

基于 SysML 的嵌入式软件系统建模与验证方法研究

仵林博¹, 陈小红², 彭艳红¹, 聂长海³

(1. 中国工程物理研究院计算机应用研究所, 四川 绵阳 621999; 2. 华东师范大学 计算机科学与软件工程学院, 上海 200062;
3. 南京大学 计算机科学与技术系, 南京 210023)

摘 要: 嵌入式软件系统由于广泛采用分布式异构网络, 使得软件系统复杂性呈现几何增长, 因此需要在系统设计的论证阶段, 对系统需求和设计方案进行正确性和充分性验证, 进而发现系统性设计缺陷, 避免可能引发或导致的严重系统性问题。为此, 提出一种基于 SysML 的嵌入式软件系统结构与行为需求建模方法。通过对嵌入式软件系统的结构和逻辑行为进行层次化建模, 并利用 Modelica 离散与连续融合的仿真特点, 在系统需求论证阶段, 基于建立的仿真模型对系统关键功能指标进行仿真, 结果表明, 该方法对系统需求论证具有可行性。

关键词: SysML 模型; 系统建模; Modelica 仿真; 系统结构; 嵌入式系统

开放科学(资源服务)标识码(OSID):



中文引用格式: 仵林博, 陈小红, 彭艳红, 等. 基于 SysML 的嵌入式软件系统建模与验证方法研究[J]. 计算机工程, 2019, 45(1): 1-8.

英文引用格式: WU Linbo, CHEN Xiaohong, PENG Yanhong, et al. Research on system modeling and verification method of embedded software based on SysML[J]. Computer Engineering, 2019, 45(1): 1-8.

Research on System Modeling and Verification Method of Embedded Software Based on SysML

WU Linbo¹, CHEN Xiaohong², PENG Yanhong¹, NIE Changhai³

(1. Institute of Computer Application Research, Chinese Academy of Engineering Physics, Mianyang, Sichuan 621999, China;
2. School of Computer Science and Software Engineering, East China Normal University, Shanghai 200062, China;
3. Department of Computer Science and Technology, Nanjing University, Nanjing 210023, China)

【Abstract】 With the widespread use of distributed heterogeneous networks in embedded software systems, the complexity of software systems increases geometrically. It is necessary to verify the correctness and sufficiency of system requirements and design schemes in the demonstration stage of system design, and discover system design flaws to avoid possible systemic problems. Therefore, a demand modeling method based on SysML for embedded software system structure and behavior is proposed. Through the hierarchical modeling of the structure and logic behavior of the embedded software system, and using the simulation characteristics of Modelica discrete and continuous fusion, in the system requirements demonstration stage, based on the established simulation model, the key functional indicators of the system are simulated and verified. Results show that the method is feasible and useful for requirements demonstration.

【Key words】 SysML model; system modeling; Modelica simulation; system structure; embedded system

DOI: 10.19678/j.issn.1000-3428.0052826

0 概述

随着网络和通信技术的快速发展, 嵌入式系统从以单片微处理器为核心的简单嵌入式系统, 逐渐发展成为通过外部总线、高速或低速网络连接起来的分布式系统^[1], 该系统称为网络化嵌入式软件系

统。与传统的软件系统相比, 网络化嵌入式软件系统通常由很多节点组成, 采用分布式网络互联, 有些系统甚至内嵌在一个更大的系统中, 形成规模更加庞大的网络结构。随着系统规模的增加, 系统交互复杂度越来越高, 系统设计时不仅考虑内部节点之间存在各种交互行为, 还要通过接口与周围环境中

基金项目: 国防基础科研计划重点项目(JCKY2016212B004)。

作者简介: 仵林博(1983—), 男, 工程师, 主研方向为软件建模与验证; 陈小红, 副教授; 彭艳红, 工程师; 聂长海, 教授。

收稿日期: 2018-10-09 修回日期: 2018-11-28 E-mail: wulinbo@caep.cn

的节点进行交互通信,特别是不同节点的并发生行为,这些行为之间需要同步完成特定功能。从上述特点可以看出,网络化嵌入式软件系统在系统设计前期,即需求论证阶段,缺乏一套系统的建模与验证方法,需通过对嵌入式软件系统进行建模来验证系统功能正确性和业务逻辑一致性。

在嵌入式软件系统建模方面,通常使用 UML、AADL^[2-3]、EASTADL^[4]等语言对系统结构进行建模和分析。系统行为建模主要有面向状态、面向活动、面向连续与离散等不同的行为模型。文献[5]针对安全关键场景,提出软件交互行为模型,面向状态的模型主要基于有限自动机 Stateflow^[6-8],但是,这些建模工作局限于系统的某个方面,没有解决系统化的建模过程,理想的建模方法应该支持系统从上至下层次化建模,能够从不同角度、不同粒度对系统进行描述^[9-10]。基于模型的验证方法可分为形式化验证和仿真验证,形式化验证方法又可以分为定理证明和模型检测,定理证明是构建一个形式化的公理系统,使用严格的推导证明来验证软件的正确性,模型检测以有限自动机为形式化模型,通过穷尽系统状态空间进行相关性质的检验,它的主要难点在于解决状态空间爆炸问题。仿真验证方法有 Simulink、Modelica 以及 Ptolemy 等,Simulink 用来对动态和嵌入式、非线性实时系统进行多领域建模设计^[11],其目的是建立严格的数学模型,并在此基础上对设计的软件系统动态行为进行仿真。基于 Modelica 语言的建模与仿真可以利用微分方程组对系统行为进行描述并实现仿真运行,支持面向对象、非因果的连续与离散混合建模仿真。文献[12]提出层次异构的思想并通过建立不同的计算模型实现系统的动态行为。形式化验证虽然能够精确地验证系统性质,但适合对规模较小的系统进行验证。针对复杂嵌入式系统,仿真验证是一种行之有效的方法。由于 SysML 是一种扩展 UML 系统建模语言,它涵盖系统建模的需求、结构以及行为 3 个方面,因此,可以建立大型复杂系统的结构、行为等可视化模型。

