一种面向嵌入式软件体系结构的形式化建模方法

许海洋[1,2],庄 毅[1],顾晶晶[1]
(1.南京航空航天大学计算机科学与技术学院,江苏南京 210016;2.青岛农业大学理学与信息科学学院,山东青岛 266109)

摘 要: 为了解决 MARTE(Modeling and Analysis of Real Time and Embedded systems)在建立嵌入式软件模型时不够精确的问题,结合 Object-Z 和 PTA(Probabilistic Timed Automation)的优点,本文提出了一种集成的形式化建模方法——PTA-OZ.该方法不仅能够对嵌入式软件模型的静态语义和动态语义进行精确描述,而且通过模型转换规则,能够将 MARTE 模型转换为 PTA-OZ 模型.并对模型转换的语义一致性进行了验证,证明本文方法在转换过程能够保持结构语义和行为语义的一致性.最后通过实例模型描述从嵌入式软件建模到属性检验的过程.

关键词: 集成模型; 模型转换; 概率时间自动机; 语义一致性
中图分类号: TP301 **文献标识码:** A **文章编号:** 0372-2112 (2014)08-1515-07
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2014.08.009

A Formal Modeling Method for Embedded Software Architecture

XU Hai-yang[1,2], ZHUANG Yi[1], GU Jing-jing[1]
(1. *College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu* 210016, *China*;
2. *College of Science and Information, Qingdao Agricultural University, Qingdao, Shandong* 266109, *China*)

Abstract: MARTE is utilized for embedded software modeling. However, it could not provide powerful methods for rigorously analyzing the correctness of software systems. This paper introduces the PTA-OZ, which combines the specification language Object-Z with the probabilistic processes PTA. It can provide accurate descriptions for both static and dynamic semantics of embedded software models. Additionally, we propose the model transformation rules for converting MARTE models to PTA-OZ models. We verify the semantics preservation in model transformation, proving that the model transformation rules can keep both the structural and behavioral consistency of software. A practical case is demonstrated throughout the processes of software modeling and property verification.

Key words: integrating model; model transformation; probabilistic timed automation; semantic consistency

1 引言

在嵌入式软件设计过程中,如何精确地描述软件模型的功能性和非功能性需求是极其关键的问题.MARTE 支持面向对象的建模和设计语言,提出了较完善的时间模型和非功能属性等建模元素,不仅为实时嵌入式软件的构造和分析提供了基础[1,2],而且适于软硬件协同设计[3].Airbus、INRIA 等许多机构在 MARTE 的基础上开展了模型驱动的实时嵌入式软件的构造、分析与验证工作[4~6].但 MARTE 缺乏精确性,无法为其模型的验证提供严格的形式化方法[7].为增强软件模型的可靠性,研究者开展了许多为 MARTE 模型提供形式化的语义的解释工作.

在 Object-Z 和 PTA 的基础上,本文提出一种集成的形式化建模方法 PTA-OZ.使得软件开发者只需要采用 MARTE 建立面向对象的结构特征和行为特征,转换规则自动将 MARTE 模型转换为 PTA-OZ 模型,采用已有的 Object-Z 和 PTA 检验工具验证模型.对于模型转换的一致性,采用不变式和三元元模型相结合的方法,证明了提出的模型转换具有结构语义和行为语义一致性.该方法结合已有研究在静态结构和动态结构的优点,采用模型转换规则将 MARTE 模型转换为 PTA-OZ 模型,有利于在设计阶段验证软件属性.

收稿日期:2013-05-17;修回日期:2013-12-12;责任编辑:马兰英
基金项目:国家自然科学基金青年科学基金(No.61202351);国家博士后基金(No.2011M500124);江苏省普通高校研究生科研创新计划(No.CXLX12_0161);南京航空航天大学基本科研业务费(No.NS2012133)

2 相关工作

针对抽象规约到具体规约的数据精化,Didier 等提出一种基于精化检测的方法[8],而 Derrick 等提出基于模型检测器的时序逻辑来精化模型[9]. Rasch 等研究了类与状态机之间的一致性[10],表明通过模型检测器可自动地执行一致性检测.针对设计阶段的 UML 序列图,他们又提出将序列图量化的方法[11].对于时序 UML 状态机,Knapp 等给出了一个工具 HUGO/RT[12],利用模型检测器 UPPALL 来验证两个时间自动机的一致性. Möller 等将 CSP 和 Object-Z 集成[13],但没有考虑时间对于软件安全性的影响. Basin 等提出使用 CSP-OZ 来描述安全自动机,将其与目标系统的组合形式化,并分析产生的系统规范的安全性[14].

