基于活动图模型的测试数据生成研究

黄 晨[1,2],于 倩[1,2],虞砺琨[1,2],左万娟[1,2],陈华南[2]
(1. 北京控制工程研究所,北京 100190;
2. 北京轩宇信息技术有限公司,北京 100190)

摘 要:星载嵌入式软件功能逻辑复杂,多个功能模块的数据存在强关联性和强耦合性,人工设计用例对测试需求的覆盖情况难以准确衡量和评估。该文采用半形式化的 SysML 活动图模型对星载嵌入式软件中的典型功能进行建模研究,为了满足测试中的有效等价类、无效等价类、边界值、MC/DC 等用例设计及数据覆盖要求,提出了需求结构覆盖准则和数据表达式覆盖准则,将测试场景与数据进行结合,经验证,针对复杂的转换型功能采用控制流、数据流的活动图建模方式,能够满足测试中的用例设计充分性以及数据覆盖性要求。

关键词:活动图;需求结构覆盖;数据表达式覆盖;控制流;数据流
中图分类号:TP311.5 **文献标识码**:A **文章编号**:1673-629X(2022)0104-06

Research on Test Data Generation Based on Activity Graph Model

HUANG Chen[1,2], YU Qian[1,2], YU Li-kun[1,2], ZUO Wan-juan[1,2], CHEN Hua-nan[2]
(1. Beijing Institute of Control Engineering, Beijing 100190, China;
2. Beijing Sunwise Information Technology Co., Ltd., Beijing 100190, China)

Abstract: The functional logic of space-borne embedded software is complex, and the data of multiple functional modules have strong correlation and coupling. It is difficult to accurately measure and evaluate the coverage of test requirements by manually designed use cases. In this paper, the semi formal SysML activity diagram model is used to model the typical functions in space-borne embedded software. In order to meet the requirements of use case design and data coverage such as effective equivalence class, invalid equivalence class, boundary value and MC/DC in testing, the requirements structure coverage criteria and data expression coverage criteria are proposed. The test scenario is combined with the data, which is verified, for complex transformation functions, the activity diagram modeling method of control flow and data flow is adopted, which can meet the requirements of use case design adequacy and data coverage in testing.

Key words: activity diagram; requirement structure coverage; data expression coverage; control flow; data flow

0 引 言

航天嵌入式软件具有多系统、高复杂性、高可靠高安全性等要求,代码的规模和复杂度显著提升,软件研制趋于敏捷化,测试迭代周期变短,需求更动迭代快。传统的测试方法存在明显不足,基于文档的需求分析模式,需求描述方法存在二义性;用例设计的质量和效率主要依赖于人员经验,缺乏规范性和标准性;测试用例设计对测试需求的覆盖性难以有效跟踪和评估。基于模型的测试属于软件需求形式化的一种体现,具有通用性、无二义性、复用性等特点,能够较好的解决当前存在的问题,模型的语义和语法定义严格,能够描述软件对象的结构和行为,使用模型描述测试需求,依据输入的需求规格说明建立测试模型抽象程度好,可以较好地指导功能测试、系统测试;模型可直接针对需求和设计进行检查和验证,尽早发现分析设计阶段的问题;建模过程可与测试、设计、编码过程并行,从而缩短软件的研制周期[1]。

流程图可以表达活动的执行过程,但是业务流程除了活动过程之外,还要理清活动和执行者、活动之间的对象流、以及执行的时间条件,业务流程建模及表示法 BPMN2.0(Business Process Model and Notation)可以对业务流程的各种元素进行精确建模,它规范详细的描述了模型元素的意义和使用规则,提高了业务流程描述的清晰度和准确性,但是它缺乏严格的形式化描述,自身不能直接推导来进行模型检验[2],文献[3]将 BPMN 图元向 Petri 网模型进行演算映射并验证,来

收稿日期:2021-10-19
作者简介:黄 晨(1983-),女,高级工程师,研究方向为软件测试技术。

保证模型的正确性。SysML 建模语言的活动图模型也可用于描述系统或子系统级的工作流程,具有形式化的语义表示[1,4]。测试用例输入数据的选择非常关键,对于数据处理、转换型的功能,测试数据主要是由一些基本类型的数值组合而成,对于交互式、反应式的功能,操作序列无疑也是测试用例的重要组成部分。