本文基于 SysML 提出一种系统建模和基于 Modelica 的仿真验证方法,并将 SysML 模型自动转换为 Modelica 模型进行仿真,以验证系统需求的正确性和逻辑一致性。

1 层次化系统建模

由于现代嵌入式系统的复杂程度不断递增,使得系统建模过程变得更加复杂,解决这类问题有效的方法就是建立系统的一个抽象层次结构^[13],层次化的、被清晰定义的抽象层可以有效降低系统的复杂性,系统建模人员可以在不同粒度的层次上创建和管理系统模型。图 1 所示定义了系统 4 层抽象层次结构,从上到下依次为系统层、功能分解层、软件/

硬件分割层和部署层,每层描述的粒度不断增加。

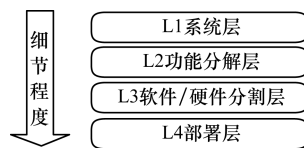


图 1 层次化系统建模架构

层次化系统建模架构包含以下 4 个方面:

1) 系统层:系统层划定系统边界,确定目标系统及其各相关方的作用关系,并确定目标系统任务整体要求,这一层系统被看作是一个黑盒,重点关注系统所处的环境。

2) 功能分解层:在功能分解层,系统被逐步划分为粗粒度的逻辑组件,并且关注每个逻辑组件的内部结构,以及组件与组件之间关联关系和行为交互。

3) 软件/硬件分割层:基于功能分解层的划分,在软硬件分割层,组件将被分解为软件组件和硬件组件,即决定系统的哪些逻辑功能由软件实现,哪些逻辑功能由硬件实现,并且定义每一个软件/硬件组件以及软件/硬件组件之间的关联和交互关系。

4) 部署层:在部署层中,系统到软件/硬件平台的部署方案被确定,部署方案包括决定哪些软件/硬件组件运行在哪些物理平台上以及物理平台之间的关联和交互关系。

在系统建模时,基于系统的抽象层次化结构,采用高内聚低耦合原则^[14],每层都将得到不同的系统模型,图 2 展示的是每一个抽象层建模时得到的模型,模型主要描述系统的 3 个方面:需求,行为场景和体系结构。

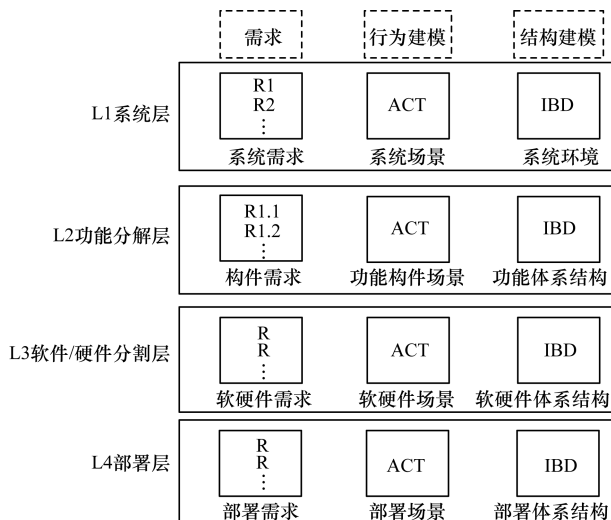


图 2 抽象层次化系统建模结构

抽象层次化结构每一层进行建模的关注点如下:

1) 系统层从系统初始需求出发,系统需求包含系统可提供的外部功能服务、输入输出等,系统行为场景定义系统整体的工作流程,体系结构关注系统

与所处环境的交互关系。

2) 功能分解层将系统逐层展开,所有模型关注系统内部功能构件,构件需求会进一步精化并添加一些针对功能构件的新需求,行为场景主要描述各功能构件之间如何一起完成某个复杂的行为以及单个功能构件内部的行为,体系结构模型包含系统到一组逻辑功能构件的分解以及功能构件之间的连接关系。

3) 软硬件分割层模型重点关注具体的软件构件和硬件构件,软硬件需求、行为场景和体系结构模型会更加关联到软件和硬件的特性。

4) 部署层的模型接近具体的实际情况,部署需求对部署到实际物理单元的软件和硬件进行定义,行为场景描述物理单元以及外部设备之间的交互行为,体系结构模型描述软硬件模块与物理单元的映射。