单一形式化建模方法在描述软件时存在不足,而且现有研究工作主要集中在 UML 模型,对于 MARTE 模型形式化的研究还比较少.此外,现有的研究没有考虑验证后的模型精化问题,没有建立图形建模、形式化验证、代码实现的系统化结构框架.

3 集成的建模方法

在需求分析阶段,最好采用形式化方法来描述软件[15],而现有的形式化方法通过静态分析无法保证给定的状态是可执行的[16].

3.1 PTA-OZ 模型

定义 1 概率时间自动机是一个八元组 $P=(S, S_{init}, trap, C, E, F, A, \rho)$[17],其中:

(1) S 是状态集.

(2) $S_{init}\in S$ 是初始状态.

(3) $trap\in S$ 是俘获状态.

(4) C 是时钟集.

(5) $E\subset S\times 2^C\times guard(C)\times S$ 是边集合,每个边用一组时钟及时钟约束标注.

(6) $F\subset S$ 是接受状态集,用于定义接受条件.

(7) A 是非空有限集合 A_s 的不相交并集,A_s 是状态 s 的行为集合.

(8) $\rho: A\times E\rightarrow[0,1]$是一个概率转移函数,$\rho(a,e)$ 表示行为 a 发生时,状态 s 使用边 e 的概率.

Object-Z 在 Z 的基础上增加了抽象数据类型和继承[18]. Object-Z 是基于状态的,PTA 是面向行为的,结合二者在建模的优点,我们提出集成模型 PTA-OZ.

定义 2 PTA-OZ 是一个六元组 $PO=(A_n, I, L, T, P, Z)$,其中:

(1) A_n 使用 inherit 继承所有的父类.

(2) I 由接口声明组成,这些声明由类提供和使用.

(3) L 是一个局部通道,它不能够从外部访问.

(4) T 用于定义模式中的类型和常量.

(5) P 是一个概率时间自动机,由定义 1 的八元组组成.

(6) Z 采用 Object-Z 格式描述类,由一个初始模式、若干状态模式和操作组成.

图 1 是一个名为 C 的 PTA-OZ 类模式.从语法上讲,声明的不同类型用 method、chan 和 local_chan 来区分.

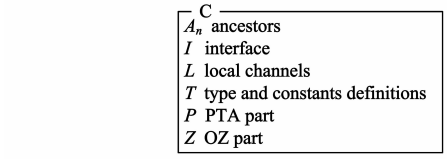


图1 PTA-OZ的类模式

定义 2 给出了 PTA-OZ 的形式化描述,还需要建立 PTA 和 Object-Z 的关联. PTA 由状态集 S 和活动集 A 等元素组成,Object-Z 的类是一组参数的模型,类的状态可以通过参数的赋值来表示,类的操作会引起状态的变化. Object-Z 类的状态集为 $S_z = Id_p \nrightarrow Value$,表示对每个 Id_p 进行赋值 $Value$ 得到的 Object-Z 类的状态集. PTA 的状态与 Object-Z 类的状态之间的关联为 $S\subseteq S_z$.

类的操作集表示为 $O = Id_o\times(Id_p \nrightarrow Value)$,其中 Id_o 表示操作名,$Id_p \nrightarrow Value$ 表示对操作的每个 Id_p 进行赋值 $Value$. 建立 PTA 的行为和 Object-Z 类的操作之间的关联 $action((n,p)) = n$,函数 $action$ 将 Object-Z 类的操作$(n,p)\in O$ 映射为 PTA 的行为 n.

3.2 模型转换规则

规则 1 类转换规则:MARTE 类名映射为 Object-Z 的类名;类的属性转换为状态模式;关联转换为类的属性;类需要的接口操作转换成通道;类提供的接口操作转换成方法.形式化描述为:

mapMARTEClassToOZ: MARTEClass→ℙ OZClass

∀ mc: MARTEClass · mapMARTEClassToOZ(mc) = {oc: OZClass | mc.name = oc.name ∧

∀ ma: mc.attributes · ∃ oa: oc.attributes ·

oa.name = mla.name ∧ oa.type = convType(ma.type) ∧

oa.visibility = ma.visibility ∧ oa.multiplicity = ma.multiplicity ∧

oa.relationship = relNone ∧ oa.navigability = navNone

∀ mro: mc.require_operations · ∃ me: oc.methods ·

me.name = mro.name ∧ me.visibility = mro.visibility

∀ mp: mro.parameters · ∃ op: me.parameters ·

op.name = mp.name ∧ op.type = convType(mp.type)}

∀ muo: mc.use_operations · ∃ ch: oc.channels ·

ch.name = muo.name ∧ ch.visibility = muo.visibility

∀ mp: muo.parameters · ∃ op: ch.parameters ·

op.name = mp.name ∧ op.type = convType(mp.type)}

自动机是状态图的基础,它能够更加准确地验证类的行为[19,20].

