软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(2):321−331 [doi: 10.13328/j.cnki.jos.004784]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

面向航天嵌入式软件的形式化建模方法*

顾 斌[1], 董云卫[1], 王 政[2]

[1](西北工业大学 计算机学院,陕西 西安 710029)

[2](北京控制工程研究所,北京 100190)

通讯作者: 王政, E-mail: irwintiger@hotmail.com, http://www.bice.org.cn/

摘 要: 航天嵌入式软件是航天型号任务成败的关键之一.航天嵌入式软件是一种周期性、多模式的软件.软件的每个模式表示系统处于一定的状态,并进行相应的复杂计算.因此,提出了一种名为 SPARDL 的形式化建模方法.为了满足型号应用的需求,对这一方法进行了若干改进.为了表达航天器的时序性质,提出了一种基于区间逻辑的性质规范语言.为了支持工业应用,还设计了代码生成方法.这一建模方法已在航天工业领域得到了应用.

关键词: 航天嵌入式软件;形式化建模方法

中图法分类号: TP311

中文引用格式: 顾斌,董云卫,王政.面向航天嵌入式软件的形式化建模方法.软件学报,2015,26(2):321−331. http://www.jos.org.cn/1000-9825/4784.htm

英文引用格式: Gu B, Dong YW, Wang Z. Formal modeling approach for aerospace embedded software. Ruan Jian Xue Bao/ Journal of Software, 2015,26(2):321−331 (in Chinese). http://www.jos.org.cn/1000-9825/4784.htm

Formal Modeling Approach for Aerospace Embedded Software

GU Bin[1], DONG Yun-Wei[1], WANG Zheng[2]

[1](School of Computer Science, Northwestern Polytechnical University, Xi'an 710029, China)

[2](Beijing Institute of Control Engineering, Beijing 100190, China)

Abstract: The success of aerospace depends on the correctness of aerospace embedded software. Such embedded software are usually period-driven and can be decomposed into different modes with each mode representing a system state observed from outside. Such software may also involve intensive computing in their modes. Currently, there is lack of domain-specific formal modeling languages for such software in the relevant industry. To address this problem, this article proposes a formal visual modeling approach called SPARDL as a concise and precise way to specify such software. To capture the temporal properties of aerospace embedded software, a property specification language is provided based on interval logic. To support application in industry, code generation methodology is also discussed. This modeling approach is applied in some real-life cases in aerospace industry.

Key words: aerospace embedded software; formal modeling method

航天嵌入式控制软件实现了航天器的控制算法,是航天型号任务成败的关键之一.作为一款周期性控制系统,航天嵌入式控制软件具有周期性、开放性、计算复杂、逻辑状态组合繁多、模式切换条件复杂等特征.软件研制成本较高,保证软件正确性的代价高昂.

本研究的目的是为航天软件需求工程师提供一套需求捕获的方法学,有利于需求工程师清晰、无二义性地表述航天嵌入式控制软件的需求,并能在需求分析阶段进行分析和验证,以便尽早发现可能存在的问题,有效地提高软件质量,控制软件开发成本.

基于以上目的,我们提出了一套框架:航天需求描述语言(spacecraft requirement description language,简称

* 基金项目: 国家自然科学基金(90818024, 91118007); 上海市高可信计算重点实验室开放课题(07dz22304201304)
收稿时间: 2014-07-03; 修改时间: 2014-10-31; 定稿时间: 2014-11-26

SPARDL)[1].为了更好地适应航天型号任务的需求,我们在 SPARDL 的基础上,以模式图[2]为核心,为航天需求工程师提供了一整套建模、分析的手段,并在该模型上进行仿真和分析.本研究致力于以 SPARDL 为原型,再结合一个简单、易操作的界面,开发出一个航天需求描述工具,能够交付给需求设计人员使用,帮助他们解决前面提到的那些问题.SPARDL 的整体架构如图 1 所示.