基于上述的 4 层抽象层次化系统建模结构,在实际建模设计过程中,从系统的初始需求出发,按照 4 个层次将建模设计过程分为 7 个子过程,每一个建模子过程完成不同的建模任务,图 3 描述了系统建模设计的 7 个子过程以及它们之间主要的流程。

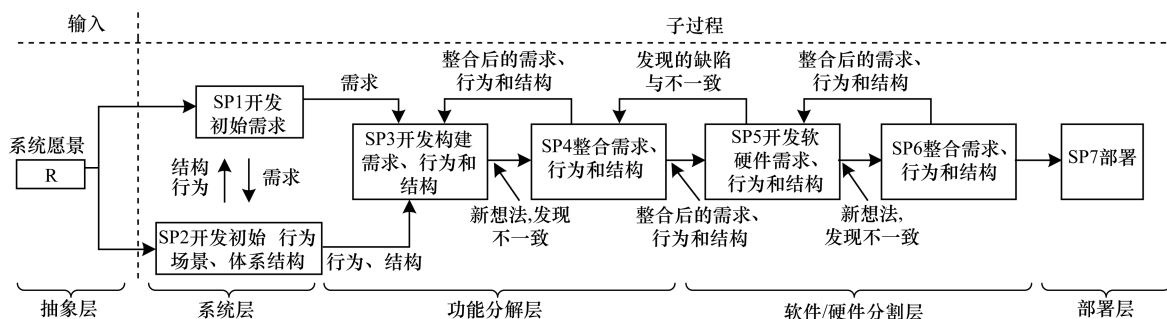


图 3 系统建模子过程

系统原始输入是系统愿景和相关领域知识,每一个子过程的概述如下:

1) 子过程 SP1 从系统层初始需求出发,根据系统愿景和相关利益者的需求,从系统实现的角度描述系统顶层需求。

2) 子过程 SP2 描述系统层初始行为场景和体系结构,初始行为场景描述系统整体的工作流程,初始体系结构主要包括划定系统边界、定义系统接口以及系统与外部环境实体的关联关系。

3) 子过程 SP3 基于 SP1 和 SP2 的输出,在功能分解层进行初始需求、行为场景和体系结构逐步精化和扩展,该过程是一个不断迭代的过程,直到一个合适的粒度,其中精化的需求可以分配到某个功能组件或行为场景上,此时需求、行为和结构之间的一致性可被识别。

4) 子过程 SP4 负责协调和整合跨系统层、功能分解层以及功能分解层层内的需求、行为和结构模型不一致性。

5) 子过程 SP5 基于 SP4 整合后的结果进行软硬件层的需求、行为和体系结构的描述,关注点从功能构件转向硬件或软件构件。

6) 子过程 SP6 与子过程 SP4 类似,该过程负责协调和整合跨功能分解层和软硬件分割层以及软硬件分割层层内的需求、行为和结构模型不一致性。

7) 子过程 SP7 负责描述将软件和硬件构件部署到实际物理单元上的映射关系和交互行为。

2 基于模型的仿真验证方法

针对嵌入式软件系统进行系统建模后,基于系

统的结构与行为模型,依据仿真验证规则进行优化并自动转换为 Modelica 模型进行仿真验证。仿真验证方法如图 4 所示,仿真验证的系统性质主要针对系统需求中功能的正确性和业务逻辑一致性,在验证时为便于仿真验证需要将系统性质进行转化。仿真验证的充分性依赖于系统仿真数据的生成,本文通过模型图中参数变量依赖关系和时间影响关系,设计基于覆盖模型图路径的仿真数据自动生成算法,将生成的仿真测试数据自动导入 Modelica 模型进行仿真验证,设计验证的要点包括 SysML 模型自动转换和测试数据的自动生成。

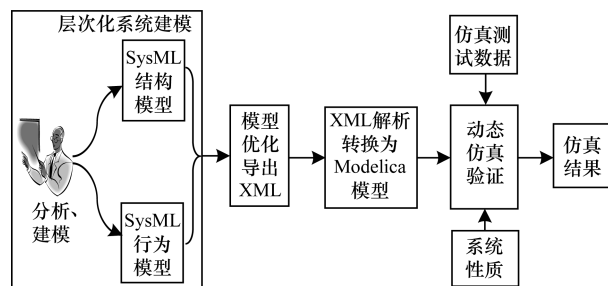


图 4 基于模型的需求验证方法

2.1 SysML 模型到 Modelica 模型的自动转换

本文将系统结构模型和行为模型导出为 XML 文件,由于 XML 文件包含模型图中的各种数据信息,通过使用 DOM4J 解析模型图的 XML 文件,即可读取图中的信息并将其转换成 Modelica 模型中相应的元素。本文仿真验证应用的系统结构模型和行为模型分别是 SysML 模块定义图和活动图。模块定义图显示系统的静态结构信息,其描述的主要元素为模块,

