一种结合AADL与Z的嵌入式软件可靠性建模与评估方法

李 蜜 庄 毅 胡镡文
(南京航空航天大学计算机科学与技术学院 南京 211106)

摘 要 在嵌入式软件开发早期,为其建立可靠性模型能够尽早发现软件设计中存在的问题,从而节约嵌入式软件开发成本。AADL从软件结构和故障传播两个角度来建立软件可靠性模型,但是AADL的半形式化性质使得基于AADL建立的可靠性模型难以对可靠性、安全性等非功能属性进行严格的分析与验证。形式规格说明语言Z语言具有很强的逻辑描述能力,能够精确表达软件中的各种约束,这使得基于Z语言建立的可靠性模型能够很好地进行严格的分析和验证。因此,考虑到AADL和Z的特征,文中提出了一种将AADL与Z相结合的形式化可靠性模型(embedded software Reliability Model combined with Z and AADL,ZARM),该模型具有AADL的描述能力和Z的精确性。文中给出了ZARM故障模型、结构模型和行为模型的建模方法,并在谓词中描述了与可靠性相关的数据约束。在ZARM模型的基础上,文中提出了一种面向概率的基于DTMC的可靠性评估方法,来对ZARM模型进行可靠性定量评估和分析。最后,通过一个飞行管理系统对应用ZARM模型进行可靠性建模的过程进行了说明,并采用所提评估方法对其进行了可靠性评估。评估结果与文献[19]结果的对比说明了所提方法的正确性和有效性。

关键词 嵌入式软件,可靠性,AADL,Z语言,DTMC
中图法分类号 TP311.5 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.08.036

Embedded Software Reliability Model and Evaluation Method Combining AADL and Z

LI Mi ZHUANG Yi HU Xin-wen
(College of Computer Science and Technology,Nanjing University of Aeronautics and Astronautics,Nanjing 211106,China)

Abstract In the early stage of embedded software development,a reliability model is established for it to discover problems in software design as early as possible,thereby saving embedded software development costs. AADL establishes software reliability model from two aspects of software structure and fault propagation. However,the semi-formal nature of AADL makes it difficult to analyze and verify the non-functional attributes such as reliability and security. The formal specification language Z language has a strong logical description ability and can accurately express various constraints in the software,which makes the reliability model based on the Z language well rigorously analyzed and verified. Therefore,considering the characteristics of AADL and Z,an embedded software reliability model combined with Z and AADL (ZARM) was proposed. The modeling methods of ZARM fault model,structure model and behavior model were given,and the data constraints related to reliability were described in the predicate. Based on the ZARM model,a probabilistic DTMC-based reliability evaluation method was proposed to quantitatively evaluate and analyze the ZARM model. Finally,the process of reliability modeling using ZARM model was described by a flight management system (FMS),and the reliability evaluation was carried out by using the proposed evaluation method. The comparison between the evaluation results and the reference [19] results shows the correctness and effectiveness of the proposed method.

Keywords Embedded software,Reliability,AADL,Z language,DTMC

1 引言

体系结构分析与设计语言(Architecture Analysis and Design Language,AADL)于2004年由美国汽车工程师协会SAE在MetaH,UML的基础上提出[1]。已有许多专家学者利用AADL来建立软件模型[2-3]。西北工业大学基于AADL的错误模型附件和危害模型附件建立了安全模型,并将该安全模型转换到GSPN模型中,以对软件的安全性进行分析[4]。NASA的Munoz建立了AADL信息流模型,并对其进行了延迟分析,以证明AADL具有可应用于太空系统的潜力[5]。

但AADL作为一种半形式化语言,不利于对可靠性进行严格的分析与验证,因此须对AADL形式语义进行研究。一些学者研究总结出两种AADL形式语义描述方法[6]。

到稿日期:2018-07-06 返修日期:2018-10-01 本文受国家自然科学基金面上项目(61572253),航空基金XXX专项(2016ZC52030),"十三五"装备预研领域基金(61402420101HK02001)资助。

李 蜜(1993—),男,硕士生,主要研究方向为软件可靠性、形式化方法;庄 毅(1956—),女,教授,博士生导师,主要研究方向为可信计算、形式化方法,E-mail:zy16@nuaa.edu.cn(通信作者);胡镡文(1994—),女,博士生,主要研究方向为软件安全性、形式化方法。

AADL 形式语义描述主要采用转换的方式(Translational Semantics),大致可以分为两类:1)显示描述,采用一种具有精确语义的形式语言来定义 AADL 语义,再依据语义进行转换;2)隐式描述,直接将 AADL 模型转换到另一种形式化模型[7-8]。目前流行的 AADL 可靠性建模验证都是采用隐式描述。但隐式描述存在以下不足:它仅仅假设转换的语义是一致的,但语义描述可能不够精确;模型转换的依据是 AADL 已有的语义,而 AADL 中的一些语义是用自然语言和例子进行解释的,给出的语义不精确,可能导致语义转换不完整。