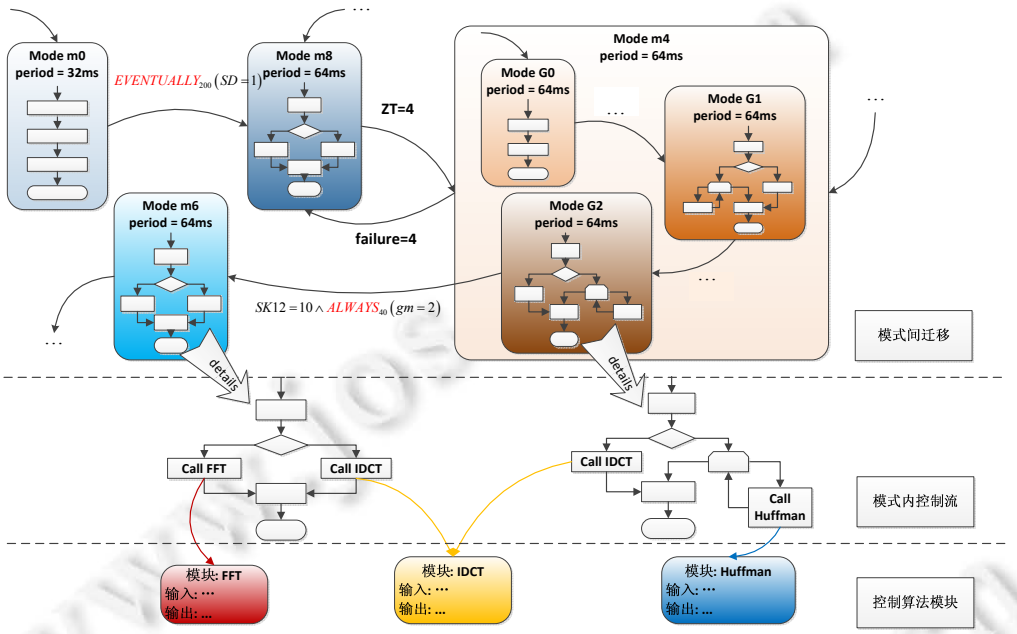


Fig.1 Architecture of SPARDL

图 1 SPARDL 的整体架构

1 航天需求描述语言

为了清晰且无二义地描述航天型号软件的需求,我们定义了航天需求描述语言 SPARDL[1].为了便于和用户(需求设计人员)交互,我们还设计了它的图形表示[2].

SPARDL 的核心是模式图,模式图中的节点对应嵌入式控制软件的模式.节点允许嵌套,以便支持子模式等特征.节点内部含有控制流,用于描述模式中的计算任务.模式的计算任务可以调用封装的控制算法,这些控制算法也由控制流加以描述.计算任务、控制算法共用全局变量,这些全局变量由数据字典加以描述.图 2 给出了一个模式图的示例.

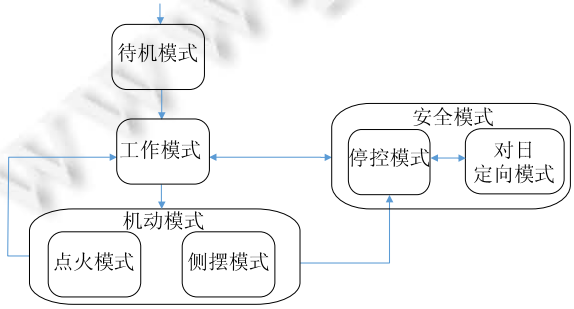


Fig.2 An example of mode diagram

图 2 模式图示例

模式图具有以下特征:

(1) 基于模式:SPARDL 的基本建模元素是“模式(mode)”,每个模式表明系统处于特定的状态,使用特定的控制算法.

(2) 时序谓词:控制系统的行为不仅取决于系统当前的状态,而且也取决于系统的历史状态.为了刻画这一特征,SPARDL 提供了两种时序谓词:$ALWAYS_L(condition)$和 $EVENTUALLY_L(condition)$.以图 3 为例,为了判定时间谓词 $ALWAYS_2(x>10)$在状态 $s6$ 上是否成立,我们检查布尔表达式 $x>10$ 在状态 $s4$,$s5$ 和 $s6$ 上成立是否.$x>10$ 在状态 $s4$,$s5$ 和 $s6$ 上都成立,因此,$ALWAYS_2(x>10)$在状态 $s6$ 上成立.为了判定时间谓词 $EVENTUALLY_3(x>5)$在状态 $s5$ 上是否成立,我们检查布尔表达式 $x>5$ 在状态 $s2$ 上成立是否.因为 $x>5$ 在状态 $s2$ 上成立,因此,$EVENTUALLY_3(x>5)$在图中的状态 $s5$ 上成立.

(3) 时序控制流:在实际物理世界中,被控对象具有一定的响应时间,因此,控制系统的输出应当符合被控对象响应时间的约束.SPRADL 提供了两种基本的时序控制流:$ALWAYS_L(v:=e)$和 $EVENTUALLY_L(v:=e)$.前者表示在接下来的 L 个控制周期中均会执行赋值操作 $v:=e$;后者表示从当前周期开始,L 个控制周期之后会执行一次 $v:=e$.

(4) 周期驱动:控制系统是一种不会终止的反应式系统(reactive system),这种系统按照给定的时间反复执行计算任务.SPARDL 模型中的模式阐明了控制软件的周期.并不要求各个模式的周期相同,因此,控制软件处于不同的模式时,可以有不同的周期.例如,对于遥感卫星的控制软件来说,在处于轨道控制模式时,需要非常精准地调整星体姿态,因此周期较短;而在处于巡航模式时,由于星体姿态已经稳定,只需稍加调整,维持星体姿态即可.因此,与巡航模式相比,轨道控制模式的周期更短,以便实现更加精细的控制.