该文比对了两种建模语言及方法,结合星载嵌入式软件特点,选取 SysML 活动图建立软件功能模型,根据建立的测试场景,对活动图进行简化转换,从控制流和数据流的角度,对模型中的表达式进行求解和数据自动生成,采用两种覆盖准则,保证测试用例集的覆盖性。

1　建模语言及方法

业务流程图和活动图是当前测试建模中使用非常多的模型之一,这两种建模语言属于不同的建模标准。目前,MBSE 主要使用 SysML 作为建模标准语言[5],多个领域在进行系统分析时,也会采用流程图进行需求分析。

1.1　BPMN 流程图

业务流程建模和表示法(Business Process Model and Notation,BPMN2.0)是一种构建业务流程图的建模语言标准,它是由对象管理组(Object Management Group,OMG)创建的,主要目标是提供被所有业务用户(包括业务分析者、软件开发者以及业务管理者等)理解的一套标记语言。BPMN2.0 规范具有 3 种图,协作图、编排图和流程图。协作图通常包含两个及以上的多个池,代表相互协作的参与者,相互协作过程中通过消息流进行交互;编排图也是一种流程图,注重业务参与者的交互,以及消息交互;流程图描述了一个组织内部开展一定目的的工作所需进行的有序活动,执行语义有限。

BPMN2.0 定义的流程图是由图形对象组成的网状图,图形对象包括活动和用于定义这些活动执行顺序的流程控制器。BPMN 流程图法继承了许多业务过程建模符号中的元素,流程图由活动节点和控制节点组成,活动节点用于描述事件或任务,控制节点用于描述活动间的控制流。BPMN 着重描述处理过程,面向过程建模,它的主要控制结构有顺序、分支和循环,各个处理过程之间存在严格的顺序和时间关系,缺乏形式化的语义和分析技术,产生的过程模型容易出现一系列的语义错误,如死锁、活锁[3]。

1.1.1　图元表示

BPMN 流程图所包含的图元主要分为五类:流对象、数据、连接对象、泳道和描述信息。

流对象分为活动、事件和网关三类。活动分为两种类型可以是任务或者是当前流程的子处理流程,子流程嵌入在主流程中,可以被主流程或其他流程调用;事件分为:开始、中间、结束;网关用来控制业务流程走向,排他网关只有一条路径会被选择、并行网关所有路径会被同时选择、包容网关可以同时执行多条线路,也可以在网关上设置条件、事件网关专门为中间捕获事件设置,允许设置多个输出流指向多个不同的中间捕获事件,当流程执行到事件网关后,处于等待状态,需要等待抛出事件才能将等待状态转换为活动状态。

数据分为数据对象、数据输入、数据输出、数据存储。连接对象分为顺序流、消息流、关联、数据关联。泳道,分为池、道,表示流程的参与者,用来对活动进行组织和分类。描述信息,包含分组和文本注释。

1.1.2　执行语义

BPMN 规范定义了许多用于定义流程的语义概念,并将它们与图形元素、标记和连接相关联。定义如下:

$$M=(A,G,E,S) \tag{1}$$

其中,A 表示活动,表示在流程图中具备声明周期状态的元素;G 表示网关,主要包含 XOR 网关、OR 网关和 AND 网关;E 表示事件,利用事件机制,为系统增加辅助功能,S 表示流向[6]。

1.2　SysML 活动图

系统建模语言(Systems Modeling Language,SysML1.4)作为系统工程应用开发的标准建模语言被 OMG 提出,是目前国际上系统工程领域最新的标准建模语言,使用自然语言描述约束和详细语义,涵盖了从系统需求到设计阶段的各项要求,广泛应用于复杂系统建模,已应用于航空航天软件开发过程[7-8]。

SysML 为系统提供了四种类型模型(结构模型、行为模型、需求模型和参数模型)的九种图(模块定义图,内部模块图,包图,活动图,序列图,用例图,状态机图,参数图和需求图)的完整语义,见图 1。

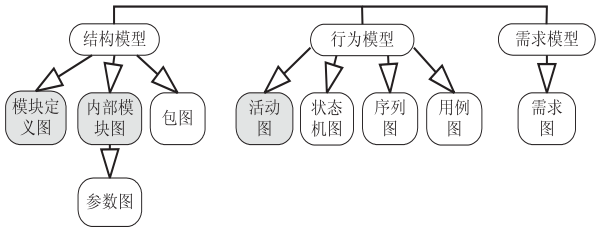

图 1　SysML 模型分类及组成

行为模型强调系统中对象的行为,包括它们的活动、交互和状态历史。活动图属于行为模型的一种,通过描述一个流程中各项操作的顺序来表示一个系统的行为视图,可显示一个活动中各项操作之间的流程,还可以显示并行或并发流程和备用流程。活动图使用活动节点和活动边来对操作之间的流对象进行建模,如