规则 2　状态图转换规则:将状态映射到 PTA 的状态节点;将事件映射到 PTA 的行为;变迁映射到 PTA 的边.事件的时间约束映射为 PTA 边上的时钟约束标注,状态变迁发生的概率映射为 PTA 边上的概率转移函数.形式化描述为:

mapMARStateChartToPTA:MARSC→PTA

∀msm:MARSC·mapMARStateChartToPTA(msm) = {p:PTA |

∀ms:msm.state·∃ps:p.S·ps = mapMARTEClassToOZ(ms)

∀me:msm.event·∃pa:p.·pa = me

∀mt:msm.transition·∃pe:p.E·pe.name = mt.name∧pe.source = mt.source∧

pe.guard = mt.guard∧pe.acton = mt.action∧pe.target = mt.target∧pe.reset = mt.clocks

∀mp:msm.probability·∃ρ:p.ρ·ρ = mp∧ρ.a = me∧ρ.e = pe.acton}

规则 3　根据规则 1、2,则 MARTE 模型和 PTA-OZ 模型的转换规则为:(1)MARTE 类转换成 PTA-OZ 模型的 Object-Z 类;(2)MARTE 状态图转换成 PTA-OZ 模型的 PTA 表达式;且 PTA 部分的行为与 OZ 部分的方法存在映射关系.形式化描述为:

mapMARTEToPTA_OZ:MARTE→PTA_OZ

∀mar:MARTE·mapMARTEToPTA_OZ(mar) = {po:PTA_OZ |

∀mc:mar.class·∃oz:po.oz·oz = mapMARTEClassToOZ(mc)

∀ms:mar.statechart·∃pta:po.pta·pta = mapMARStateChartToPTA(ms)

pta.S⊆oz∧pta.A = = action((n,p))∧n = oz.methods}

3.3　PTA-OZ 模型的检验

利用已有的模型检测工具,对转换后的模型进行检验.对 OZ 部分,检测 Object-Z 规范的语法和类型的正确性,保证所有操作严格地使用状态模型.形式化工具 Z/EVES 和 PVS 能够检查、分析 Z 形式规格说明.

但是 Object-Z 只适用于数据精化,而 PTA-OZ 模型通过验证操作的正确性,可以实现操作精化.形式化工具 PRISM 支持 MDPs 和 PTA,本文采用 PRISM 来检验 PTA-OZ 模型的动态结构.

4　模型转换的一致性验证

在模型转换正确性验证[21]的基础上,对语义一致性的研究多基于模型检测技术[22,23].根据不变式检测图模型转换[24],我们采用不变式和三元元模型来验证模型转换的一致性.设 £(M)表示类型为 M 的所有模型集合,δ_M 表示 M 上的约束.

定义 3　设有两个模型 S、T,如果存在模型转换 MT,使得 S 在 MT 下能够转换为 T,且 T 的语义包含 S 的语义,则称 MT 具有语义一致性.

定义 4　转换规则由三个元模型组成 TR = (S,C,T),其中 $S = \{sm_1,\cdots,sm_p\}$ 表示源元模型集,$T = \{tm_1,\cdots,tm_q\}$ 表示目标元模型集,C 表示 S 和 T 之间的对应关系.

定义 5　设 MTS = (R,M)为一个元模型转移系统,其中 R 表示定义在元模型 M 上的规则集合.对于模型 $M \in £(M)$,规则 $\rho \in R$,以及不变式 *inv*.如果 M 满足 *inv*,且在 ρ 下能够转换为模型 M′,有 $M' \vdash inv$,则称 *inv* 为 MTS = (R,M)的诱导不变式.

定义 6　设 LTS(MTS,M_0)为由 MTS = (R,M)和初始模型 M_0诱导生成的标记转移系统,记作 LTS(MTS,M_0) = (Init,→,Node,Label).l(LTS(MTS,M_0))表示由重标记映射 l:R→A 定义的标记转移系统.

定义 7　设 TMTS = (MTS,$S_0C_0T_0$)为由 MTS = (R,SCT)和模型转换的初始元模型 $S_0C_0T_0$ 诱导生成的三元元模型转移系统.用 MT(*tmts*,δ_{tmts})表示三元元模型转移系统 *tmts*,在约束 δ_{tmts}下实现的源模型到目标模型之间的转换.

对通常情况下的三元元模型转移系统,根据实际要求进行约束,构造适用于在 3.2 节定义的转换规则的模型转换.