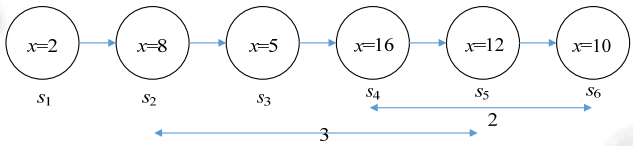


Fig.3 An example of timed predicate

图 3 时间谓词示例

与本项研究类似的工作包括状态图(StateChart)[3]、状态流(Stateflow)[4]、航空架构描述语言(AADL)[5]和 Giotto[6]等.

状态图(StateChart)被广泛应用于软件的建模与分析.模式图中的模式类似于状态图中的状态,但是有以下区别:

- 在状态图中,当迁移条件成立时,系统立即发生迁移;但是因为模式图用于描述周期驱动的控制系统,所以模式之间的迁移行为是被周期驱动的,仅仅在周期结束时,才允许发生迁移.
- 在状态图中,迁移条件是关于状态图当前状态的布尔表达式;而在模式图中,迁移条件可以是关于历史状态序列的逻辑表达式(时间谓词).
- 当状态图处于某个状态时,可以认为状态图不发生动作;对于模式图来说,在处于某个模式时,需要执行该模式对应的计算任务.

状态流(Stateflow)是商业软件 Matlab/Simulink 中提供的一种建模语言,类似于状态图.状态流在状态图的基础上融合了基于控制流的计算过程,可以与 Simulink 中的物理部件组合使用,用于建模各种控制系统;而模式图仅关注于周期驱动的控制系统,为了更方便、准确地描述这种系统,模式图将“周期”列为第 1 类元素.状态流允许在迁移上附带一个控制流图,用于描述迁移发生时的动作,而模式图仅允许在模式内描述计算过程.在提供建模语言模式图的同时,SPARDL 还提供了基于区间时序逻辑的性质描述语言,而状态流只能使用 Simulink 提供的各种检查组件描述系统应该满足的性质.本文为 SPARDL 系统定义了完整的语义,而状态流本身并无形式

化语义.

AADL 是一套从航天/航空领域发展而来的架构分析与设计语言,它的功能强大,覆盖面广.它自身并没有形式化验证的能力,但是作为一个开放式的构架,可以将现有的方法集成到 AADL 框架.主要有两种思路:一是将 AADL 模型转化为现成的形式化模型,如时间自动机、Petri 网等;二是为 AADL 设计专门的形式化模型,然后以这个模型为基础,设计形式化验证方法.

Giotto 是 Henzinger 等人设计的一种周期驱动的建模语言.SPARDL 与 Giotto 的主要区别在于各自的计算机制不同.在 Giotto 中,模式内允许多个并发的计算任务,计算任务的过程和细节被省略.而在模式图中,计算任务是串行的,其过程由控制流图描述.这样,模式图可以在设计阶段就描述具体的控制算法,而 Giotto 则将控制算法的描述推迟到实现阶段.在模式的结构方面,Giotto 不支持模式嵌套,也不支持与历史时序有关的迁移条件.在仿真方面,Giotto 专注于并发计算任务调度的可实现性.本文中,SPARDL 仅描述顺序模型,在下一步的工作中,将考虑引入多任务、中断等机制,以适应日益复杂的航天嵌入式软件.

SPARDL 用一系列模式来体现嵌入式周期控制系统中的行为特征,每个模式可以周期性地执行一系列的过程.系统可以周期性地运行在某个模式中,直到满足某个迁移条件,系统可能会进入另一个模式.模式 Mode 的定义见表 1.

Table 1 Syntax of mode

表 1 模式的语法

Mode	::=	$\langle name,period,Init,(Proc\|Mode^{+}),Trans\rangle$
Init	::=	*stmt**
Proc	::=	*stmt**
Trans	::=	$\langle priority,condition,Action,target\rangle$
Action	::=	*stmt**

在表 1 中,*name* 代表模式的名称,*period* 代表模式的周期,*Init* 代表模式初始化,*Proc* 表示模式处理过程,*Trans* 表示模式迁移.模式也可以由若干子模式组成.模式初始化和模式处理过程均为若干控制流语句.模式迁移包括优先级、条件、迁移操作和目标模式.

条件和控制流语句的定义见表 2.

Table 2 Syntax of condition and control flow

表 2 条件和控制流的语法

stmt	::=	IF *condition* THEN *stmt** (ELSE *stmt**)? ENDIF| WHILE *condition stmts**| $ALWAYS_L(v:=e)\|EVENTUALLY_L(v:=e)\|v:=e$
condition	::=	true|false|$p(e_1,\ldots,e_n)$| *condition* AND *condition* |*condition* OR *condition* |NOT *condition*| $ALWAYS_L(condition)\|EVENTUALLY_L(condition)$
expression	::=	$c\|v\|f(e_1,\ldots,e_n)$

下面给出一个 SPARDL 的条件和控制流描述控制系统行为的示例.