Z 语言[9]是一种基于一阶谓词逻辑和集合论的形式化规格说明语言,由牛津大学的 Abrial 提出。Z 语言的模式结构分为声明部分与谓词部分,其描述形式有垂直和水平两种。Z 语言问世以来,得到了持续的改进与发展,如 Object-Z[10],Z++[11]等,并已在学术界与工业界取得了许多研究与应用成果[12-13]。目前,虽然 Z 语言与 AADL 结合的研究工作很少,但 Z 语言与其他半形式化建模语言相结合的研究成果[14-15]较多,借鉴这些研究成果可以指导本文 ZARM 模型的设计。

因为软件的故障发生和故障传播都是基于概率的,所以 ZARM 模型将整个软件系统看作一个概率系统。在概率系统中,系统状态的变化只与当前状态有关,这满足马尔可夫性质,因此概率系统的模型检测一般采用马尔可夫链(Markov Chain)作为系统模型。由于 ZARM 模型的状态空间集合是离散的,因此本文采用离散时间马尔可夫链(Discrete Time Markov Chain,DTMC)[16]来对 ZARM 模型进行可靠性的定量评估。已有学者利用 DTMC 来对 AADL 进行可靠性评估。文献[17]将 AADL 可靠性模型的基本元素和错误传播的特殊元素转换到交互式马尔可夫链模型进行可靠性定量分析。文献[18]提出了一个可扩展的可靠性评价框架,该评价框架支持 AADL,UML 在内的多种建模语言,将输入模型自动转换到 IO IMC(Input/Output Interactive Markov Chains)模型,然后基于 CADP 工具进行可靠性分析,并可以进行组合可靠性分析,以支持复杂系统的需求。本文希望基于 ZARM 模型提出一种新的可靠性评估方法。

本文首先提出一种嵌入式软件可靠性模型 ZARM。首先借鉴 AADL 错误模型附件,建立 ZARM 的故障模型。其次以故障模型中的部分元素为基础,提出 ZARM 结构模型和行为模型,并分别从结构和行为两个视角建立嵌入式软件模型。然后研究如何采用 DTMC 来刻画 ZARM 模型,设计了基于 DTMC 的可靠性定量评估方法。最后通过一个示例软件对应用该模型进行可靠性建模的过程进行了说明,并应用本文的可靠性评估方法对该实例的 ZARM 模型进行了可靠性评估,将评估结果与文献[19]的结果进行对比,以说明本文建模方法和评估方法的正确性和有效性。

2 ZARM 模型

首先提出 ZARM 故障模型,以对嵌入式软件中的故障元素进行描述。由于 ZARM 结构模型和行为模型是基于故障模型建立的,因此本文将三者结合来描述软件的可靠性行为。

下面分别给出 ZARM 故障模型、结构模型和行为模型的形式化定义,以模板的形式给出建模元素结构,并给出建模规则的描述。采用模板的意义在于:在建模过程中可通过选择合适的模板生成一类 Z 模式,从而形成形式化的软件规约。

2.1 ZARM 故障模型

AADL 扩展了错误模型附件(Error Model Annex,EMA),用于嵌入式软件的可靠性建模与验证。第 2 版 EMV2(EMA volume2)[3]于 2013 年被提出,相较于 EMV1,其做出了一些改变,其中故障模型不再由类型(error model type)和实现(error model implementation)组合声明,而是通过故障类型、故障传播、故障行为分开说明,并且 EMV2 为故障模型元素添加了属性信息。故障类型为统一说明故障状态、故障事件、故障传播的特征。故障传播表示不同组件处于故障状态时是如何影响其他组件的,被分为传出故障和传入故障。故障行为定义了故障事件、故障状态以及事件是如何影响故障状态的。但 EMV2 是用自然语言和例子对故障模型进行描述的,对此我们将提取 EMV2 的核心概念,用 Z 语言模板给出定义,并建立 ZARM 故障模型。

定义 1 ZARM 故障模型的枚举类型有故障传播点类型 PropagationType、故障类型 ErrorType 和概率分布类型 DistributionType。其中,PropagationType 包括 incoming,outgoing 两种故障传播点类型;ErrorType 包括 ServiceError,TimingRelatedError,ValueError,ReplicationError 4 种故障类型;DistributionType 包括 Fixed,Poisson,Exponential,Normal,Gauss,Weibull 与 Binominal 7 种概率分布类型。ZARM 故障模型的基本类型主要是谓词约束 Predicate。

定义 2 ZARM 故障模型 ZA_{error} 是一个如式(1)所示的三元组:

$$ZA_{error}=(State,ErrorEvent,Epp) \tag{1}$$

其中,*State* 表示状态集合,*ErrorEvent* 表示故障事件,*Epp*(*ErrorPropagationPoint*)表示故障传播点集合。

下面分别给出 ZA_{error} 三元组的具体定义。

(1)状态集合 *State*

状态集合 *State* 包括组件的正常状态和故障状态,并且包含多个按照下面的 Z 模式方式定义的状态,每个状态定义了是否为初始状态(*isInitial*)、状态是否为当前状态(*isArrive*)等属性。

