例研究,证明了该系统性的转换方法的可行性和有效性。

2.1 用例图

用例图到 Event-B 形式化方法的一些转换方法已经被提出。例如,文献[20]介绍了将 UML 用例图转换成 Event-B 方法的转换步骤。基于以上的转换方法,该文基于关系的角度改进了 UML 用例图到 Event-B 的转换方法,对用例图中的元素实现了更全面的翻译。

用例图中的参与者和用例转换成上下文中的常量。用例图中的各种关系则转换为 Event-B 中的事件。其中,关联关系和扩展关系转换成抽象机器中的事件。泛化关系和包含关系则通过 Event-B 的精化机制实现。特别的,如果是参与者元素构成的泛化关系,则在扩展的上下文中添加公理表示参与者之间的泛化关系。如果是用例元素构成的泛化关系,则在精化的机器中添加事件表示用例之间的泛化关系。表 1 展示了用例图到 Event-B 方法的转换规则。

表 1 用例图转换规则

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

2.1.1 参与者和用例

将用例图中的参与者和用例转换成 Event-B 中的

常量,并在上下文中分别创建表示参与者和用例的集合 ACTOR,USECASE。添加公理声明上述的常量的类型。

CONSTANTS
  actor1
  usecase1
  …
AXIOMS
  axm1 : partition(ACTOR,{actor1}…)
  axm2 : partition(USECASE,{usecase1}...)
END

2.1.2　关　系

用例图中的关系都可以转换成 Event-B 中的事件。用例图中的关联关系转换成 Event-B 事件的具体过程展示如下:首先,抽象机器中需要创建变量 actor,usecase 以及 basic_relation 分别表示用例图中的参与者、用例和关联关系。其次,在不变量中声明上述三个变量的类型,其中,basic_relation 声明为 actor 到 usecase 之间的映射关系,表示构成该关联关系的参与者和用例。Event-B 的机器中表示关联关系的变量声明如下所示:

INVARIANTS
  inv1 : actor ∈ ACTOR
  inv2 : usecase ∈ USECASE
  inv3 : basic_relation ∈ {actor}→{usecase}

最后,在 Event-B 中的机器中创建事件表示关联关系。如下所示的 usecase1 事件表示参与者 actor1 和用例 usecase1 之间的关联关系。事件的动作模块 act1-act3 对表示参与者、用例以及关联关系的变量赋值。用例图中的其他关系与关联关系转换成 Event-B 形式化方法类似。特别的,由参与者构成的关联关系在上下文中添加公理进行表示。

usecase1 ≙
BEGIN
  act1 : actor := actor1
  act2 : usecase := usecase1
  act3 : basic_relation:∈{actor1}→{usecase1}
END

2.2　类　图

iUML-B 是一个“类似 UML”的图形前端,用于为 Event-B 形式化方法的面向对象概念建模。iUML-B 支持类图和状态机图的建模[21],在 Rodin 平台中可以实现自动地将 UML 中的类图和状态机图转换成 Event-B 形式化方法。iUML-B 应用广泛,许多使用 iUML-B 建模的研究实例已经给出。例如,文献[22]使用 iUML-B 中的类图和状态机图为欧洲铁路控制系统建模;文献[23]使用 iUML-B 为血液透析机建模。

iUML-B 中的类图提供了可视化建模数据关系的方法。类图中的类、属性和关系与 Event-B 中的常量、变量等元素相关联。特别的,类图中的类还可以与 Event-B 中的集合元素相关联。在转换的过程中,可以对这些转换而来的元素添加公理或不变量进行约束。类中的方法转换成 Event-B 中的事件,事件的动作模块表示了方法的具体执行过程。表 2 展示了 iUML-B 中类图到 Event-B 形式化方法的转换规则。

表 2　类图转换规则

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

2.3　顺序图

顺序图到 Event-B 形式化方法的一些转换方法已经被提出[16]。基于以上的转换方法中,顺序图在转换成 Event-B 方法时,顺序图中的通信对象和消息转换成上下文中的常量,机器中添加变量控制顺序图中消息的传递顺序,消息传递的源对象以及目标对象。消息的具体传递过程则在机器中的事件进行展现。表 3 展示了顺序图到 Event-B 形式化方法的转换规则。