控制系统的行为需求如下:如果连续 1 600 个控制周期角速度超限($|\omega|>1.0$),那么先关闭发动机($REAG=0$),32 个控制周期后再关闭自锁阀($LV=0$),自锁阀关闭两次,间隔 8 个控制周期.

该行为需求的 SPARDL 控制流如下:

IF $ALWAYS_{1600}(|\omega|>1.0)$ THEN

  $REAG$:=0

  $EVENTUALLY_{32}(LV:=0;EVENTUALLY_8(LV:=0))$;

ENDIF

也可以表示为

IF $ALWAYS_{1600}(|\omega|>1.0)$ THEN

$REAG$:=0

$EVENTUALLY_{32}(LV$:=0);

$EVENTUALLY_{40}(LV$:=0);

ENDIF

设从状态 S_i 开始,角速度超限($|\omega|>1.0$),则该行为需求对应的状态序列如图 4 所示.

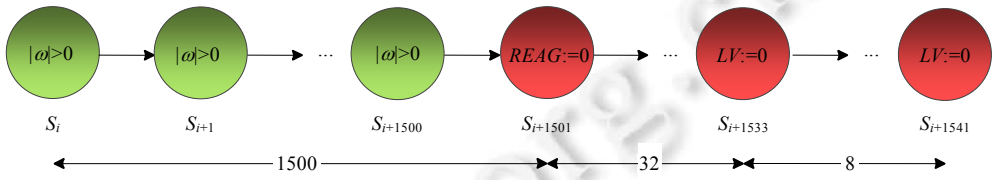


Fig.4 An example of behaviour requirements

图 4 行为需求示例

2 需求性质描述语言

为了分析需求模型的正确性,我们以区间时序逻辑(interval temporal logic,简称 ITL)[7,8]为基础,给出了相应的性质描述语言.ITL 以时段为模型解释时序逻辑公式,非常适合航天嵌入式控制软件的应用场景.ITL 的语法见表 3.

Table 3 Syntax of interval temporal logic

表 3 区间时序逻辑的语法

项	Term	::=	$c\|v\|f(T_1,\ldots,T_n)\|L$
公式	Formula	::=	TRUE|FALSE|$p(T_1,\ldots,T_n)$| NOT Formula| F_1 AND F_2| F_1 OR F_2| F_1~F_2
F 算子	F (Formula)	::=	TRUE~(Formula~TRUE)
G 算子	G (Formula)	::=	NOT (F(NOT Formula))

ITL 公式的项是常量(c)、变量(v)或函数符号(f),其中,L 是一个特殊的变量,表示区间的长度.ITL 的公式是项或公式的逻辑组合.其中,F_1~F_2 表示一个可以分为两个部分的区间,F_1 在前一部分的区间上成立,F_2 在后一部分的区间上成立,~被称为 Chop 算子.时序逻辑中的 G 算子和 F 算子由 Chop 算子定义.

图 5 描述了这样的场景:“当系统处于模式 m_4 时,如果发生错误,那么必须在 100ms 内切换到模式 m_8”.经典的线性时态逻辑(LTL)无法完全刻画这一性质,LTL 公式 $G(m_4 \wedge failure \rightarrow F(m_8))$仅仅表示了“切换到模式 m_8”这一时序信息,而遗漏了“在 100ms 内”内这一区段信息.用 ITL 公式可以精确地描述这一场景:

$$G\begin{pmatrix}(m_4 \wedge (\neg failure \sim failure) \sim \text{TRUE}) \rightarrow \\ (m_4 \wedge (\neg failure \sim (failure \wedge l \leqslant 100)) \sim m_8 \sim \text{TRUE})\end{pmatrix}.$$

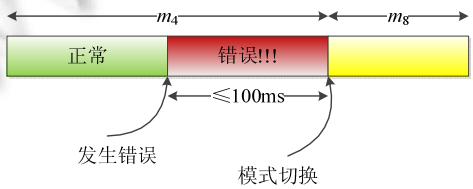


Fig.5 An example of ITL formular

图 5 ITL 公式示例

区间时序逻辑的语义定义在状态序列上.设状态序列 $S=s_0,s_1,\ldots,s_n,\ldots$,其中的每个状态均是变量到其取值的映射.在区间时序逻辑中,项和公式的语义解释见表 4.