⟨*stateName*⟩*State*

isInitial:ℕ

isArrive:ℕ

isInitial∈{0,1}

isArrive∈{0,1}

其中,⟨ ⟩中的元素为可替换;*State* 为状态固定后缀。下面定义的模式与该模式类似,ℕ 表示自然数。

(2)故障事件 *ErrorEvent*

故障事件需要定义事件是否发生(*isEventOccur*)以及发生的概率分布,*Occurrence* 为概率分布参数。事件默认不发生。具体定义如下:

⟨*eventName*⟩ *ErrorEvent*

Occurrence:ℝ

distributionType:*DistributionType*

isEventOccur:ℕ

Occurrence= 0..1

isEventOccur∈{0,1}

其中,$\mathbb{R}$ 表示实数集合,0..1 表示取值范围为[0,1]区间。

(3)故障传播点 *Epp*

故障传播点须说明其所属传播点类型 PropagationType,也要定义是否发生(*isEppOccur*)。故障传播点默认不发生。具体定义如下:

⟨*EppName*⟩ *Epp*

propagationType:PropagationType

Occurrence:$\mathbb{R}$

isEppOccur:$\mathbb{N}$

isEppOccur$\in\{0,1\}$

2.2 ZARM 结构模型

结构模型就是从体系结构的角度来描述嵌入式软件。嵌入式系统包括应用软件、计算机执行平台、物理系统等体系结构,将应用软件、计算机执行平台和物理系统分别映射至 AADL 的应用软件组件、执行平台组件和综合组件,这属于基于 AADL 的嵌入式系统架构级建模层面。

定义 3　ZARM 结构模型 $ZA_{\text{structure}}$ 是一个如式(2)所示的四元组:

$$ZA_{\text{structure}}=(EleType,CE,Com,ComCon) \tag{2}$$

其中,*EleType*(*ElementType*)表示组件元素类型集合,*CE*(*ComponentElement*)表示组件元素集合,*Com*(*Component*)表示组件集合,*ComCon*(*ComponentConnection*)表示组件连接集合。

(1)组件元素类型 *EleType*

组件元素类型即 AADL 组件结构的组成元素,包括端口、访问、绑定等。具体定义如下:

[*EleType*]

EleType=$\{Port,Access,Binding,Bindings\}$

(2)组件元素 *CE*

一个组件可以包含一个组件元素或者多个组件元素,这些组件元素可以是不同类型也可以是相同类型。每种元素都可以与零个或者多个故障传播点绑定。*occurEpp* 表示元素当前绑定的故障传播点,若未绑定故障传播点则为 Ø。具体定义如下:

⟨*elementName*⟩ *CE*

elementType:*EleType*

occurEpp:*F Epp*

Conditions:seq *Predicate*

其中,*F* 表示所有有穷子集的集合;seq 表示 Z 语言中的序列类型。

(3)组件 *Com*

组件包含上述组件元素,并须对组件所处故障状态进行描述,另外对此组件包含的子组件须进行说明。

⟨*comName*⟩ *Com*

⟦Ξ⟨*elementName*⟩*CE*⟧

Δ|Ξ⟨*stateName*⟩*State*

⟦*subCE*:*F Com*⟧

Conditions:seq *Predicate*

其中,⟦ ⟧表示所包含的元素可有零个或者多个;Ξ 用于声明状态空间,即在⟨*comName*⟩*Com* 模式中包含⟨*stateName*⟩*State* 的所有变量;Δ 用于同时声明状态空间的前状态与后状态,即状态空间中的每个变量都声明其前状态值与后状态值(在变量后加'表示其后状态值),在谓词约束中,以前状态值与后状态值的关系来描述操作前后变量的变化情况。

(4)组件连接 *ComCon*

组件连接为控制和数据信息在组件之间的传输提供了路径,也为故障在组件之间的传播提供了可能。组件连接 *ComCon* 表示两个组件所包含的组件元素之间存在连接,因此需要声明 *ComCon* 所连接的两个组件元素 *CE*。

⟨*comconName*⟩*ComCon*

⟦⟨*elementName*⟩*CE*⟧

Conditions:seq *Predicate*

2.3 ZARM 行为模型

软件的行为一般指被执行的操作以及这些操作对软件状态的改变,而在 ZARM 行为模型中软件的行为特指故障事件 *ErrorEvent* 导致组件状态发生的转移和故障在组件之间的传播。利用 ZARM 故障模型中的故障传播点 *Epp* 来描述故障传播的条件与路径,从而建立 ZARM 行为模型。