定义 8　设 $lts_1 = \langle I_1,\rightarrow,N_1,L_1\rangle$、$lts_2 = \langle I_2,\rightarrow,N_2,L_2\rangle$为两个标记转移系统,$lts_1$和 lts_2在字母表 A 上是互模拟的当且仅当存在一个二元关系 $Bis \subseteq N_1 \times N_2$,使得当$(n_1,n_2) \in Bis$,$\alpha \in A$ 时,有下列关系成立:(1)如果 $n_1 \xrightarrow{\alpha} n'_1$,那么 $n_2 \xrightarrow{\alpha} n'_2$,且$(n'_1,n'_2) \in Bis$;(2)如果 $n_2 \xrightarrow{\alpha} n'_2$,那么 $n_1 \xrightarrow{\alpha} n'_1$,且$(n'_1,n'_2) \in Bis$.

采用 $Bis(\delta_{Bis},SCT)$表示三元元模型 SCT 在约束 δ_{Bis} 条件下的互模拟关系.

定理 1　设 S_M 为结构源元模型,T_M 为结构目标元模型.模型转换 MT(*tmts*,δ_{tmts}):£(S_M)×£(T_M)是结构语义一致的,当且仅当下述条件成立:

(1)对于 $S_0 \in S_M$,存在 $T_0 \in T_M$,使得(S_0,T_0)MT(*tmts*,δ_{tmts}).

(2)对于任意的 $S \in S_M$且 $S \in$ LTS(MTS,S_0),存在 $T \in T_M$,使得(S,T)∈MT(*tmts*,δ_{tmts}).

证明　若模型转换 MT(*tmts*,δ_{tmts})是结构语义一致的,则结论显然成立.

反之,设 Closure(M)表示元模型 M 的传递闭包.由定理的条件(1)$S_0 \in S_M$ 知,对于 Closure(S_0),总有 Closure(T_0),且(S_0,T_0)∈MT(*tmts*,δ_{tmts}),所以 Closure(S_0)⊆Closure(T_0).

由定理的条件(2)S∈LTS(MTS,S_0),及定义 6 可知,

存在 $\rho_s \in R$,使得 $S_0 \Rightarrow_{\rho s} S$. 于是有 $S \in Closure(S_0)$, $Closure(S) \in Closure(S_0)$. 假设一般情况下 $\rho_s \cap \rho_t = \varnothing$,当 $l_s(\rho_s) = l_t(\rho_t)$时,对于三元元模型 S_0CT_0的 ρ_s 诱导等价于对于 S_0CT_0 的 ρ_t 诱导,即 $S_0CT_0 \Rightarrow_{\rho t} S_0CT$,于是有 $T_0 \Rightarrow_{\rho t} T$. 于是有 $T \in Closure(T_0)$, $Closure(T) \subseteq losure(T_0)$. 所以 $Closure(S) \subseteq Closure(T)$.

所以,根据定义3及模型转换结构依赖关系的一致性,可知模型转换是结构语义一致的.

定理 2 设互模拟约束 $\delta_{Bis} = \delta_{RT} \wedge \delta_{Pair} \wedge \delta_{tmts}$,其中 δ_{RT}为定义在 SCT 上的运行时约束,δ_{Pair}为约束对,$\delta_{Pair} = \{(\rho_s, \rho_t) \mid l_s(\rho_s) = l_t(\rho_t) \wedge \rho_s \in R \wedge \rho_t \in R\}$. 模型转换 $MT(tmts, \delta_{tmts})$: $\pounds(S_M) \times \pounds(T_M)$是行为语义一致的,当且仅当下述条件成立:

(1) $S_0C_0T_0 \vdash \delta_{RT} \wedge \delta_{Pair}$.

(2) $\delta_{RT} \wedge \delta_{Pair}$为$(R_M, S_MC_MT_M)$的诱导不变式.

(3)设 $\gamma(l_s, l_t) = \{\rho_s + \rho_t \mid (\rho_s, \rho_t) \in \delta_{Pair}\}$, $\delta_{RT} \wedge \delta_{Pair}$为$(\gamma(l_s, l_t), S_MC_MT_M)$的诱导不变式.

证明 由条件(1)和定义8知,对于任意$(S_1, T_1) \in MT(tmts, \delta_{tmts})$,如果有 $S_1 \rightarrow S_2$,那么 $T_1 \rightarrow T_2$ 且$(S_2, T_2) \in Bis(\delta_{Bis}, SCT)$. 如果有 $S_1 \Rightarrow_{\rho s} S_2$ 且 $l_s(\rho_s) = l_t(\rho_t)$,则只需证明 $T_1 \Rightarrow_{\rho t} T_2$ 且$(S_2, T_2) \in Bis(\delta_{Bis}, SCT)$.