表 3　顺序图转换规则

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

2.3.1　通信对象和消息

顺序图中的消息转换成上下文中的常量,上下文中添加表示消息的集合 MESSAGE,公理中声明表示消息的常量为集合 MESSAGE 中的元素,转换过程如下所示。类似的,通信对象也转换成上下文中的常量。

SETS
MESSAGE
CONSTANTS
  message1
  …
AXIOMS
  partition : partition(MESSAGE,{message1},…)
END

2.3.2　交互过程

机器中创建变量 source_obj,target_obj,message

和 order 分别表示消息传递的源对象、目标对象、待传递的消息以及消息的传递顺序。机器中上述变量的类型声明如下所示：

INVARIANTS
inv1 ： source_obj ∈ OBJECT
inv2 ： target_obj ∈ OBJECT
inv3 ： message ∈ MESSAGE
inv4 ： order ∈ N1

顺序图中对象之间消息的传递过程转换成 Event-B 中的事件。顺序图中 message1 消息的传递过程转换的事件如下所示。守卫条件 grd1 通过判断变量 order 的值确定消息的传递顺序是否正确。动作模块 act1-act4 明确了消息传递的源对象、目标对象和待传递的消息。

send_message1 ≙
WHEN
grd1 ： order = 1
THEN
act1 ： message：= message1
act2 ： source_obj：= object1
act3 ： target_obj：= object2
act4 ： order：= order+1
END

2.4 状态图

iUML-B 同样也支持状态机图建模。状态机图转换成 Event-B 方法时，存在两种转换方式。一种是基于 Enumeration translation 的转换方式；另一种是基于 variables translation 的转换方式[24]。两种转换方式最大的区别在于是否自动产生上下文，由于产生上下文的转换方式更容易理解，因此，该文选择的是基于 Enumeration translation 的转换方式以便于理解和应用。

表 4 状态图转换规则

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

在使用 iUML-B 为状态机图建模时。状态机图中的状态转换成常量，转换可以链接到 Event-B 中的事件，事件的发生会触发状态的改变，动作语句表示系统的具体操作。状态机图中的复合状态则通过 Event-B 的精化机制实现，抽象的上下文会通过 Event-B 方法中的扩展关系扩展出新的上下文，复合状态在扩展后的上下文中转换成常量。表 4 中明确了状态机图到 Event-B 方法的转换规则。

3 转换方法的应用

电梯是对可靠性和安全性要求较高的实时硬件系统。电梯控制系统是安全领域最常见的例子[25]，大批学者在安全控制领域中常常以电梯控制系统作为研究实例[26-27]，而且电梯是一个中型化的系统[28]，适合于该系统化的转换方法。本节将以电梯控制系统作为实例应用在该系统化的转换方法中，以验证该方法的可行性和可靠性。用户请求使用电梯时，电梯控制器根据用户请求的楼层信息调度电梯向上或向下运行，电梯运行过程中，传感器始终处于感应状态。电梯控制器在接收到传感器发送的传感信息后，控制电梯停止在相应楼层，电梯停止在该楼层后，电梯门将在一定的时间间隔内处于打开状态，等待用户进出电梯。

3.1 用例图

根据电梯控制系统的需求描述，得到电梯控制系统的用例图，如图 1 所示。

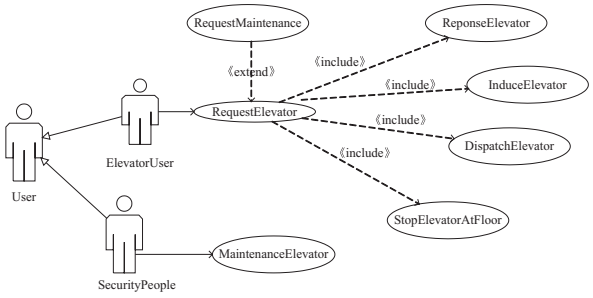


图 1 电梯模型用例图

根据上述 UML 用例图到 Event-B 的转换方法。抽象机器 m0 中的 RequestElevator 事件由用例图中 ElevatorUser 参与者和 RequestElevator 用例转换而来。在 RequestElevator 事件中，表示用户请求状态的参数 any_request 被创建，动作模块 act1-act4 为表示关联关系的变量和参数赋值。