利用模块的值属性来提取实体类型包含的变量,通过定义变量(Variable)和参数(Parameter)这 2 种数据类型来存储数据信息。Variable 数据类型是在解析模块定义图 XML 文件时用来存储 block 中变量的信息,如表 1 所示,包含 2 个属性:name 和 type,name 表示变量名,type 表示变量类型。Parameter 数据类型存储 block 中参数变量的信息。活动图描述的是系统的动态行为信息,其描述的主要元素为活动分区、节点和转移边,这些基本元素信息分别对应于 XML 文件标签中的 group、node 和 edge,将解析后的这些元素信息通过定义活动分区(ActivityPartition)、节点(Node)和转移边(Transition)这 3 种数据类型进行存储,它们包含的属性及对应映射关系如表 1 所示。

表 1 XML 标签与图元素映射关系

类型	图元素	XML 标签	定义数据类型	包含属性
模块	模块	packagedElement	Variable	type, name
定义图	值属性	ownedAttribute	Parameter	type, name, value
	活动分区	group	Activity Partition	name, classname, id
活动图	节点	node	Node	id, name, classname, owner, type
	转移边	edge	Transition	id, source, target, guard

基于上述对模块定义图和活动图 XML 文件的分析以及自定义的数据类型,通过遍历 XML 文件中每一个需要的标签,并将标签下的元素信息依据创建的相应数据类型进行存储。

2.1.1 转换规则

SysML 模块定义图和活动图中元素需要转换为 Modelica 仿真模型中元素,其中,模块定义图中的元素转换为 Modelica 模型声明区内的元素,活动图中的元素主要转换为 Modelica 模型算法区域内的元素。

SysML 模块定义图中每一个 block 元素对应一个 Modelica 模型实例,block 值属性被转换为其对应 Modelica 模型声明区的元素。值属性中的参数将被转换成 Modelica 模型中的参数,包括参数类型、名称以及默认值。值属性中的变量转换成 Modelica 模型的变量,包括变量类型和变量名。

活动图中每一个活动分区元素对应一个 Modelica 模型实例,活动参数对应模型声明区的参数,其余活动图元素将会转换为 Modelica 模型算法区域的元素,基本动作以及发送信号动作转换为方程。等待时间动作节点根据不同类型的时间事件分别转换为不同的元素,绝对时间事件的等待时间动作转换为 when 语句,相对时间事件的等待时间动作需要在声明区先声明一个 Timer 类型实例,之后转换为 if 语句,接受事件动作转换为 Modelica 模型中的 if 语句。控制节点组合形成的控制逻辑,转换成

Modelica 模型不同的逻辑控制语句,活动图中决策节点与其分支转换为 if 语句,合并节点与决策节点组合形成的循环结构转换为 while 语句。

2.1.2 自动转换算法

依据转换规则设计转换算法将模块定义图和活动图转换成 Modelica 仿真模型。主要内容是先调用活动图 XML 文件的解析算法获取所有活动分区、节点以及边集合,对每一个活动分区创建一个 Modelica 模型,调用模块定义图 XML 文件的解析算法获取对应模块的变量以及参数集合,并依次在模型声明区对这些变量和参数进行声明,如果活动图中包含相对时间事件的等待时间动作,则在模型声明区创建一个 Timer 实例,接着从活动分区的开始节点出发向下遍历,并根据节点的类型在模型算法区域转换输出不同的内容,详细的转换算法如算法 1 所示。

算法 1 Modelica 模型转换算法

输入 活动分区集合 AP,节点集合 N,边集合 T,变量集合 V,参数集合 P

输出 Modelica 模型

Procedure SysMLtransformModelica(M) Begin

1. for each ap in AP do

2. 创建 Modelica 模型;

3. 根据变量集合 V 和参数集合 P 依次输出变量以及参数的声明;

4. 获取所有等待时间动作节点的集合 TimerNodes;

5. for each TimerNode in TimerNodes

6. 声明一个 ModelicaTimer 实例;

7. end for

8. 输出“algorithm”;

9. node = 活动分区的初始节点;

10. while(node! = null) do

11. if(节点 node 类型 = “Action”) then

12. 输出“node.name”;

13. else if(node 类型 = “Decision”) then

14. 输出“if + node.name + then”;

15. else if(node 类型 = “AcceptEvent”) then

16. 输出“if + node.name then”;

17. else if(node 类型 = “AcceptEventTimer”) then

18. if 相对时间事件 then

19. 输出“if time > 相对时间 then”;

20. else

21. 输出“when time > 绝对时间 then”;

22. end if

23. else if(node 类型 = “Merge”) then

24. if 该合并节点与决策节点构成循环 then

25. 输出“while + 决策节点 name 属性值 + loop”;

26. else

27. 输出“end if;”(与其对应的决策节点闭合)

28. end if

29. else if(node 类型 = “Fork”) then

30. 每一个并发分支继续调用转换算法;

31. else if(node = “ActivityFinal” or “FlowFinal”) then

```

32. 输出 if 语句、when 语句相匹配的结束语句;
33. break;
34. end if
35. 查找 node 下一个节点 nextNode;
36. node = nextNode;
37. endwhile
38. end for
End Procedure SysMLtransformModelica

```

基于上述的模型转换算法设计一个 Modelica 模型自动转换工具(Model Transformation),该工具的输入为 SysML 活动图和模块定义图的 XML 文件,通过对模型图 XML 文件的解析,基于模型转换算法进行模型转换,输出为仿真文件 ModelicaText,最终将 ModelicaText 导入到 OpenModelica 工具进行联合仿真。