Table 4 Semantics of ITL

表 4 区间时序逻辑的语义解释

项的语义解释			
常量	$I(S,c)=c$		
变量	$I(S,v)=s_0(v)$		
函数	$I(S,f(T_1,\ldots,T_n))=f(I(S,T_1),\ldots,I(S,T_n))$		
区间长度	$I(S,L)=$	$j-i$	如果 $S=s_i,\ldots,s_j$
		无穷	如果 S 是无穷状态序列
公式的语义解释			
常量	$I(S,\text{TRUE})=\text{TRUE}$		
	$I(S,\text{FALSE})=\text{FALSE}$		
谓词	$I(S,p(T_1,\ldots,T_n))=p(I(S,T_1),\ldots,p(S,T_n))$		
否	$I(S,\text{NOT } F)=\text{NOT } I(S,F)$		
合取	$I(S,F_1 \text{ AND } F_2)=I(S,F_1) \text{ AND } I(S,F_2)$		
析取	$I(S,F_1 \text{ OR } F_2)=I(S,F_1) \text{ OR } I(S,F_2)$		
Chop	$I(S,F_1{\sim}F_2)=\text{exists } k.\ I(s_0,\ldots s_k,F_1) \text{ AND } I(s_k,\ldots,F_2)$		

虽然区间时序逻辑适于描述周期控制系统的行为性质,但是对于领域工程师来说,直接使用区间时序逻辑公式描述性质过于冗长、繁琐.因此,本节在区间时序逻辑的基础上,考虑周期控制系统需求的特殊性、领域工程师的使用习惯等因素,设计了若干性质模板,如表 5 和图 6 所示.

Table 5 Templates of ITL properties

表 5 ITL 性质模板

编号	ITL 公式	说明
1	$G((\neg p{\sim}p{\sim}\neg p){\rightarrow}(\neg p{\sim}(p\wedge l\geqslant t){\sim}\neg p))$	如果性质 p 成立,那么至少连续成立 t 个时间单位
2	$G((\neg p{\sim}p{\sim}\neg p){\rightarrow}(\neg p{\sim}(p\wedge l\leqslant t){\sim}\neg p))$	如果性质 p 成立,那么最多连续成立 t 个时间单位
3	$G\begin{pmatrix}(\neg p\sim(p\wedge\neg q)\sim\text{TRUE})\rightarrow\\(\neg p\sim(p\wedge\neg q\wedge l\leqslant t)\sim q)\end{pmatrix}$	如果性质 p 从不成立变为成立,那么最多经过 t 个时间单位,性质 q 也会成立
4	$G\begin{pmatrix}(\neg p\sim(p\wedge l\geqslant t_1)\sim\text{TRUE})\rightarrow\\(\neg p\sim p\wedge((p\wedge\neg q\wedge l\leqslant t_2)\sim q))\end{pmatrix}$	如果性质 p 持续成立至少 t_1 个时间单位,那么最多经过 t_2 个时间单位,性质 q 也会成立

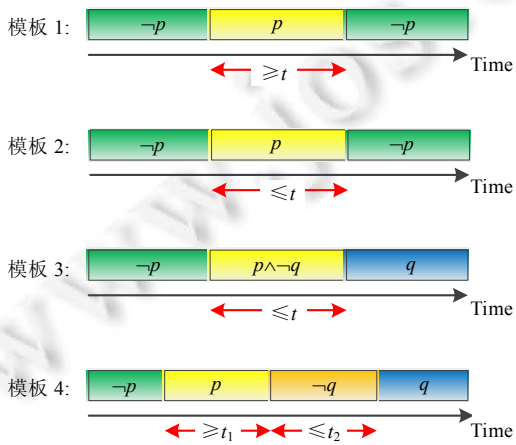


Fig.6 Templates of properties for software model

图 6 软件模型的性质模板

除了更加方便工程师撰写性质之外,使用性质模板的另一个好处是提高了验证的效率.如果直接按照区间时序逻辑的语言实现验证算法,其效率是低下的,时间复杂度级别是 $O(n^l)$,其中,n 是状态序列中的状态数,l 是待验证的公式中的 Chop 算子的个数.进行验证时,根据模板本身的特点,可以减少不必要的计算.

3 代码生成、仿真与概率模型检查

根据 SPARDL 形式化模型可以生成原型代码,然后编译产生可执行程序.这样在需求分析阶段,软件工程师就可以方便地观察软件的动态行为[9].软件动态行为也还可以作为概率模型检查的输入[10].图 7 给出了从模型生成代码并进行快速仿真的流程.