RequestElevator ≙
ANY
any_request
WHERE
grd1 ： any_request ∈ BOOL
THEN
act1 ： actor：= ElevatorUser
act2 ： usecase：= RequestElevator
act3 ： association ：∈ {ElevatorUser} → {RequestElevator}
act4 ： control_request：= any_request
END

3.2 类 图

类图到 Event-B 形式化方法的转换，抽象电梯模

型中不再进行展示,具体转换过程将在精化模型的类图中进行描述。精化电梯模型的类图如图 2 所示。根据电梯模型需求的描述,精化的电梯模型中抽象出电梯用户类和维修人员,这两个类在转换时,则转换成上下文中的常量,这与用例图中类之间的泛化关系相一致。

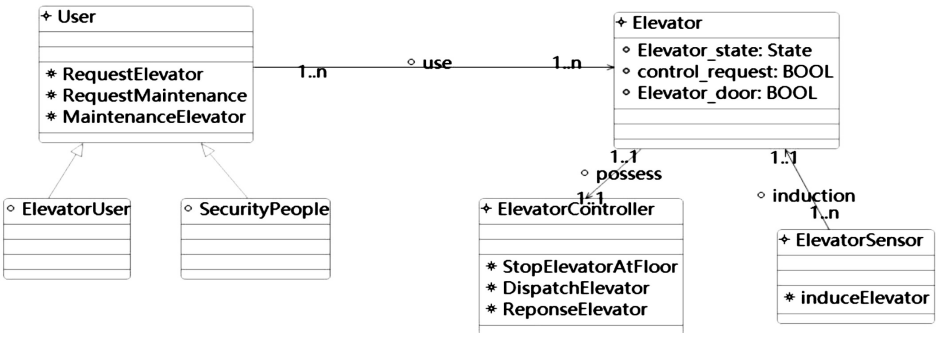


图 2　电梯模型类图

User 类中的 RequestMaintenance 方法表示用户请求维修电梯,转换成的事件如下所示。该事件的动作模块 act1 - act2 语句对变量 control_request, Elevator_state 进行赋值,表示电梯处于故障状态且无法响应用户的请求。

RequestMaintenance ≙
ANY
　this_User　//　generated class instance
WHERE
　instanceType_this_User_User : this_User ∈ User
THEN
　act1 : control_request :∈ Elevator → {FALSE}
　act2 : Elevator_state :∈ Elevator → {fault}
END

3.3　顺序图

图 3 描述了电梯响应用户请求的执行过程。用户向电梯发送请求后,电梯控制器会根据电梯的状态判断是否对用户的请求进行响应。电梯处于非故障状态,电梯控制器调度电梯运行,各楼层的传感器持续感应电梯,电梯控制器在接收到传感信息后,将电梯停止在相应的楼层。