2.2 基于路径的仿真测试数据生成

在生成 Modelica 仿真模型后,仿真测试数据的生成是基于活动图中的路径覆盖,由于活动图本质

上是一个有向图^[15-16],在覆盖活动图路径时考虑 2 种情况:一种是无并发结构的基本路径覆盖;另一种是含有并发结构的并发路径覆盖。通过对这 2 种类型路径覆盖可以得到活动图的所有路径,之后可根据路径生成仿真测试数据。

2.2.1 基本路径覆盖

基本路径覆盖是指采用图论中路径遍历算法,求解从活动图起始点到终点的路径,在路径生成时采用基于深度的路径覆盖策略,如图 5 所示,即以判定节点为分割点对整个活动图进行分割,将活动图分为以判定节点为关联的相邻路径,通过设置相邻路径的覆盖深度来生成满足与被测件风险级别相对应的路径集合。设置覆盖深度越大,即覆盖程度越高,生成的路径越多,如覆盖深度为 1,则生成的路径只需要覆盖活动图中的小路径分支,不考虑路径组合的情况;如覆盖深度为 N ,则生成的路径考虑路径组合的情况。

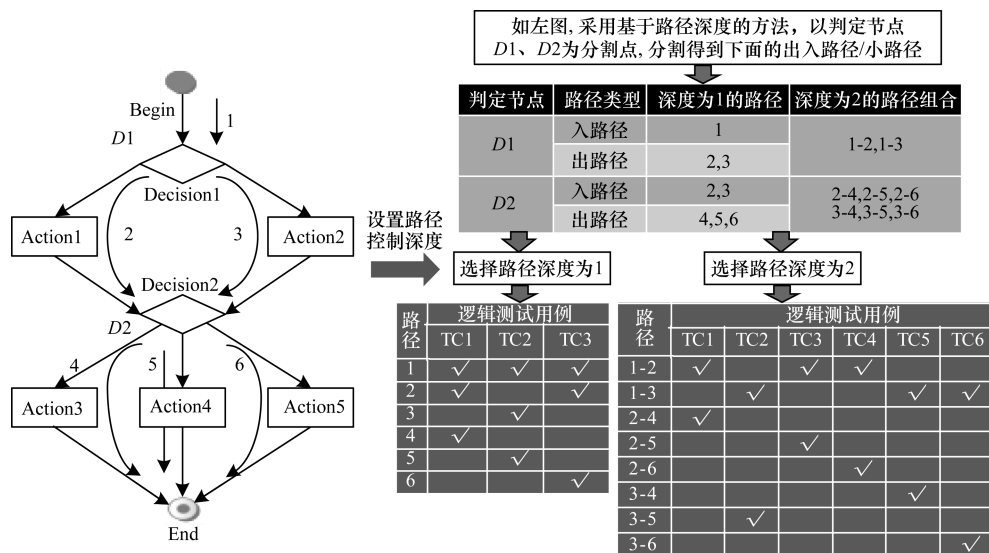


图 5 基于深度的路径生成示意图

2.2.2 并发路径覆盖

针对含有并发结构的活动图,首先依据并发活动图中并发序列的标识识别活动图中的并发结构,将含有并发结构的部分作为一个独立模块进行处理,生成测试路径集合时,不仅要考虑并行的分支数,而且要考虑并发分支执行的先后顺序,即考虑并行度和并行动作组合深度:

1) 并行度:并行的分支数。例如参数为 1,则只要并行分支中有一条分支执行完成,即可生成活动图的测试路径;参数为 2,则需要并行分支中选择 2 条分支执行完成,不考虑 2 条分支的执行顺序关系。

2) 并行动作组合深度:并行路径的顺序关系。例如参数为 2,代表需要考虑 2 条分支的执行先后顺序关系。

在对并发模块进行处理后,对不含并发结构的部分,按照基本路径覆盖策略生成路径,最后将含并发结构生成的路径与不含并发结构部分生成的路径按次序进行组合,生成最终的路径,完成含有并发结构的图活动图路径生成。

2.2.3 基于路径的仿真数据生成

根据上文对活动图中路径覆盖分析,每条路径上各个决策节点选择的分支生成相对应的一组仿真数据,直到完成所有路径上的数据生成。仿真数据生成的关键是根据决策节点上的条件约束确定关键变量的数值^[17],这些数值决定了 Modelica 模型在仿真时所选择的分支以及执行的方程,并最终影响仿真结果,条件约束表达式记录在决策节点 name 属性值(XML 文件中提取),提取 name 字符串中的逻辑运算符{>, <, =, &&, ||, !}以及之后的数值或者