定义 4　ZARM 行为模型的基本类型只有谓词约束 *Predicat*。

定义 5　ZARM 行为模型 ZA_{behavior} 是一个如式(3)所示的三元组,其中 *CSTransition*(*ComponentStateTransition*)代表组件状态转移,*EPFlow*(*ErrorPropagationFlow*)表示故障传播流,*EP*(*ErrorPropagation*)代表故障传播。

$$ZA_{\text{behavior}}=(CSTransition,EPFlow,EP) \tag{3}$$

(1)组件状态转移 *CSTransition*

组件状态转移 *CSTransition* 中需要声明源状态 *sState* 和目标状态 *tState*。若组件状态发生转移是故障事件⟨*eventName*⟩*ErrorEvent* 导致的,则还须声明故障事件。定义方法如下:

⟨*CSTransitionName*⟩*CST*

⟦Ξ⟨*eventName*⟩ *ErrorEvent*⟧

sState:*State*

tState:*State*

preConditions:seq *Predicate*

postConditions:seq *Predicate*

(2)故障传播流 *EPFlow*

故障传播需要路径,故障传播路径由相连接的故障传播点组成,由两个故障传播点连接成的路径用故障传播流表示。用 *sEpp* 表示 *Flow* 的起点,即 outgoing 类型的故障传播点;用 *tEpp* 表示 *Flow* 的终点,即 incoming 类型的故障传播点。定义方法如下。

⟨*flowName*⟩*Flow*

sEpp:*Epp*

tEpp:*Epp*

(3)故障传播 *EP*

故障从源组件到目标组件的传播过程中,需要所经过的组件 *Com* 含有已与故障传播点 *Epp* 绑定的组件元素,且故障传播点之间存在 *Flow*。故障传播发生后,目标组件的状态将被改变。具体定义如下:

⟨*EPName*⟩ *EP*

⟦⟨*comName*⟩*Com*⟧

〚〈*flowName*〉*Flow*〛

preConditions:seq *Predicate*

postConditions:seq *Predicate*

2.4 ZARM 模型分析

ZARM 故障模型、结构模型与行为模型之间的关系如图 1 所示。ZARM 故障模型主要定义了软件中描述故障传播和故障行为的故障元素,包括故障状态、故障事件、故障传播点、操作及各类约束;ZARM 结构模型则对 AADL 组件的结构元素进行了描述,并说明元素与故障传播点的绑定关系,以及组件所处状态;ZARM 行为模型基于故障传播点说明故障传播所经历的连接以及组件的状态转移。ZARM 模型的建立并不是直接在 AADL 建模元素上添加形式化语义,而是对 AADL 的基本结构进行映射,并在模型中添加描述相关约束的元素与结构。

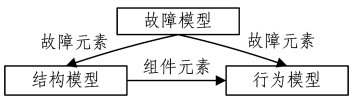

图 1 ZARM 模型关系

Fig. 1 Relationship of ZARM model

3 基于 DTMC 的可靠性评估方法

3.1 离散时间马尔可夫链 DTMC

由于故障行为与故障传播具有基于概率发生的特性,且系统状态的变化只受当前状态的影响,因此可以将本文研究的系统视为满足马尔可夫性质的随机系统。而马尔可夫链是针对随机系统的模型检测常用的概率模型检测方法。由 ZARM 故障模型的定义可知,ZARM 的状态空间集合是离散的。因此本文基于 DTMC 对 ZARM 模型进行可靠性评估,为此本文对 DTMC 进行了定义,以便能更好地对 ZARM 模型进行可靠性评估。

定义 6 DTMC 是一个如式(4)所示的四元组:

$$DTMC=(S_M,S_{in},\boldsymbol{A},T_m) \tag{4}$$

其中:1)S_M 是有限状态空间集合,对应 ZARM 模型中组件状态,包括故障状态和正常状态;2)$S_{in}\in S_M$ 是初始状态,对应 ZARM 模型中的组件的正常状态;3)$\boldsymbol{A}=[a_{ij}]$是状态转移矩阵,$a_{ij}\in[0,1]$表示状态 $s_i\in SM$ 到状态 $s_j\in SM$ 的转移概率;4)$T_m\subseteq S_M\times S_M$ 是状态转移关系集合,对应 ZARM 中的状态转移。

由定义 6 可知,DTMC 集中描述了 ZARM 模型的状态及状态转移关系,显然这能够很好地对 ZARM 模型中单个组件的可靠性进行描述,而 ZARM 模型中的故障传播也反映在组件的状态转移中,因此 DTMC 可以完整地述 ZARM 模型中的可靠性约束,并对其进行定量的可靠性评估。

3.2 DTMC 的可靠性评估

本文将系统处于正常状态的概率作为系统的可靠性评估指标。引申至 DTMC 中,假设 S_{in} 是 DTCM 的初始状态和正常状态,系统的可靠性评估指标即为 DTMC 从 S_{in} 开始经过一系列的状态转移后,处于 S_{in} 的概率。

假设概率向量 $\boldsymbol{P}=[p_i]$表示 DTMC 的状态概率向量,其中 p_i 表示 DTMC 处于 s_i 的概率,可知 DTMC 的初始概率向量为 $\boldsymbol{P}_0=[1,0,0,\cdots,0]$,表示初始条件下 DTMC 必定处于 S_{in}。