控制流和数据流,通过控制流能够清楚的知道系统内部的执行流程,通过数据流可以对系统运行测试数据的实际输出和预期输出相比较,能够知道系统是否正常运行。

1.2.1 图元表示

活动图的图元[9]主要是节点和活动边。

(1)节点。

a. 活动节点:活动由一系列不连续的活动节点组成,当操作执行完毕、对象和数据变得可供操作使用、在流外部发生了事件,就会调用活动中的动作。活动可以有子活动,有入口动作和出口动作,也可以有内部转移。

b. 动作节点:不可再分的原子动作、调用动作、接收信号、发送信号等,动作节点存在 effect 影响。

c. 控制节点:初始节点是所有活动的起点;终止节点是子活动流程的结束;结束节点是整个活动流程的结束;分支与合并,分支可以有一个进入流和两个或多个离去流,在每个离去流上放置一个布尔表达式,在进入这个分支时被判断一次,在所有离去流中,监护条件不应该重叠,应覆盖所有的可能性;合并无需监护条件;分叉与汇合与并发结构相关。

(2)活动边。

活动边是两个活动节点之间的有向连接,当完成一个活动中的特定操作时,活动边会继续将流引向该序列中的下一项操作,活动边表示节点之间的迁移关系,分为控制流和对象流两种。

a. 控制流:用来对控制流从一个节点流向另一个节点的情况进行建模。当一个动作或活动节点结束执行时,控制流将马上传递到下一个动作或者活动节点,类似于控制流程,对象是前一个活动的输出,后一个活动的输入。

b. 对象流:分为数据流和信号流,用来对数据流或信号流从一个节点流向另一个节点的情况进行建模。数据流表示的是一种数据流关系,对象表示的是一个活动的输入或输出;信号流将一个信号发送、信号接收与一个对象相连接,表示向该对象发送、由该对象接收一个信号[1]。

1.2.2 形式化语义

活动图包括节点和边两大类图元[10-11],数学模型可表示为:

$$AD = (A_D, T_D) \tag{2}$$

其中,A_D 表示节点的集合,T_D 表示边的集合。

$$A_D = \{I_D, F_D, S_D, C_D, B_D, M_D, K_D, J_D, G_S, G_r, O_D\} \tag{3}$$

初始节点 I_D 和终止节点 F_D、基本活动 S_D 和组合活动 C_D,条件分支 B_D 及其汇聚结点 M_D,还包含了并发分支 K_D 和并发汇聚结点 J_D,信号发送 G_S 和信号接收活动 G_r,以及对象节点 O_D,组合活动实际上是另一个嵌套的子活动图。

活动边集合可定义为:

$$T_D = \{I_F, S_F, D_F\} \tag{4}$$

活动边表示的是某种流的关系,分为控制流 I_F,信号流 S_F 和数据流 D_F。迁移的发生需要满足一定的约束条件,可以用活动边上的标识来描述:

$$I_F:(A_D, F_L) \rightarrow A_D,\ S_F:(G_S, F_L) \rightarrow G_r,\ D_F:(O_D, F_L) \rightarrow A_D \tag{5}$$

其中,F_L 表示迁移条件,当发生了某个事件并且迁移条件已经满足时将产生迁移,同时,在发生迁移时,会伴随着触发某个动作。活动边上设置 * 表示迭代,表示反复发生的事件将触发活动的反复执行,可以依次或者并发执行。

1.3 分析比较

1.3.1 相同点

(1)通用性:都具备可视化的模型和描述能力,基于标准建模语言,具有广泛的适用性,具有多个通用性工具的支持,并持续维护和升级。

(2)重用性:多个研制阶段、软件版本的模型可重复使用。

1.3.2 差异性

(1)表示形式:活动图是 SysML 建模语言的一种表现形式,可用于构造和记录软件系统的工件,主要集中在软件系统建模,它采用面向对象的方法进行建模;业务流程图主要目标是提供一个易于所有业务用户理解的符号,主要集中在业务过程建模,侧重于建模业务流程,它采用面向过程的方法来进行系统建模[12-13]。