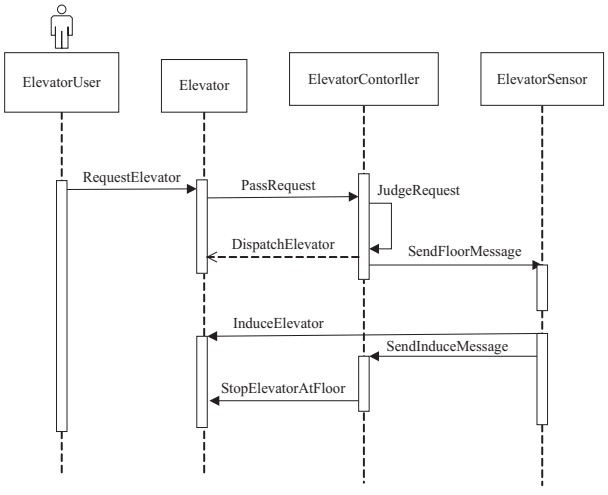


图 3　电梯模型顺序图

JudgeRequest 消息的传递过程转换成的事件如下所示。首先,守卫条件 grd1 会根据变量 order 的值判断消息的传递顺序。其次,动作 act1 - act3 明确了传递的消息为 JudgeRequest,消息传递的源对象和目标对象都是电梯控制器。最终,act4 将 order 的值增一,表示顺序图中该消息传递过程的结束。

JudgeRequest ≙
WHEN
　grd1 : order = 3
THEN
　act1 : source_obj := ElevatorController
　act2 : message := JudgeRequest
　act3 : target_obj := ElevatorController
　act4 : order := order + 1
END

3.4　状态图

精化的电梯模型状态机图如图 4 所示,主要对运行状态和故障状态进行精化。电梯在运行过程中,电梯控制器调度电梯向上或向下运行,传感器始终在感应电梯是否到达相应的楼层。所以,running 状态中添加了 dispatch 和 induce 状态。电梯处于故障状态时,安保人员在接收到维修请求后会进行维修。因此,在故障状态中,添加了 maintenance 状态表示电梯被安保人员维修。

根据前面介绍的转换规则,状态图中的 InduceElevator 事件转换的 Event-B 方法中的事件如下所示:

InduceElevator ≙
ANY
　any_Induce
WHERE
　isin_running : Elevator_Statemachine = running
　any_Induce_type : any_Induce ∈ BOOL
THEN
　act1 : isInduce : = any_Induce

enter_induce ： Elevator_running：＝induce

END

电梯在运行过程中，电梯传感器始终处于感应状态，以此感应电梯是否到达该楼层。该事件中添加了 BOOL 类型的参数 any_Induce，以参数 any_Induce 的值判断电梯传感器是否感应到电梯的到达，事件的守卫条件 isin_running 和动作 act1 明确了电梯处于被感应状态。

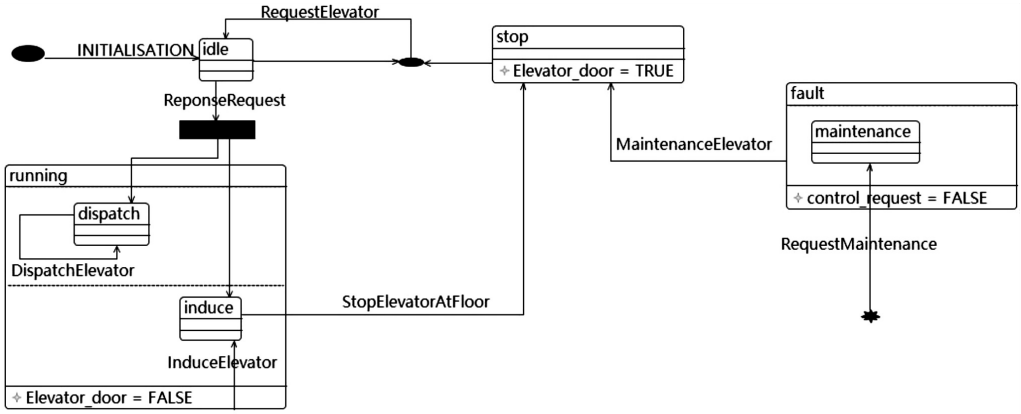


图 4 电梯模型状态机图

4 模型验证方法

4.1 验证方法

该文使用了上述所提到的 UML 到 Event-B 的系统转换方法在 Rodin 平台中为电梯控制系统建模。Rodin 平台中自带的证明器和 Atelier B 等提供的外部证明器对模型中产生的证明义务进行解除。但是尽管模型中产生的证明义务全部得到了解除，这也并不能保证模型中的事件在动态的运行过程中不会出现死锁问题。因此，又使用了 ProB[29] 提供的模型检查工具，补充验证了模型的定理证明技术，提高了模型的可靠性和精确性。此外，ProB 还提供了一个动画模拟的功能，可以动态地模拟模型中事件的执行过程，以验证模型中可能会出现的问题，增强模型的可读性和可理解性。因此，借助于 ProB 的动画模拟功能，动态地模拟了电梯模型中事件的执行过程。