设已知 $S_1 \Rightarrow_{\rho s} S_2$,则有 $S_1CT_1 \Rightarrow_{\rho s} S_2CT_1$. 由于 $S_1CT_1 \vdash \delta_{Pair}$,于是当 $l_s(\rho_s) = l_t(\rho_t)$时,对于 S_1CT_1 的 ρ_s 诱导等价于对于 S_1CT_1 的 ρ_t 诱导,即 $S_1CT_1 \Rightarrow_{\rho t} S_1CT_2$,有 $T_1 \Rightarrow_{\rho t} T_2$.

设 $\rho_s \cap \rho_t = \varnothing$,则 $S_1CT_1 \Rightarrow_{\rho s} S_2CT_1 \Rightarrow_{\rho t} S_2CT_2$且 $\rho_s + \rho_t \in \gamma(l_s, l_t)$. 由条件(3),如果 $S_1CT_1 \vdash \delta_{RT} \wedge \delta_{Pair}$,则 $S_2CT_2 \vdash \delta_{RT} \wedge \delta_{Pair}$. 已知 $MT(tmts, \delta_{tmts})$为模型转换,$S_0C_0T_0 \vdash \delta_{tmts}$以及 $S_1CT_1 \Rightarrow_{\rho s + \rho t} S_2CT_2$,由定义5,有 $S_2CT_2 \vdash \delta_{tmts}$,故 $S_2CT_2 \vdash \delta_{RT} \wedge \delta_{Pair} \wedge \delta_{tmts}$,即$(S_2, T_2) \in Bis(\delta_{Bis}, SCT)$.

对于任意初始状态(S, T),已知 $SCT \in \pounds(tmts, \delta_{tmts})$,有$SCT \vdash \delta_{tmts}$.

由条件(1),有 $S_0C_0T_0 \vdash \delta_{RT} \wedge \delta_{Pair}$;由条件(2)以及定义5,假设 $S_nC_nT_n \vdash \delta_{RT} \wedge \delta_{Pair}$, $S_nC_nT_n \Rightarrow S_{n+1}C_{n+1}T_{n+1}$,则 $S_{n+1}C_{n+1}T_{n+1} \vdash \delta_{RT} \wedge \delta_{Pair}$,因此$SCT \vdash \delta_{RT} \wedge \delta_{Pair}$.

由归纳法可知$SCT \vdash \delta_{RT} \wedge \delta_{Pair} \wedge \delta_{tmts}$,即$(S, T) \in Bis(\delta_{Bis}, SCT)$.

综上所述,存在一个互模拟关系 $Bis(\delta_{Bis}, SCT)$,根据定义8和定义3,模型S和T的语义等价.因此模型转换 $MT(tmts, \delta_{tmts})$是行为语义一致的.

5 实例分析

以某指控系统中的嵌入式软件为例,该软件主要包括传感器、信息处理、管理、设备控制、通信和武器单元等模块.设各个模块的平均失效时间,传感器为2个月,武器单元为3个月,信息处理、管理和设备控制为1年.

5.1 实例的 MARTE 模型

采用 MARTE 建立软件模型,该模型由类图和状态图组成.RtUnit 和 PpUnit 表示实时嵌入式软件中的活动对象,PpUnit 用来对模型中的受限资源进行建模,由 RtUnit 所拥有并进行控制.

本文仅研究管理模块通过通信模块与其它模块进行的交互:信息处理模块解析传感器数据,通过通信模块将结果数据发送给管理模块;管理模块根据该数据,通过通信模块向设备控制模块发送指令信息.

在图2中共有2个 rtUnit 单元,分别为管理模块 Manager 和通信模块 Communication;1个 ptUnit 单元,为数据库模块 DataBase. Manager 为主操作,它依赖 Communication 与其它单元进行通信,通过 DataBase 共享数据. Manager 和 Communication 动态地创建一个可调度资源,来执行服务,Communication 的可调度资源池大小为4.

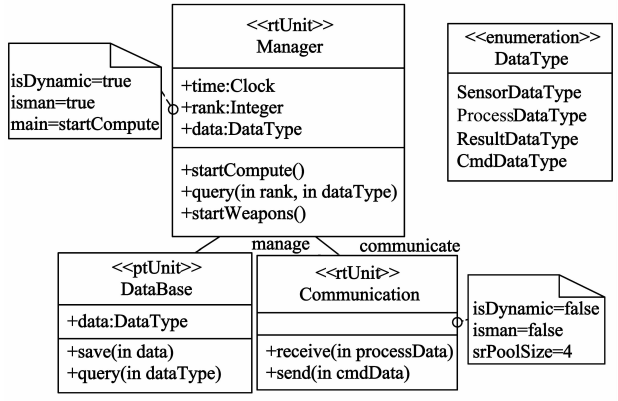


图2 管理模块的类图

使用构造型≪timedProcessing≫指定一个行为的持续时间.图3描述了管理模块的主要状态:Init、Detecting、Computing、WPreparing、Fail 和 Success,有些状态上具有时间约束.