DTMC 满足马尔可夫链的平稳分布性质,平稳分布性质的定义如下:

马尔可夫链的状态转移矩阵和状态概率向量分别为 $\boldsymbol{A}$ 和 $\boldsymbol{P}$,那么存在一个概率分布向量 $\boldsymbol{P}_{\text{stable}}$,使得式(5)成立。

$$\begin{cases}\boldsymbol{P}\boldsymbol{A}^n=\boldsymbol{P}_{\text{stable}}\\ \boldsymbol{P}_{\text{stable}}\boldsymbol{A}=\boldsymbol{P}_{\text{stable}}\end{cases} \tag{5}$$

其中,n 表示状态转移次数,$\boldsymbol{P}_{\text{stable}}$ 表示 DTMC 的平稳分布概率向量,即 DTMC 处于稳定状态下的概率分布。从式(5)可以看出,$\boldsymbol{P}_{\text{stable}}$ 与初始状态概率 $\boldsymbol{P}$ 无关。DTMC 中对转移矩阵 $\boldsymbol{A}$ 进行了定义,且 DTMC 的初始状态概率可以设为 $\boldsymbol{P}_0=[1,0,0,\cdots,0]$。求解式(5)即可得到 DTMC 的平稳状态概率分布。本文将 DTMC 平稳状态概率分布中正常状态的概率作为可靠性定量评估指标。

4 ZARM 建模与评估方法应用实例

文献[19]采用了一个飞行管理系统(Flight Management System,FMS)作为实例,描述了 AADL 可靠性的建模过程,并将其转换到广义随机 Petri 网进行可靠性评估。本文也采用此 FMS 作为实例来对 ZARM 模型的建模过程和评估方法进行演示说明,在设置相同概率参数的条件下,将评估结果与文献[19]中的评估结果进行对比。

4.1 实例系统描述

根据功能划分,FMS 有 6 个线程组件,分别是导航传感器处理线程(Navigation Sensor Processing,NSP)、综合导航线程(Integrated Navigation,INav)、制导处理线程(Guidance Processing,GP)、飞行性能优化线程(Aircraft Performance Calculation,APC)、飞行规划线程(Flight Plan Processing,FPP)和数据传输(Period IO,PIO)。在 6 个线程组件中,只有 PIO 与外界存在交互,因此从外界的角度来看,PIO 的可靠性就代表 FMS 的可靠性。各个线程之间的数据传输如下所述。首先将 FMS 的输入数据在 NSP 线程上进行基本处理;然后将处理后的数据传输给 INav 线程并产生综合导航数据传输给 GP,APC 和 PIO 这 3 个线程;其次将经过 GP 和 APC 线程处理后的制导和性能优化的数据交给 FPP 线程,得出飞行计划数据;最终,所有输出给其他子系统的数据都经过 PIO 线程传输。图 2 为 FMS 的 AADL 结构模型(以 FMS 的内部为视角)。

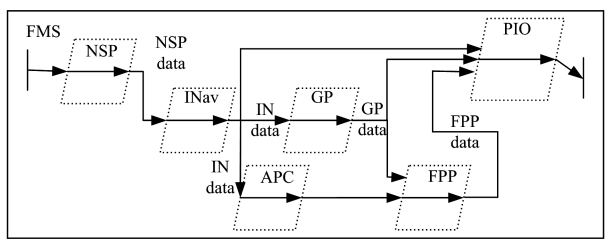

图 2 FMS 的 AADL 架构

Fig. 2 AADL architecture of FMS

4.2 FMS 的 AADL 可靠性模型

以 NSP 为例建立 FMS 的 AADL 可靠性模型,如图 3 所示。其中,为 NSP 定义一个包含两个失效事件 FailEvent 和重启事件 RestartEvent 的故障模型,并为这两个事件分别设置了发生概率,同时设置了两个状态(正常状态 Normal、故障

状态 Failed)和两个状态转移关系(FailEvent 导致 Normal 向 Failed 转移、RestartEvent 导致 Failed 向 Normal 转移)。在 NSP 的结构模型中,定义了一个输入接口 get 和一个输出接口 send,并将 send 设置为故障传播输出点,同时 send 也是故障传播路径 NSPtoINav 的源组件。

与 NSP 的 AADL 模型建立类似,可以为 FMS 其余 5 个线程建立 AADL 模型。每个线程的可靠性相关参数设置如表 1 所列,其中的概率分布均为 *Poisson*。故障传播参数的设置如表 2 所列。

表 1　FMS 组件的可靠性参数

Table 1　Reliability parameters of FMS component

事件	概率	事件	概率	事件	概率
NSP FailEvent	0.0005	GP FailEvent	0.0005	FPP FailEvent	0.0002
NSP RestartEvent	0.1	GP RestartEvent	0.1	FPP RestartEvent	0.1
INav FailEvent	0.001	APC FailEvent	0.001	PIO FailEvent	0.0005
INav RestartEvent	0.1	APC RestartEvent	0.1	PIO RestartEvent	0.1