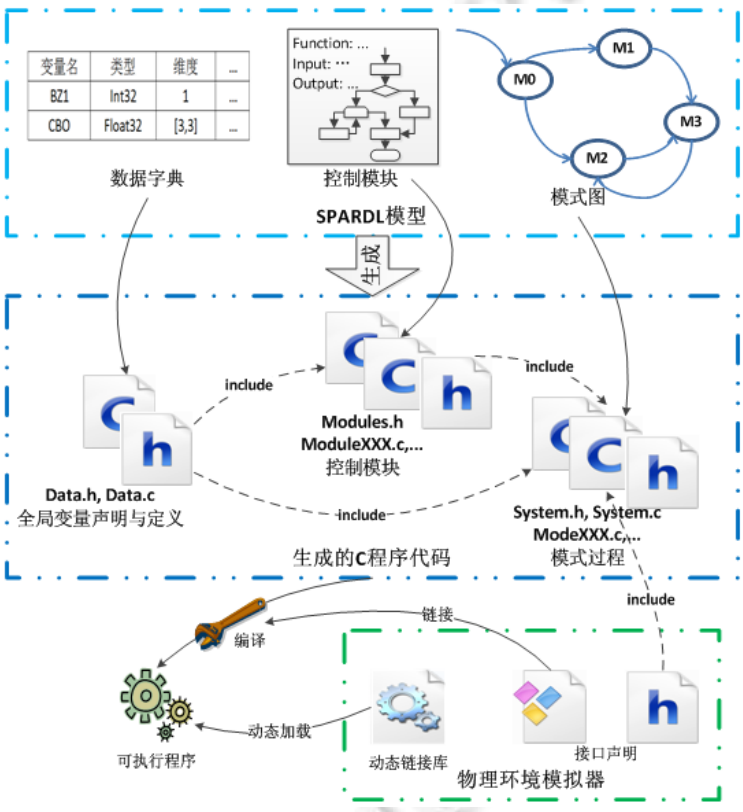


Fig.7 Process of code generation

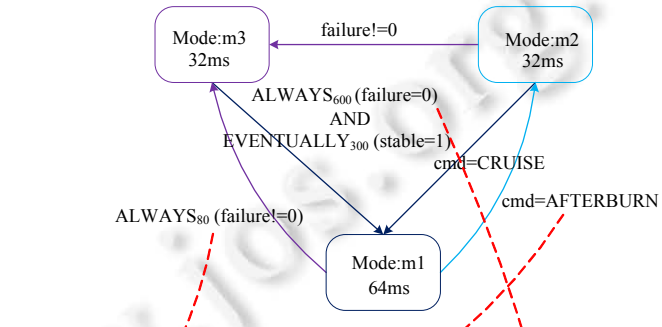
图 7 代码生成的过程

作为嵌入式控制软件的模型,SPARDL 系统的行为不仅由其自身决定,而且依赖于环境,即,传感器的输入和执行机构的动作.SPARDL 建模语言本身并没有提供描述传感器和执行机构的建模元素,仅通过数据字典标注了来自传感器输入的只读变量和发送到执行机构的只写变量.传感器和执行机构的行为通过物理环境模拟器加以模拟.自动生成的 C 程序代码依赖于物理环境模拟器.在实践中,物理环境模拟器以动态链接库的形式由控制工程师提供.在生成 C 程序代码进行模拟的方法中,C 程序代码将调用动态链接库中提供的物理环境模拟器的接口.在解释执行的方法中,解释器在运行时动态加载物理环境模拟器,并调用其中的接口.

SPARDL 模型的一个特色是时间谓词,即,该谓词的真假不仅依赖于系统的当前状态,而且依赖于系统的历史状态.这一特色不被 C 语言直接支持.为了判断一个时间谓词是否成立,没有必要完整地记录历史状态,只需关注这个时间谓词中的布尔表达式在各个历史状态中是否成立.因此,可以为每个时间谓词设置一个计时器,用于

记录其布尔表达式在历史状态序列中成立的时段的长度.

生成 C 程序代码时,时间谓词被转化为函数调用.预先定义两个函数 *always* 和 *eventually*,分别处理 $ALWAYS_L(condition)$和 $EVENTUALLY_L(condition)$.图 8 展示了这两个函数:函数的第 1 个参数 *guard* 是一个整型量,用于表示时间谓词中的布尔表达式 *b* 在当前状态的真假;第 2 个参数 *bound* 的类型是浮点型,对应于时间谓词中的 *L*;第 3 个参数也是浮点型,表示当前最内层模式的周期长度;最后一个参数是一个整型量,表示时间谓词的编号,用于区分不同的时间谓词.每个出现在 SPARDL 模型中的时间谓词都被转化为对函数 *after* 和 *duration* 的调用.数组 *cnt_a* 和 *cnt_d* 是时间谓词的计时器,这两个数组的长度分别是 SPARDL 模型中出现的 *ALWAYS* 谓词和 *EVENTUALLY* 谓词的数目.




```
void mode_1_ transition ()
{
    if (always (failure!=0,0.08,0.064,0))
    {
        mode_flag=3;
    }
    else if (cmd==AFTERBURN )
    {
        mode_flag=2;
    }
}

float clock_a[1]={0,0f};

int eventually (int b,
                float bound,
                float periodLen,
                int idx)
{
    if (clock_a[idx]>0.0f)
    {
        clock_a[idx]+=periodLen;
        if (clock_a[idx]>=bound)
        {
            clock_a[idx]=0.0f;
            return 1;
        }
    }
    else
    {
        if (b!=0)
            clock_a[idx]+=periodLen;
    }
    return 0;
}
```

```
void mode_3_ transition ()
{
    if (always (failure!==0,0.06,0.032,1)
        && eventually (stable==1,0.3,0.32,0))
    {
        mode_flag=1;
    }
}

float clock_d[2]={0,0f};

int always (int b,
            float bound,
            float periodLen,
            int idx)
{
    if (b!=0)
    {
        clock_d[idx]+=periodLen;
        if (clock_d[idx]>=bound)
        {
            clock_d[idx]=0.0f;
            return 1;
        }
    }
    else
    {
        clock_a[idx]=0.0f;
    }
    return 0;
}
```