布尔值等进行约束求解,结合转移边 guard 的属性值生成相对应的仿真数据,基于路径的仿真数据生成算法如算法 2 所示。

算法 2 基于路径的仿真数据自动生成算法

输入 SysML 活动图 XML 文件

输出 仿真测试数据

Procedure GenerateBasicPathData () Begin

```

1. 节点 node = 活动图初始节点;
2. while( node != null ) do
3. if( 节点 node 类型 == "Decision" ) then
4. 将节点压入栈 Stack1 中;
5. 选择节点的 true 分支上下一个节点
6. nextNode, 并将监值 "true" 压入栈 Stack2 中;
7. else if( node 类型 == "ActivityFinal" || "FlowFinal" )
8. for i from 0 to Stack1. size-1
9. 调用 generateData ( Stack1 第 i 个元素 name 属性,
Stack2 第 i 个元素 );
10. end for
11. while( Stack1 != null ) do
12. if( Stack1 决策节点 false 分支未覆盖 ) then
13. 选择节点的 false 分支上 nextNode,
14. 弹出 Stack2 元素并将监值压入栈中;
15. else
16. 弹出 Stack1 和 Stack2 的栈顶元素;
17. end if
18. end while
19. else
20. 查找节点的下一个节点 nextNode;
```

```

21. end if
22. node = nextNode;
23. end while
End Procedure GenerateBasicPathData
```

3 应用验证与结果分析

返回舱降落伞系统是载人航天活动的最后阶段^[18],即回收着陆过程,着陆过程中关键一环就是适时控制降落伞的开启,降落伞开启的工作过程由一系列程序和动作按照特定次序组成,程序和动作都不可逆。如果在执行过程中系统发生差错,就有可能导致着陆失败,因此,需要确保降落系统的可靠性和安全性。本文选取返回舱降落伞系统作为应用验证的对象,根据层次化系统建模方法对返回舱降落伞系统进行建模,建立系统的需求、结构以及行为模型,并将其中的活动图和模块定义图转换成 Modelica 模型进行仿真,通过对仿真结果的分析,验证是否满足系统的正确性和一致性要求。

3.1 基于 SysML 的返回舱降落伞系统建模

SysML 建模工具采用 Enterprise Architect,在对返回舱降落伞系统建模设计过程中,采用层次化建模思想,对整个系统自顶向下逐步精化,分为 4 个层次(系统层、功能层、软件/硬件分割层和部署层)分别从需求、结构以及行为 3 个方面完成系统建模,输出各类模型图共计 53 个,部分模型如图 6 所示。