4.2 自动证明结果

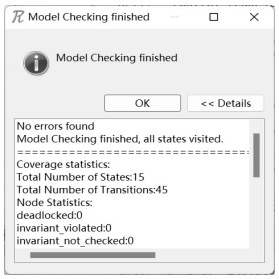
基于 Rodin 平台将电梯控制系统的 UML 图转换成 Event-B 形式化方法后，模型中产生的证明结果如图 5 所示。图 5(a)中显示抽象机器 m0 中共产生了 26 条证明义务，图 5(b)中显示精化机器 m1 中产生了 17 条证明义务，这些证明义务均已通过自动证明和交互式证明得到了解除，模型证明的结果验证了抽象转换方法的可行性。

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)机器 m0 的证明义务

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)机器 m1 的证明义务



(c)ProB 模型检查结果

图 5 模型证明结果

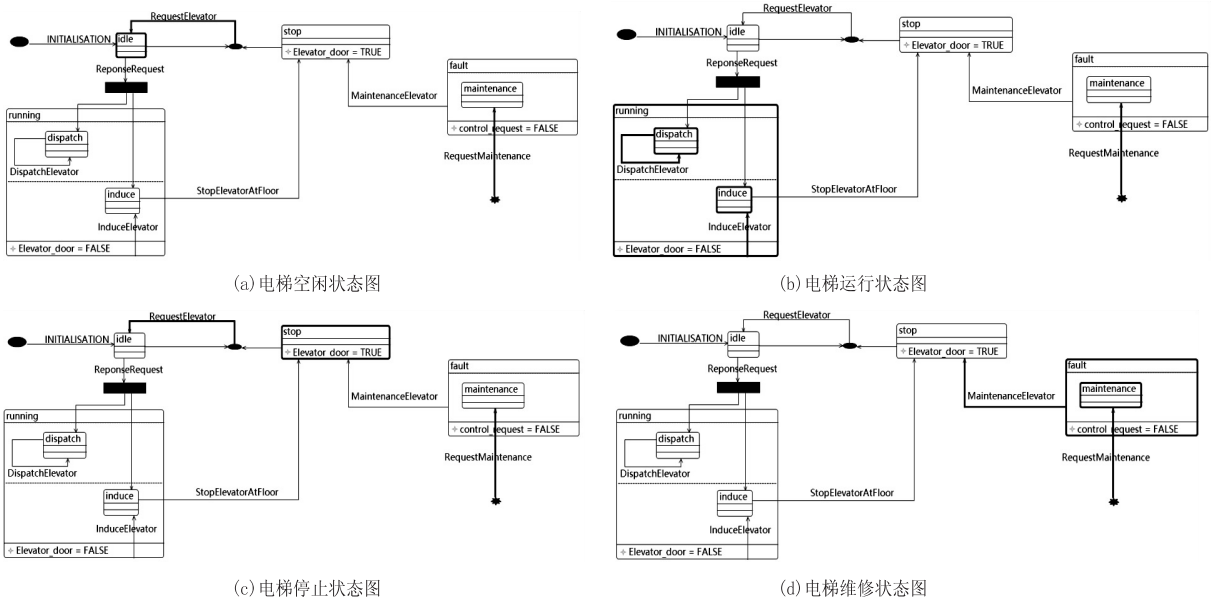
4.3 ProB 验证结果

图 5(c)中展示了电梯模型中的事件在动态执行过程中，共经历了 45 个变迁，在这些变迁转换的过程中，ProB 证明器排除了模型中可能会出现的死锁等相关问题，提高了模型的精确性，验证了抽象转换方法的正确性。