Fig.8 Code generation for temporal predicate

图 8 时序谓词的代码生成 bound

许多建模工具都支持从模型生成代码,例如 MATLAB,RT-CASE,Giotto 等.与其他工具相比,SPARDL 提供了针对航天嵌入式领域特点的建模元素,因此,生成的代码具有较高的可读性和可用性.

航天嵌入式控制软件是一个复杂的异构系统,区间时态逻辑也比一般的时态逻辑更为复杂.经典的模型检查算法难以处理这种复杂的问题,因此,应用概率模型检查方法验证模型是否满足性质.给定系统的初始状态、系统的运行环境,模式图的行为就是确定的.但是在检查 SPARDL 模型是否满足区间时序逻辑性质时,有两个原因将随机因素引入模式图:第一,需求文档并不会一一列举每一个初始状态,而是给定初始状态的取值范围,即初始状态在给定的范围内随机地取值;第二,作为一种反应式系统,SPARDL 模型的行为受外界环境的影响.外界环境被视为黑盒,即,每次采样,随机地更新输入变量的值.因此,SPARDL 模型开始运行时,每个变量的取值是一个符合一定概率分布的随机变量.模式图在每个周期进行数据采集时,输入变量的值被更新为新的随机变量.

概率模型检查把性质 p 在模型的一次执行(一条路径)上是否成立视为一个服从两项分布的随机变量.概率模型检查主要考虑以下两个问题:给定模型 M 和性质 p,

1) 估算在模型 M 的任意一条路径 t 上,p 成立的可能性,即参数估计;

2) 判断上述可能性是否大于给定的阈值,即假设检验.

这两个问题的算法见表 6.

Table 6 Stochastic model checking

表 6 概率模型检验

过程:定量参数估计.
输入:*m*:SPARDL 模型生成的程序,*p*:性质,*delta*:置信区间半长,*e*:置信度.
输出:*prob*:性质 *p* 成立的概率.

```
begin
    N=(4×log(1/delta))/(e×e);
    a=0
    for i:=1 to N do
        生成初始状态 s0
        执行 m,获得状态序列 tr
        if p 在 tr 上成立 then a=a+1
    end loop
    return a/N
end
```

过程:定性参数检验.
输入:*m*:SPARDL 模型生成的程序,*p*:性质,*prob*:性质 *p* 成立的概率,*delta*:置信区间半长,*e*:置信度.
输出:是/否:性质 *p* 在 *m* 上成立的概率是否大于 *prob*.

```
begin
    r=0, p0=prob+delta, p1=prob−delta,
    while true do
        生成初始状态 s0
        执行 m,获得状态序列 tr
        if p 在 tr 上成立 then r=r+log(p1/p0) else r=r+log((1−p1)/(1−p0));
        if r≤log(beta/(1−alpha)) then return Yes
        if r≥log((1−beta)/alpha) then return No
    end loop
end
```

在表 6 中,*alpha* 和 *beta* 分别是是假设检验的强度参数,可以分别设为略大于 1 和略小于 1 的值.

4 应用案例

我们在某深空探测型号的软件研制中进行了应用.该系统的主要任务是在地外天体进行钻探,获取土壤样本.图 9(a)是该软件需求分析的顶层模式图.软件分为遥控模式和自主模式.其中,自主模式又分为钻进、分离、提取这 3 个子模式.图 9(b)是其中钻进子模式进行功能分解之后的模式图.系统上电启动时,处于遥控模式.地面测控人员发送遥控指令,可以人工控制系统工作.测控人员也可以发送遥控指令,令其转入自主模式,由软件自主控制系统进行钻探获取土壤样本的工作.

由于此前仅有美国进行过类似的工作,相关资料比较匮乏,系统设计方案变化频繁,导致软件需求变更频繁.传统的基于文档的软件需求分析方式,难以适应任务需求.因此,在软件研制过程中,使用 SPARDL 模型描述

软件需求,并生成代码.通过仿真、调试生成的代码,实现了系统设计方案合理性、正确性的快速验证,提高了软件研制的效率.