5.2 实例的 PTA-OZ 模型

根据 MARTE 模型和 PTA-OZ 模型的转换规则,将图2的类图和图3的状态图转换为 PTA-OZ 模型的类模式,转换结果见图4.

关联类 Communcation 依赖 Manager 类,转换为 Manager 类的属性 comm. Manger 类的属性 data、time 等转换为 PTA-OZ 模型状态模式. Manager 类自身的操作 startCompute、query、startWeapons 转换为 PTA-OZ 模型的 method, Manager 类需要的操作 send 转换为 PTA-OZ 模型的 chan.

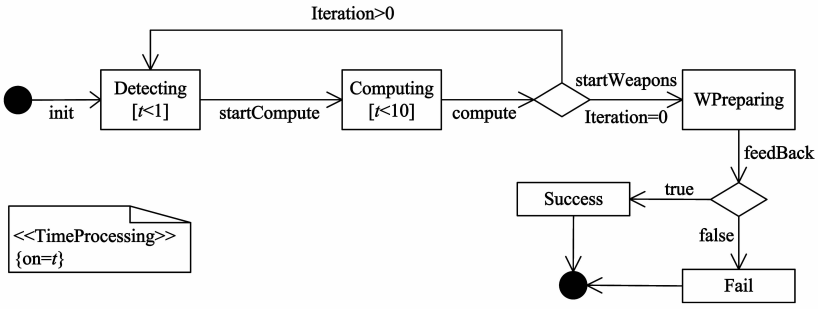


图3　管理模块的状态图

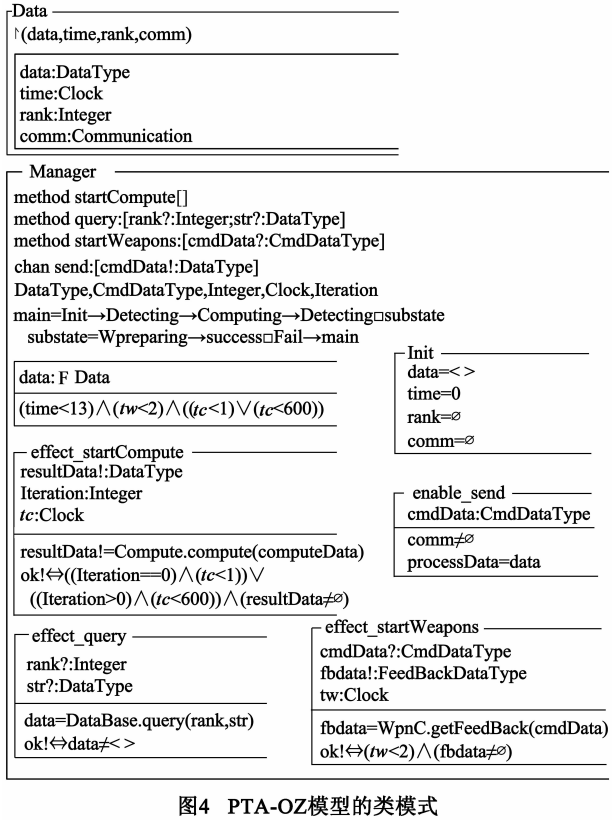


图4　PTA-OZ模型的类模式

操作 enable_send 的 guard 条件为已经建立 comm 连接,并且对传感器的数据已经解析完成.操作 effect_op 描述操作 op 引起的状态转变:effect_query 描述根据操作人员的权限对数据库中数据进行查询后的状态变化;effect_startCompute 发起计算,信息处理模块解析传感器的原始数据,进行预处理等.

对于图 4 类模式中的 PTA 部分,我们根据 PTA 的定义,详细定义管理模块的 PTA 模型(见图 5).由于只有一个时钟,因此在表示边集合时,采用三元组表示一条边.