(2)形式化定义:活动图将自然语言与某种规约的表示方法相结合,侧重于标准语言,通过定义元模型统一语义并且构建通用符号,对系统行为的描述更加清晰、准确,可采用形式化的数学模型进行描述,避免了测试需求建模的二义性;流程图缺乏准确的语法及形式化的语义定义,它并不是一种设计方法,而是用于记录设计的表示法,其主要目标是提供一个易于所有业务用户理解的符号,仅面向过程的方法来建模系统,仅涵盖符号元素的描述,表现形式相对简单。

(3)顺序性:活动图主要体现用户与系统的交互行为,描述活动所必要的工作顺序;业务流程图明确指定了每个活动的先后顺序。

(4)面向对象与面向过程:活动图面向对象,描述的是对象活动的顺序关系所遵循的规则,着重表现的是系统的行为,是系统的一种动态图,说明随着时间的推移,行为和时间的发生序列,通过行为表示对象事件、信号或者数据的流动,关注系统操作时对象是如何

在行为的执行过程中被访问和修改的;业务流程图面向过程,它的主要控制结构是顺序、分支和循环,各个处理之间有严格的顺序和时间关系。

(5)实际应用:活动图应用于主要用于 Web 应用程序,商业应用程序,嵌入式系统等领域,可用于生成代码和测试用例;业务流程图的应用形式涵盖多个领域的系统模型,包括银行和金融服务,电信,零售,运输等。

1.3.3　小　节

总的说来,基于 SysML 标准的活动图所表示的语义和语法更加丰富,使用范围更广,对于复杂控制逻辑、业务流程、并发活动的描述更加强大,星载嵌入式软件计算算法复杂、业务逻辑与背景知识相关,数据、时间特性显著等,大多数模块难以完全采用业务流程图进行描述和表示,对于系统内部行为较多而外部触发较少,算法类的、逻辑复杂的内部流程难以用状态转换进行刻画,针对这类转换类功能模块,为能够更加全面、准确的描述其复杂逻辑,使用活动图对其进行建模和分析,生成测试用例。

2　测试用例及数据生成

测试用例是针对软件的某些有预期质量要求的特性指标,以及软件自身特性规格说明中的规定、软件的内部结构和外部环境特性等,为达到或满足预定的软件测试目标而设计和生成的测试数据、操作序列和与之对应的预期输出结果的集合[1]。通过解析活动图模型得到的测试路径即为测试场景,根据选定的测试覆盖准则,将测试场景与数据流、对象流或者消息流相结合,实现需求结构覆盖、数据覆盖的集合,称为测试用例集。

测试用例集的数学表达式为:

$$TC = (Path, Data) \tag{6}$$

其中,Path 表示活动图某条测试路径,由活动图中一系列相关的活动组成,包括活动的转移信息,触发信息等;Data 表示测试数据,对应于某条测试路径的输入数据信息,以及相应的预期输出结果[14-15]。

2.1　覆盖准则

测试用例集是满足一定覆盖准则的测试用例的集合,可以表示为:

$$TS = \{RSC, DEC\} \tag{7}$$

根据覆盖范围将覆盖准则分为两类,需求结构覆盖 RSC(Requirement Structure Coverage)确认选择的测试用例对需求结构的覆盖程度是否满足测试要求或者标准;数据表达式覆盖 DEC(Data Expression Coverage)是分析测试用例对测试用例设计方法的覆盖程序。

(1)需求结构覆盖。

可理解为对模型图元的覆盖准则。

a. 节点覆盖:路径集合应覆盖模型中全面的节点;

b. 边覆盖:路径集合应覆盖模型中全部的边;

c. 基本路径覆盖:所有边至少执行 1 次;

d. 单路径覆盖:所有节点至少执行 1 次。

(2)数据表达式覆盖。

可理解为针对不同类型的表达式以及求解表达式数值的覆盖准则,如:逻辑表达式,关系表达式,算术表达式等。

a. 逻辑表达式:具有逻辑操作符(‖、&&),应满足表达式 MC/DC 覆盖要求;

b. 关系表达式:具有关系运算符(<、<=、>、>=、==、!=),应满足模型定义的输入域范围内,表达式的有效值、无效值、边界值覆盖要求;

c. 枚举表达式:具有关键字 Switch case、if, else if, else 等类型表达式,应满足所有枚举值覆盖要求,例如:Switch(i){case 0:break;case 1;break;}、If(a<100){}else if(a<200){}else if(a<300){}else{};

d. 算术表达式:具有算术、赋值运算符(+、-、*、/、=、++、--),该类型表达式的数据值为确定的,会对变量的计算结果产生影响,一般不考虑其覆盖要求。

2.2　测试场景

测试场景生成时需要模块模型的结构,将模型经过简化、转换后形成有向图,需保证遍历和访问有向图中的每一个节点,采用深度优先遍历,将所有节点进行未访问的初始化,从图中节点开始,依次访问未访问过的节点,递归调用,直到与节点相连的所有节点都被访问。