表 2　FMS 组件故障传播的可靠性参数

Table 2　Reliability parameters of FMS component fault propagation

传播路径	概率	传播路径	概率
NSPtoINav	0.80	APCtoFPP	0.65
INavtoGP	0.75	GPtoPIO	0.72
INavtoAPC	0.75	INavtoPIO	0.75
GPtoFPP	0.70	FPPtoPIO	0.50

4.3　ZARM 故障模型的建立

以 NSP 的 ZARM 建立为例来演示 ZARM 模型的建立过程。

导航传感器线程 NSP 的 AADL 模型如下所示。

```
error behavior simple
event
    FailEvent:error event;
    RestartEvent:repair event;
states
    Normal:initial state;
    Failed:state;
transitions
    FailTransition:Normal-[FailEvent]->Failed;
    RestartTransition:Failed-[RestartEvent]->Normal;
properties
    EMV2::OccurrenceDistribution =>
    [ProbabilityValue =>5.0×10^-4;Distribution => Poisson;]
    applies to FailEvent;
    EMV2::OccurrenceDistribution =>
    [ProbabilityValue =>1.0×10^-1;Distribution => Poisson;]
    applies to RestartEvent;
end behavior
device NSP
features
    get:in data port;
    send:out data port;
annex EMV2{**
    use types ErrorModelLibrary;
    use behavior autopiolt::simple;
    error propagations
    send:out propagation {BadValue};
    flows
    NSPtoINav:error source send {BadValue};
    end porpagations;
    component error behavior
    transitions
    BadValueTransition:
    Normal-[gent]->Failed;
    propagations
    Failed-[]-> send{BadValue};
    end component;
**};
end NSP;
```

由此可以得到如图 3 所示的 NSP 的状态转移图。

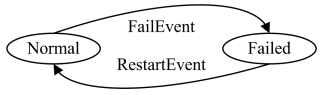

图 3　NSP 的状态转移图

Fig. 3　State transition diagram of NSP

根据 AADL 模型和状态转移图可以建立 ZARM 故障模型,具体定义如下。

(1)状态集合 *State*

从图 3 可以看出,NSP 具有两个状态,即正常状态(Normal)和失效状态(Failed),因此,以 NSP 的状态定义为例进行 ZARM 故障模型的状态空间定义的说明。具体定义如下。

NormalState

$isInitial:\mathbb{N}$

$isArrive:\mathbb{N}$

$isInitial = 1$

$isArrive = 1$

FailedState

$isInitial:\mathbb{N}$

$isArrive:\mathbb{N}$

$isInitial=0$

$isArrive = 0$

(2)故障事件 *ErrorEvent*

从图 3 可以看出,NSP 的故障模型具有两个故障事件失效事件 FailEvent 和重启事件 RestartEvent,再结合 AADL 模型中的故障事件参数,两个故障事件的具体定义如下。

FailEvent

$Occurrence:\mathbb{R}$

$distributionType:DistributionType$

$isEventOccur:\mathbb{N}$

$Occurrence=5.0\times10^{-4}$

$distributionType = Poisson$

$isEventOccur=0$

RestartEvent

$Occurrence:\mathbb{R}$

$distributionType:DistributionType$

$isEventOccur:\mathbb{N}$

$Occurrence= 1.0\times10^{-1}$

$distributionType=Poisson$

$isEventOccur=0$

(3)故障传播点 *Epp*

在 NSP 的 AADL 模型中定义了一条故障传播流，其涉及到 2 个故障传播点，如 NSP 的输出接口 send 被定义为一个 outgoing 类型的故障传播点，与之对应的 incoming 类型的故障传播点是 INav 的数据输入接口 getINav。具体定义如下。

sendOutEpp
propagationType:*PropagationType*
Occurrence:$\mathbb{R}$
isEppOccur:$\mathbb{N}$
propagationType=*outgoing*
Occurrence=0.8
isEppOccur=0
getINavInEpp
propagationType:*PropagationType*
Occurrence:$\mathbb{R}$
isEppOccur:$\mathbb{N}$
propagationType=*incoming*
Occurrence=0.8
isEppOccur=0

4.4 ZARM 结构模型的建立

(1)组件元素 *CE*

从 NSP 的 AADL 模型可以看出，NSP 包含两个 port 类型的组件元素，分别是数据传入接口 get 和数据传出接口 send，并且 send 和一个 outgoing 类型的故障传播点绑定。具体定义如下。

getInCE
elementType:*EleType*
occurEpp:*F Epp*
elementType=*Port*
sendOutCE
elementType:*EleType*
occurEpp:*F Epp*
elementType=*Port*
occurEpp={*sendOutEpp*}

其中，*sendOutEpp* 为故障模型中定义的 outgoing 类型的故障传播点。

(2)组件 *Com*

线程 NSP 包含上述两个组件元素，还包含组件的两个状态 Normal 和 Failed。状态模式的变量 *isArrive* 设置为 1 表示组件当前处于该状态，否则 *isArrive* 为 0。具体定义如下。