5.3　属性检验

对于软件的属性要求,需要考虑各个模块的平均失效时间,根据指控系统的需求,要求软件模型保持如下属性规范:

S={Init,Detecting,Computing,Wpreparing,Success,Fail}
S_{init}={Init}
C=time
guard(C)={$t_{startCompute}$, $t_{startWeapons}$, $t_{feedback}$}
E={(Init,Detecting , True),(Detecting ,Computing , $t_{startCompute}$),(Computing , Wpreparing , $t_{startWeapon}$),(WPreparing,Success, $t_{feedBack}$),(WPreparing,Fail, $t_{feedBack}$)}
A={A_{s1},A_{s2},A_{s3},A_{s4},A_{s5},A_{s6}}={init,startCompute,startWeapons,feedBack}
ρ(Init,Detecting)=1-1/(365*24*60*60)
ρ(Detecting,Computing)=1-1/(365*24*60*60)-1/(24*60*60)
ρ (Computing,WPreparing)=1-1/(365*24*60*60)-1/(24*60*60)
ρ(WPreparing,Success)=ρ(WPreparing, Fail)
=1-1/(2*365*24*60*60)-1/(24*60*60)

图5　PTA部分详细描述

属性 1　系统无死锁.

属性 2　管理模块从传感器发送数据,到接收到设备控制模块的反馈消息的时间之和不超过 13s.

属性 3　计算一天内该指控系统各个模块发生故障的概率.

将上述属性表示为 CSL 逻辑公式,并采用 PRISM 属性规范语言编码,见表 1.

表 1　属性表示

	CSL 公式	PRISM 编码
属性 1	¬deadlock	! deadlock
属性 2	$\mathscr{S}^{=?}[\mathcal{E}^{<T}]$	R{"up"}=?[C<T]
属性 3	$\mathscr{P}_{=?}[\neg$ shutdown$\mathcal{U}$ fail$]$	P=?[!"down" U<=24*3600"fail-module"]

容易验证采用 PTA-OZ 模型表示的软件模型的行为属性 1 是满足的.在假设平均失效时间的情况下,属性 2 也满足.对于属性 3,我们计算传感器、武器单元和管理模块发生故障的概率分布图,从图 6 可见,在初始阶段,管理模块发生故障的概率较高,传感器和武器单元发生故障的概率较低.为了更加清楚地对比传感器和武器单元的故障率,图 7 给出 1 小时内二者故障发生概率的分布图,可见武器单元的故障概率要远小于传感器.但从长期工作状态来看,传感器模块发生故障的概率更高(见图 8).

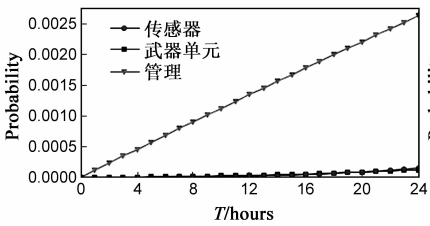


图6 1天内故障率

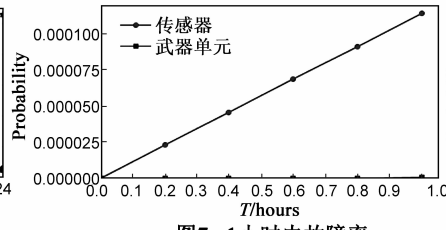


图7 1小时内故障率

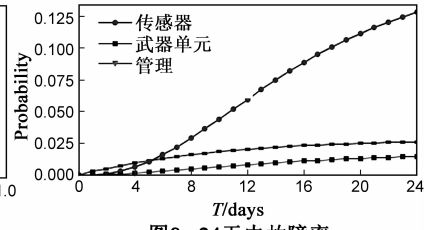


图8 24天内故障率

6 结论

本文结合 Object-Z 和 PTA 在表示软件模型方面的优点,提出了一种集成的形式化方法,利用 Object-Z 描述 MARTE 模型的静态结构,利用 PTA 描述 MARTE 模型的行为语义,证明了模型转换过程的语义一致性.使得软件开发者专注于利用 MARTE 进行嵌入式软件的可视化建模设计,集成方法能够将图形模型自动转换为形式化模型,而软件语法检查和行为属性的验证工作由 PTA-OZ 模型来完成,同时使得软件模型的实现变得相对简单.

参考文献

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

[2] André C, Mallet F. Specification and verification of time requirements with CCSL and Esterel[J]. ACM SIGPLAN Notices. 2009, 44(7): 167 – 176.

[3] 于苏东,刘雷波,尹首一,等.嵌入式粗颗粒度可重构处理器的软硬件协同设计流程[J].电子学报,2009,37(5):1136 – 1140.
Yu Sudong, Liu Leibo, Yin Shouyi, et al. Hardware-software Co-Design flow for embedded coarse-grained reconfigurable processer[J]. Acta Electronica Sinica, 2009, 37(5): 1136 – 1140. (in Chinese)

[4] Mallet F. Clock constraint specification language: specifying clock constraints with UML/MARTE[J]. Innovations in Systems and Software Engineering. 2008, 4(3): 309 – 314.