4.4 动画模拟

ProB 可以动态地模拟模型中事件的执行过程以及系统当前所处的状态，能够被触发的事件以加粗的箭头显示，加粗显示的状态表示系统当前的状态。图 6(a)中显示系统当前的状态为 idle，加粗显示的事件为 RequestElevator，即用户可以请求使用电梯，但是由于维修保养等其他外部因素，电梯可能无法响应用户的请求，因此，ReponseRequest 事件触发的前提是 RequestElevator 事件以参数 TRUE 触发。ReponseRequest 事件被触发后，图 6(b)中显示 running，dispatch 和 induce 状态均被选中，其中，running 状态中添加了不变量 Elevator_door = FALSE，保证电梯门在电梯运行过程中处于关闭状态。电梯控制器在收到了传感器发送的感应信息后，控制电梯停止在该楼层。因此，

StopElevatorAtFloor 事件触发的前提是 InduceElevator 事件以参数 TRUE 触发。电梯停止后,即 StopElevatorAtFloor 事件触发后,图 6(c)中显示电梯处于 stop 状态。由于电梯发生故障的不确定性,所以 RequestMaintenance 事件在任何状态下都可以被触发,图 6(d)中显示了该事件被触发后,fault 状态被选中,变量 control_request 的值为 FALSE,表示电梯处于故障状态下,无法响应用户的请求。



(a)电梯空闲状态图　　(b)电梯运行状态图

(c)电梯停止状态图　　(d)电梯维修状态图

图 6　ProB 动画模拟状态图

5　结束语

该文提出的转换方法系统地将半形式化建模语言(UML)转换成 Event-B 形式化方法。该转换方法将 UML 与 Event-B 形式化方法结合带来了许多优点。一方面,使 UML 精确化,可以在软件设计的早期对所建立的模型进行形式化的验证,提高软件设计的可靠性和准确性,有利于软件从业人员的使用。另一方面,增强了形式化方法的可理解性和可读性,有利于形式化方法的推广和应用。采用了四种图系统地将 UML 转换成 Event-B 形式化方法,一般的中型系统采用这四种图就可以很好地表达清楚。对于复杂的特殊系统建模时可能需要添加其它的 UML 图去补充。例如,对于实时嵌入式系统,可能需要添加时间图进行补充。将来,可以对转换的方法提出改进,实现更全面的将 UML 图转换成 Event-B 形式化方法。

参考文献:

[1] OZKAYA M, ERATA F. A survey on the practical use of UML for different software architecture viewpoints[J]. Information and Software Technology, 2020, 121(4): 106275.

[2] KOCHALEEMA K H, KUMAR G S. Methodology for formal verification of UML models[J]. Defence Science Journal, 2022, 72(1): 40-48.

[3] ZHANG F, CHENG J. Verification of fuzzy UML models with fuzzy description logic[J]. Applied Soft Computing, 2018, 73(8): 134-152.

[4] OZKAYA M. Do the informal & formal software modeling notations satisfy practitioners for software architecture modeling? [J]. Information & Software Technology, 2017, 95(10): 15-33.

[5] 邹盛荣,郑国梁. B 语言和方法与 Z、VDM 的比较[J]. 计算机科学,2002,29(10):136-138.

[6] FERRARI A, BEEK M. Formal methods in railways: a systematic mapping study[J]. ACM Computing Surveys, 2023, 55(4): 1-37.

[7] ABRIAL J. On B and Event-B: principles, success and challenges[C]//Abstract state machines. [s. l.]: Springer, 2018: 31-35.

[8] LE H A, TRUONG N T, NAKAJIMA S. Verifying eventuality properties of imprecise system requirements using Event-B[C]//ACM symposium on applied computing. Salamanca: Association for Computing Machinery, 2015: 1651-1653.

[9] ALKHAMMASH E, BUTLER M, FATHABADI A S, et al. Building traceable Event-B models from requirements[J]. Science of Computer Programming, 2015, 111(2): 318-338.