NSPCom
Ξ*getInCE*
Ξ*sendOutCE*
Δ*NormalState*
Δ*FailedState*
NormalState.*isArrive*=1
FailedState.*isArrive*=0

4.5 ZARM 行为模型的建立

ZARM 行为模型描述组件的状态转移和故障在组件之间的传播。本文根据 NSP 的故障模型、结构模型及图 3 所示的状态转移图，给出 NSP 行为模型。具体定义如下。

(1)组件状态转移 *CSTransition*

当故障事件 FailEvent 发生时，NSP 的状态会由 Normal 转移到 Failed；当故障事件 RestarEventt 发生时，NSP 的状态会由 Failed 恢复到 Normal。具体定义如下。

NSPNtoFCST
sState:*State*
tState:*State*
FailEvent
sState=*NormalState*
tState=*FailedState*
if *FailEvent*.*isEventOccur*=1 then
  sState′.*isArrive*=0
  tState′.*isArrive* =1
NSPFtoNCST
sState:*State*
tState:*State*
RestartEvent
sState=*FailedState*
tState=*NormalState*
if *RestartEvent*.*isEventOccur*=1 then
  sState′.*isArrive*=0
  tState′.*isArrive*=1

(2)故障传播流 *Flow*

NSP 的 AADL 模型定义了一条故障传播流 *NSPtoINav*，来表示 NSP 向 INav 传播故障。使用 ZARM 故障传播流 *Flow* 的具体定义如下。

NSPtoINavFlow
sEpp=*sendOutEpp*
tEpp=*getINavInEpp*

(3)故障传播 *EP*

故障从 NSP 的 sendOutEpp 经由故障传播流 *NSPtoINavFlow* 传播到 INav 的 *getINavInEpp*，使得 INav 变为 Failed 状态。具体定义如下。

NSPtoINavEP
NSPCom
Δ*INavCom*
Δ*NSPtoINavFlow*
if *NSPCom*.*Failed*.*isArrive*=1 or
  NSPtoINavFlow.*sEpp*.*isEppOccur*=1 then
    NSPtoINavFlow.*sEpp*.*isEppOccur*=1
    NSPtoINavFlow.*tEpp*.*isEppOccur*=1
    INavCom.*Failed*.*isArrive*=1
    INavCom.*Normal*.*isArrive*=0

至此，已经结合 FMS 中的 NSP 组件对 ZARM 的故障模型、结构模型和行为的建模过程进行了说明。

4.6 ZARM 可靠性评估过程

仍然以 NSP 为例来说明 ZARM 模型的评估过程。通过前面的 ZARM 建模过程，已经为 FMS 建立了 ZARM 模型。从 NSP 的 ZARM 模型中提取可靠性信息，为 NSP 建立 DTMC 模型，$DTMC=(S_M,S_m,A,T_m)$，其中：

(1)$S_M=\{Normal,Failed\}$；

(2) $S_{in}=Normal$；

(3) $\boldsymbol{A}=\begin{bmatrix}1-0.0005 & 0.0005\\ 0.1 & 1-0.1\end{bmatrix}$；

(4) $T_m=\{(\int Normal, Normal),(Normal, Failed),(Failed, Failed),(Failed, Normal)\}$；

设 NSP 的初始概率分布为 $\boldsymbol{P}_0=[1,0]$，将 $\boldsymbol{P}_0$ 与 $\boldsymbol{A}$ 代入式(5)，可得如下方程组，

$$\begin{cases}[1,0]\begin{bmatrix}1-0.0005 & 0.0005\\ 0.1 & 1-0.1\end{bmatrix}^n=\boldsymbol{P}_{\text{stable}}\\ \boldsymbol{P}_{\text{stable}}\begin{bmatrix}1-0.0005 & 0.0005\\ 0.1 & 1-0.1\end{bmatrix}=\boldsymbol{P}_{\text{stable}}\end{cases}$$

利用 MATLAB 求解上述方程可得：

$\boldsymbol{P}_{\text{stable}}=[0.9950248, 0.0049754]$

即 NSP 的可靠性为 0.9950248。利用同样的方法可以求得 FMS 其余组件的可靠性，如表 3 所列。

表 3　FMS 各组件的可靠性

Table 3　Reliability of FMS components

组件	可靠性(本文)	可靠性(文献[19])
NSP	0.9950248	0.9950249
INav	0.9822576	0.9822720
GP	0.9763785	0.9763780
APC	0.9725823	0.9725944
FPP	0.9512731	0.9512654
PIO	0.9309105	0.9310102
SYSTEM	0.9309105	0.9310102

从表 3 的可靠性评估对比结果可以看出，采用本文所提 ZARM 模型和评估方法得到的可靠性定量评估结果与文献[19]得到的结果十分接近，证明了本文方法的正确性和有效性。