[5] Mallet F, De Simone R. MARTE: A profile for RTE systems modeling, analysis and simulation[A]. Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops[C]. Marseille: ACM Press, 2008. 1 – 8.

[6] Woodcock J, Larsen P G, Bicarregui J, et al. Formal methods: Practice and experience[J]. ACM Computing Surveys, 2009, 41(4): 1 – 40.

[7] 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[J]. Journal of Systems Architecture. 2012, 58(5): 178 – 194.

[8] Didier A, Farias A, Mota A. Checking Z data refinements using traces refinement[A]. Proceedings of the Eleventh Brazilian Symposium on Formal Methods[C]. Amsterdam: Elsevier, 2009. 129 – 148.

[9] Derrick J, North S, Simons A J H. Z2SAL-Building a Model Checker for Z[A]. Abstract State Machines, B and Z[C]. Berlin: Springer-Verlag, 2008. 280 – 293.

[10] Rasch H, Wehrheim H. Checking consistency in UML diagrams: classes and state machines[A]. Formal Methods for open, Object-based Distributed Systems[C]. Berlin: Springer-Verlag, 2003. 229 – 243.

[11] Rasch H, Wehrheim H. Checking the validity of scenarios in UML models[A]. Formal methods for open, object-based distributed systems[C]. Berlin: Springer-Verlag, 2005. 67 – 82.

[12] Knapp A, Merz S, Rauh C. Model checking timed UML state machines and collaborations[A]. Formal Techniques in Real-Time and Fault-Tolerant Systems[C]. Berlin: Springer-Verlag, 2002. 395 – 414.

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

[14] Basin D, Olderog E R, Sevinc P E. Specifying and analyzing security automata using CSP-OZ[A]. Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security[C]. New York: ACM Press, 2007. 70 – 81.

[15] 朱敏,李必信,陈乔乔,等.基于微分动态逻辑的 CPS 建模与属性验证[J].电子学报,2012,40(6):1126 – 1132.
Zhu Min, Li Bixin, Chen Qiaoqiao, et al. Transforming hybrid UML to hybrid program for CPS property verification[J]. Acta Electronica Sinica, 2012, 40(6): 1126 – 1132. (in Chinese)

[16] Gotlieb A. TCAS software verification using constraint programming[J]. The Knowledge Engineering Review, 2012, 27(3): 343 – 360.

[17] Kwiatkowska M, Norman G, Segala R, et al. Automatic verification of real-time systems with discrete probability distributions[J]. Theoretical Computer Science, 2002, 282(1): 101 – 150.

[18] Duke R, Rose G, Smith G. Object-Z: A specification language

advocated for the description of standards[J]. Computer Standards & Interfaces, 1995, 17(5): 511 – 533.

[19] Kim S K, Carrington D. A formal metamodeling approach to a transformation between the UML state machine and Object-Z [A]. Formal Methods and Software Engineering[C]. Berlin: Springer-Verlag, 2002. 548 – 560.

[20] Kim S K, Burger D, Carrington D. An MDA approach towards integrating formal and informal modeling languages[A]. Formal Method[C]. Berlin: Springer-Verlag, 2005. 448 – 464.

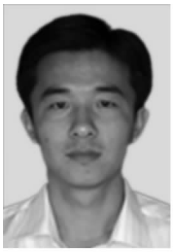
[21] Huzar Z, Kuzniarz L, Reggio G, et al. Consistency problems in UML-based doftware development[A]. UML Modeling Languages and Applications[C]. Berlin: Springer-Verlag, 2005. 1 – 12.

[22] Varró D. Automated formal verification of visual modeling languages by model checking[J]. Software and Systems Modeling. 2004, 3(2): 85 – 113.

[23] 万勇兵,徐中伟,梅萌.一种符号化执行的实时系统一致性测试生成方法[J].电子学报,2013,41(11):2276 – 2284.
Wan Yongbing, Xu Zhongwei, Mei Meng. A symbolic execution method for confromance test generation of real-time system[J]. Acta Electronica Sinica, 2013, 41(11): 2276 – 2284. (in Chinese)

[24] Giese H, Lambers L. Towards automatic verification of behavior preservation for model transformation via invariant checking[A]. Graph Transformations[C]. Berlin: Springer-Verlag, 2012. 249 – 263.

作者简介



许海洋　男,1981 年生于山东威海.博士生,讲师,主要从事形式化建模、模型检测等方面的研究工作.
E-mail: xhy@nuaa.edu.cn

庄　毅(通信作者)　女,教授,博士生导师.主要从事可信计算、信息安全等方面的研究工作.
E-mail: zy16@nuaa.edu.cn