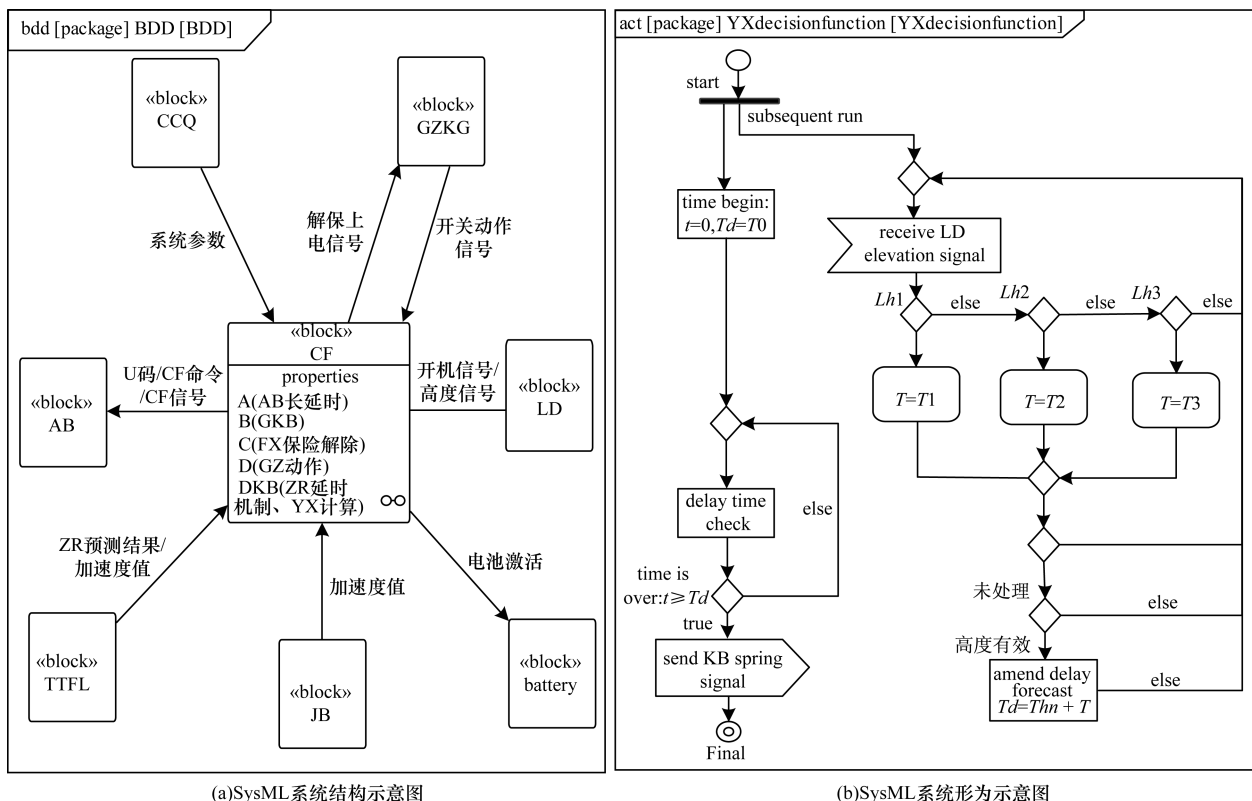


图 6 返回舱降落伞系统建模模型

3.2 系统性质

选取功能层中描述各子系统行为场景的活动图以及描述各子系统值属性的模块定义图,导出 XML 文件并通过模型转换工具生成相对应的 Modelica 模型,将获取的 Modelica 模型导入工具 OpenModelica 中,并在生成的数据中选取多组不同的仿真数据进行仿真,但在仿真之前需定义返回舱降落伞系统待验证的系统性质,系统性质包括系统功能正确性和业务逻辑一致性,系统功能正确性关注功能是否正确实现,可根据系统实际需求进行转化,针对返回舱降落伞系统,系统功能正确性是指系统一定或者满足特定条件下会触发某个动作信号,业务逻辑一致性是指共享变量同一时刻不能出现多种取值,以下性质 1、性质 2 和性质 3 是系统功能正确性要求,性质 4 是系统业务逻辑一致性要求。

性质 1 系统发出 EmeSignal 信号或者 OpenCmd 命令。

性质 2 SepSignal 信号发出后,系统一定发出 OpenCmd 命令。

性质 3 OvSignal 信号发出后,系统一定发出 OpenCmd 命令。

性质 4 共享变量同一时刻不能出现多种取值。

3.3 仿真验证结果

采用基于路径的仿真数据自动生成方法,通过遍历活动图路径得到所有的仿真数据,仿真数据的生成考虑了活动图之间的路径组合,每个活动图的路径依照第 2.2 节中的关系进行组合,整个系统活动图组合后的路径共有 480 ($1 \times 5 \times 6 \times 16 \times 1$) 条,基于路径自动生成 480 组仿真数据。

在 OpenModelica 仿真结果中对信号进行有效性判断:所有信号的发出都必须在活动图结束之前,否则该信号无效;在多个互斥信号中,最先发出的信号有效。通过对仿真结果进行分析,发现存在仿真数组不满足系统性质 1~性质 3 的情况,具体仿真数据如下: $\{T_m = 8; T_d = 10; Time_1 = 40; Work Mode = 0; Mode = 1; AccType = 1; a_1 = 5; a_2 = [9, 11, 12, 8, 11, 12, 13, 7, 11, 5, 6, 7, 7, 11, 23, 8, 26]; OlValue = 14; T = 10; time_2 = 60; Altitude = 1\ 000; notHandled = false; available = false\}$,性质 1~性质 3 均未满足。

针对性质 1:在图 7 的仿真结果中,下方的 3 条仿真曲线都未发生变化,表示直到仿真结束时系统都未发出 OpenCmd 命令。EmeSignal 信号在活动结束之后发出,即为无效命令。因此,不满足性质 1。

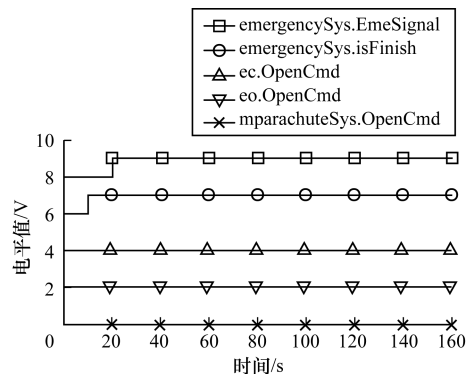


图 7 性质 1 反例结果

针对性质 2:在图 8 的仿真结果中,伞舱盖弹射分离活动中的 SepSignal 信号在仿真时间为 20 s 时发出并且为有效信号,但是直到仿真结束时系统都未发出 OpenCmd 命令。因此,不满足性质 2。

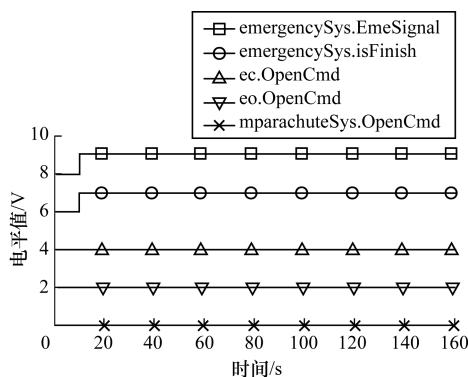


图 8 性质 2 反例结果

针对性质 3:在图 9 的仿真结果中,主伞状态监测活动中 OvSignal 信号在仿真时间为 40 s 时发出并且为有效信号,但是直到仿真结束时系统都未发出 OpenCmd 命令。因此,不满足性质 3。

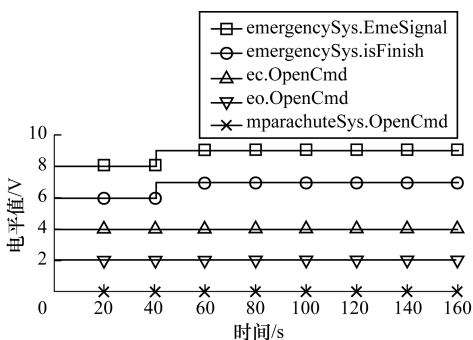


图 9 性质 3 反例结果

针对性质 4:为验证系统业务逻辑的一致性,即共享变量取值相同,需要对所有的仿真结果进行分析。返回舱降落伞系统中的共享变量为伞舱盖弹射分离和主伞包开启活动图中的 SepFlag 变量,分析所有的仿真结果可以发现该变量取值始终相同,如