在文献[19]中，使用 AADL 作为实例建立可靠性模型之后，需要将其转换为广义随机 Petri 来进行可靠性评估，而这个转换过程较为复杂，需要研究两种模型之间的转换规则，该转换过程有可能导致模型信息丢失。而对于本文的 ZARM，建立之后不需要对其进行转换，能够使用本文的评估方法进行可靠性评估，提高了评估效率，简化了评估过程。这说明了本文的 ZARM 模型的优越性。

结束语　本文针对实时嵌入式软件可靠性建模与评估问题，提出了一种嵌入式软件模型 ZARM，给出了包括故障模型、结构模型与行为模型的 ZARM 模型形式化定义。另外文中最后通过一个示例软件的建模过程对 ZARM 模型的应用进行了说明。可以看到，本文定义的 ZARM 模型可以为软件故障传播过程中的各类元素定义严格的数据约束，弥补了现有的形式化模型在可靠性约束规约方面的不足。在 ZARM 模型的基础上，本文提出了基于 DTMC 的 ZARM 模型可靠性评估方法，以对嵌入式软件的可靠性进行定量评估和分析。

本文提出的 ZARM 模型还需手动进行建模，无法依靠已有的 AADL 模型直接得到。因此，我们后续的研究工作旨在研究 AADL 模型到 ZARM 模型的自动转换机制，包括两种模型之间的转换规则和转换工具的实现，以充分利用 AADL 的易用性和 ZARM 的准确性、严谨性。

参 考 文 献

[1] FEILER P H, GLUCH D P, HUDAK J J. The architecture analysis & design language (AADL): An introduction[R]. Pittsburgh: Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst, 2006.

[2] LI J Y, CAO Z N. An AADL Modeling Method Based on Concurrency[J]. Computer & Modernization, 2017(5): 1-4.

[3] ZHOU D X, LI N, LIU Z X. Modeling and Analysis of Avionics Configurations Control System Based on AADL[J]. Computer Science, 2016, 43(S1): 55-59.

[4] WEI X, DONG Y W, YANG M, et al. Hazard analysis for AADL model[C]// Proceedings of the 2014 IEEE 20th International Conference on Embedded and Real-Time Computing Systems and Applications. Chongqing, China: IEEE, 2014: 1-10.

[5] MUNOZ M. Space systems modeling using the Architecture Analysis & Design Language (AADL)[C]// 2013 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW). Pasadena: IEEE, 2013: 97-98.

[6] YANG Z B, PI L, HU K, et al. AADL: An architecture design and analysis language for complex embedded real-time systems [J]. Journal of Software, 2010, 21(5): 899-915.

[7] ZHANG X C, YAN X F, ZHOU Y. A Method for Conversion of AADL Model into Dynamic Fault Tree[J]. Computer Technology and Development, 2017, 27(11): 110-114.

[8] LI D M, LI J, LIN H F. Reliability Analysis Method of Embedded System AADL Model Based on Fault Tree Analysis[J]. Computer Science, 2017, 44(6): 182-188.

[9] SPIVEY J M, ABRIAL J. The Z notation [M]. Hemel Hempstead: Prentice Hall, 1992.

[10] SMITH G. The Object-Z specification language [M]. New York: Springer Science & Business Media, 2012.

[11] LANO K, HAUGHTON H. The z++ manual [M]. Lloyds Register of Shipping, 1994: 29-62.

[12] FISCHER C. CSP-OZ: a combination of Object-Z and CSP[M]// Formal Methods for Open Object-based Distributed Systems. Springer, Boston, MA, 1997: 423-438.

[13] SMITH G, LI Q. Maze: An extension of object-z for multi-agent systems[C]// International Conference on Abstract State Machines, Alloy, B, TLA, VDM, and Z. Berlin Heidelberg: Springer, 2014: 72-85.

[14] NAJAFI M, HAGHIGHI H. An integration of UML-B and object-Z in software development process [J]. Lecture Notes in Electrical Engineering, 2013, 152: 633-648.

[15] NI S R, ZHUANG Y, CAO Z N, et al. Modeling dependability features for real-time embedded systems [J]. IEEE Transactions on Dependable and Secure Computing, 2015, 12(2): 190-203.

[16] ABRAHAM E, JANSEN N, WIMMER R, et al. DTMC model checking by SCC reduction[C]// 2010 Seventh International Conference on the Quantitative Evaluation of Systems. Williamsburg: IEEE, 2010: 37-46.

[17] CHENG Y H, HUANG Z Q, KAN S L. A system dependability modeling method using AADL and IMC[J]. Computer Engineering & Science, 2015, 37(8): 1517-1524.

[18] BOUDALI H, CROUZEN P, HAVERKORT B, et al. Arcade-A formal, extensible, model-based dependability evaluation framework[C]// 13th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2008). Belfast: IEEE, 2008: 243-248.

[19] GAO Z W. Reliability Modeling and Evaluation of Embedded Software Based on AADL[D]. Xi'an: Xidian University, 2011.