[10] RAZALI R, SNOOK C F. Comprehensibility of UML-based formal model - a series of controlled experiments[C]// ACM international workshop on empirical assessment of software engineering languages and technologies. Atlanta: Association for Computing Machinery, 2007: 25-30.

[11] SAYAR I, BHIRI M T. From an abstract specification in Event-B toward an UML/OCL model[C]//Formal methods in software engineering. Hyderabad: Association for Computing Machinery, 2014: 17-23.

[12] SUN W X, ZHANG H, FENG C, et al. A method based on

meta-model for the translation from UML into Event-B [C]//IEEE international conference on software quality, reliability and security companion. Vienna: IEEE, 2016: 271-277.

[13] ACHOURI A, HLAOUI Y B. Institution-based UML activity diagram transformation with semantic preservation[J]. International Journal of Computational Science and Engineering, 2019, 18(3): 240-251.

[14] WAKRIME A A, AYED R B. Formalizing railway signaling system ERTMS/ETCS using UML/Event-B[C]//International conference on model and data engineering. Marrakesh: Springer, 2018: 321-330.

[15] MOUAKHER I, DHAOU F. Event-based semantics of UML 2. X concurrent sequence diagrams for formal verification [J]. Journal of Computer Science and Technology, 2022, 37(1): 4-28.

[16] HLAOUI Y B, BEN YOUNES A. From sequence diagrams to event B: a specification and verification approach of flexible workflow applications of cloud services based on meta-model transformation[C]//Computer of software and applications conference. Turin: IEEE, 2017: 187-192.

[17] TRUONG N T, JEANINE S. Verification of behavioural elements of UML models using B[C]//ACM symposium on applied computing. Santa Fe New: Association for Computing Machinery, 2005: 1546-1552.

[18] ZAPATA C M, GONZÁLEZ G, GELBUKH A, et al. A rule-based system for assessing consistency between UML models[C]//Advances in artificial intelligence. Aguascalientes: Springer, 2007: 215-224.

[19] 邹盛荣,周 塔,顾爱华,等. UML 面向对象需求分析与建模教程[M]. 第 2 版. 北京:科学出版社,2018: 116-125.

[20] SUN W X, ZHANG H. A method for the translation from UML into Event-B[C]//International conference on software engineering and service science. Beijing: IEEE, 2016: 349-352.

[21] SAID M Y, BUTLER M, SNOOK C. A method of refinement in UML-B[J]. Software & Systems Modeling, 2015, 14(4): 1557-1580.

[22] DGHAYM D, DALVANDI M. Formalising the hybrid ERTMS level 3 specification in iUML-B and Event-B[J]. International Journal on Software Tools for Technology Transfer, 2020, 22(3): 297-313.

[23] HOANG T S, SNOOK C. Validating the requirements and design of a hemodialysis machine using iUML-B, BMotion studio and co-simulation[C]//Abstract state machines. [s. l.]: Springer, 2016: 360-375.

[24] PENG H, DU C L, RAO L, et al. Modelling the embedded control system using iUML-B pattern state machine[J]. Journal of Control Science and Engineering, 2018, 2018(PT. 1): 1-12.

[25] WANG X L, ZHANG X P. Research on the risk points of elevator braking system[C]//International conference on artificial intelligence and computer engineering. Beijing: IEEE, 2020: 28-31.

[26] CHEN W X, ZHENG B J, LIU J Y. A real-time matrix iterative optimization algorithm of booking elevator group and numerical simulation formed by multi-sensor combination [J]. Electronics, 2021, 10(24): 2-32.

[27] HOANG T S, ABRIAL J. Reasoning about liveness properties in Event-B * [C]//13th international conference on formal engineering methods. Durham: Springer, 2011: 456-471.

[28] JOSÉ L, SÁNCHEZ F. Modelling and evaluating real-time software architectures[C]//Ada-Europe international conference on reliable software technologies. Brest: Springer, 2009: 164-176.

[29] SCHMIDT J, LEUSCHEL M. SMT solving for the validation of B and Event-B models[J]. International Journal on Software Tools for Technology Transfer, 2022, 24(6): 1043-1077.