图 10 所示。因此,系统满足性质 4。

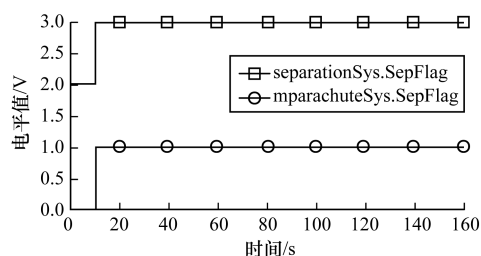


图 10 性质 4 仿真结果

4 结束语

模型驱动的系统设计方法广泛应用于航空航天、轨道交通、工控设备等领域的嵌入式系统开发。然而,随着系统综合程度的不断提高,功能逻辑越来越复杂,需要在系统研制的前期,即需求论证阶段,对系统需求和设计方案进行正确性和充分性验证,进而发现系统性设计缺陷,避免引发严重的系统性问题。为此,本文基于 SysML 提出一种系统建模和基于 Modelica 的仿真验证方法。该方法基于系统结构与行为模型,依据仿真验证规则进行优化并自动转换为 Modelica 模型进行仿真验证,结果表明,该方法对系统需求论证具有可行性。嵌入式软件系统未来的一个发展趋势是在系统层面上,需要对系统的边界及安全需求展开多层次多角度的建模分析与验证,而安全需求建模方法体系、安全需求重用模式研究、多层次的验证方法以及支持相应的建模、验证、分析方法的工具链是其中重要的内容,这也是下一步的研究方向。

参考文献

- [1] 王福刚,杨文君,葛良全. 嵌入式系统的发展与展望[J]. 计算机测量与控制,2014,22(12):3843-3847.
- [2] FEILER P H, GLUCH D P, HUDAK J J. The architecture analysis and design language: an introduction [R]. Pittsburgh, USA: Pittsburgh PA Software Engineering School, Carnegie-Mellon University, 2006.
- [3] FEILER P H, GLUCH D P. Model-based engineering with AADL: an introduction to the SAE architecture analysis and design language [M]. [S. l.]: Addison-Wesley, 2012.
- [4] DEBRUYNE V, SIMONOT-LION F, TRINQUET Y. EASTADL—an architecture description language[M]. Berlin, Germany: Springer, 2005:181-195.
- [5] 姜梦霞,江国华. 一种安全性关键软件的评估模型[J]. 计算机工程,2015,41(10):130-138,143.
- [6] TIWARI A. Formal semantics and analysis methods for Simulink Stateflow models [D]. Palo Alto, USA: Stanford International Institute, 2002.
- [7] KANG E Y, KE L, HUA M Z, et al. Verifying automotive systems in Eastadl/Stateflow using UPPAAL [C]//Proceedings of APSEC '15. Washington D. C., USA: IEEE Press, 2015:143-150.
- [8] 吴黎明,胡军,曹东,等. 无人机飞控软件系统建模与测试用例生成研究[J]. 计算机科学,2012,22(12):112-118.
- [9] FRIEDENTHAL S, MOORE A, STEINER R. A practical guide to SysML [M]. [S. l.]: Morgan Kaufmann, 2014.
- [10] DELLIGATTI L. SysML 精粹[M]. 侯伯微,朱艳兰,译. 北京:机械工业出版社,2014.
- [11] MOSTERMAN P J, PRABHU S, DOWD A, et al. Embedded real-time control via Matlab, Simulink, and xPC target [J]. Control Engineering, 2005(11):419-446.
- [12] PTOLEMAEUS C. System design, modeling, and simulation; using Ptolemy II [M]. [S. l.]: Lulu Press, Inc., 2014.
- [13] SPENDLA L, HRCKA L. Proposal of system testing integration into safety critical system design process supported by SysML [C]//Proceedings of the 8th European Modelling Symposium. Berlin, Germany: Springer, 2014:251-256.
- [14] 魏小凤,胡继承,罗永恩. 基于超图模型的软件模块自动划分[J]. 计算机工程,2016,42(1):71-76.
- [15] LASALLE J, PEUREUX F, FONDEMENT F. Development of an automated MBT toolchain from UML/SysML models [J]. Innovations in Systems and Software Engineering, 2011,7:247-256.
- [16] GAUTHIER J M. Test generation for RTEs from SysML models: context, motivations and research proposal [C]//Proceedings of the 6th IEEE International Conference on Software Testing, Verification and Validation. Washington D. C., USA: IEEE Press, 2013:503-504.
- [17] LASALLE J, PEUREUX F, GUILLET J. Automatic test concretization to supply end-to-end MBT for automotive mechatronic systems [C]//Proceedings of the 1st International Workshop on End-to-End Test Script Engineering. Toronto, Canada: [s. n.], 2011:16-23.
- [18] 张青斌. 飞船返回舱降落伞系统动力学[M]. 北京:国防工业出版社,2013.

编辑 索书志