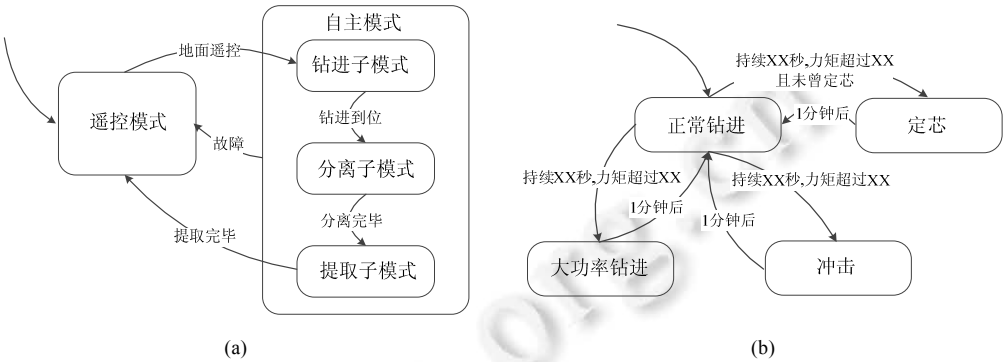


Fig.9 Mode diagram of an application

图 9 应用案例的模式图

5 结束语

本文针对航天嵌入式软件提出了一种需求分析方法:首先,用自然语言关键词规范初始需求文档;然后,对已规范化的需求文档进行解析,并构造出由 SPARDL 建模语言定义的软件模型.SPARDL 建模语言具有形式化的语法和语义,为嵌入式周期控制软件提供了一套形式、直观而准确的建模机制.

本文通过从 SPARDL 模型生成对应的 C 程序代码,开发软件原型.该原型可以模拟系统需求行为,以方便工程师来验证系统需求,这实际上是一种基于模型的测试方法.

基于本文的方法,已有相应的工具开发并试用于航天控制软件.该软件控制的航天器已经成功发射.在型号研制过程中,SPARDL 方法帮助软件工程师发现了若干软件需求问题,避免了这些问题遗留到软件测试,甚至航天器系统测试阶段.第三方测试的结果和航天器在轨表现均说明,SPARDL 方法有效地提高了航天嵌入式软件的质量.

References:

[1] Wang Z, Li JW, Zhao YX, Qi YX, Pu GG, He JF, Gu B. Spardl: A requirement modeling language for periodic control system. In: Proc. of the ISoLA (1). LNCS 6415, Heidelberg: Springer-Verlag, 2010. 594−608. [doi: 10.1007/978-3-642-16558-0_48]

[2] Wang Z, Pu GG, Li JW, He JF, Qin SC, Larsen KG, Madsen J, Gu B. MDM: A mode diagram modeling framework. In: Proc. of the FTSCS 2012. 2012. 135−149. [doi: 10.4204/EPTCS.105.10]

[3] Giese H, Burmester S. Real-Time statechart semantics. Technical Report, TR-RI-03-239, Paderborn: Software Engineering Group, University of Paderborn, 2003.

[4] The mathworks: Stateflow and stateflow coder, user's guide. 2014. http://www.mathworks.com/help/releases/R13sp2/pdf_doc/stateflow/sf_ug.pdf

[5] Architecture analysis & design language (AADL). 2014. http://http://www.aadl.info/

[6] Henzinger TA, Horowitz B, Kirsch CM. Giotto: A time-triggered language for embedded programming. Technical Report, Berkeley: Department of Electronic Engineering and Computer Science, University of California, 2001.

[7] Moszkowski BC, Manna Z. Reasoning in interval temporal logic. In: Clarke EM, Kozen D, eds. Proc. of the Logic of Programs. LNCS 164, Heidelberg: Springer-Verlag, 1983. 371−382.

[8] Dutertre B. Complete proof systems for first order interval temporal logic. In: Proc. of the 1995 ACM/IEEE Symp. on Logic in Computer Science. 1995. 36−43. [doi: 10.1109/LICS.1995.523242]

[9] Alur R, Ivancic F, Kim JS, Lee IS, Sokolsky O. Generating embedded software from hierarchical hybrid models. In: Proc. of the ACM SIGPLAN Conf. on Language, Compiler, and Tool for Embedded Systems, Vol.38. 2003. 171−182. [doi: 10.1145/780732.780756]

[10] Yang MF, Wang Z, Pu GG, Qin SC, Gu B, He JF. The stochastic semantics and verification for periodic control systems. Science China: Information Sciences, 2012,55(12):2675−2693. [doi: 10.1007/s11432-012-4750-0]

顾斌(1968－),男,湖北汉川人,研究员,CCF 高级会员,主要研究领域为嵌入式系统,可信软件设计与验证,计算机控制.

董云卫(1968－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为嵌入式系统,信息物理融合系统,可信软件设计与验证.

王政(1986－),男,博士,工程师,主要研究领域为嵌入式软件分析与设计,软件形式化方法.