针对活动图 5 种基本结构:顺序、循环、嵌套、并发和泳道,简化原则为:

a. 顺序结构:只存在动作、活动、判定等顺序执行的节点,活动图的控制流和对象流都是按照顺序进行的。顺序结构转换有向图时无需简化。

b. 循环结构:根据活动状态执行的情况,动作活动可能被执行多次的结构。循环结构需要限定循环次数,保证对循环的有效覆盖,并且缩短遍历时间。循环结构简化时抽象为两层,外层为复合节点,内层为顺序实现。

c. 嵌套结构:在嵌入式系统中系统功能复杂的情况下,会存在多个功能模型嵌套调用的结构关系。嵌套结构简化时抽象为两层,外层为复合节点,内层为顺序实现。

d. 并发结构:在嵌入式系统中多个功能模块存在并发性,实时性较高,并发模型能够较好的体现嵌入式系统的实时性等特征。并发结构相对复杂,活动、动作

之间执行顺序不受限制,可任意组合,导致用例数剧增,为了优先测试大概率路径,可通过设置权重的方式对全部路径的执行权重进行排序。并发结构简化时可将其抽象为两层,外层为复合节点,内层为顺序实现。

e. 泳道结构:主要是面向不同用户或者角色的操作,可转换为以上四种类型的结构。泳道结构简化时不考虑泳道,简化原则同其他结构。

2.3 测试数据及组合

根据活动图模型所表示的语义,从控制流和数据流两个角度(见图 2),考虑测试数据生成方法。

```
struct dataInput        //输入数据结构
{
    string time;        //注入时刻
    string name;        //名称
    string type;        //类型
    string  dataMin;//最小值
    string dataMax;//最大值
    string delta;       //步长
}
```

(a)输入结构

```
struct dataOutput       //输出数据结构
{
    string name;        //名称
    string type;        //类型
    string value;       //数值
}
```

(b)输出结构

图 2 数据结构定义

(1)控制流。

活动图描述的是某个功能的控制流程,即类似多个功能模块之间的调用关系时,对模型解析和转换后得到所有路径表达式的路径树,过滤不可达路径,仅保留可执行路径。

如果前后两个测试步骤的输入数据无直接的约束关系,需要考虑所有的数据仅覆盖一次,以及多次组合覆盖的情况。

(2)数据流。

活动图描述的是数据的流向,需对每条路径上变量的数据操作符号进行标注,根据选定数据覆盖策略得到测试路径,以及相应的输入数据。

如果数据流上的多个表达式是针对同一个变量的操作和判定,即多个表达式存在约束关系,需要根据模型的语义将表达式进行组合,求解公共数据解。

3 实例分析

(1)实例 1:基于控制流的活动图模型及用例(见图 3 和表 1)。

表 1 测试用例集

用例	测试输入	预期输出	说明
用例 1	T=0; F1=FALSE; F2=FALSE	A=2; B=10	输入域边界值 表达式 MC/DC
用例 2	T=1; F1=FALSE; F2=TRUE	A=2; B=20	输入域边界值 表达式 MC/DC
用例 3	T=20; F1=TRUE; F2=FALSE	A=2; B=20	输入域 表达式 MC/DC

续表 1

用例	测试输入	预期输出	说明
用例 4	T=29; F1=FALSE; F2=FALSE	A=2; B=10	输出域边界值 表达式 MC/DC
用例 5	T=30; F1=FALSE; F2=TRUE	A=2; B=20	输出域边界值 表达式 MC/DC
用例 6	T=31; F1=TRUE; F2=FALSE	A=1; B=20	输出域边界值 表达式 MC/DC
用例 7	T=40; F1=FALSE; F2=TRUE	A=1; B=20	输入域 表达式 MC/DC
用例 8	T=59; F1=TRUE; F2=FALSE	A=1; B=20	输入域边界值 表达式 MC/DC
用例 9	T=60; F1=FALSE; F2=FALSE	A=1; B=10	输入域边界值 表达式 MC/DC

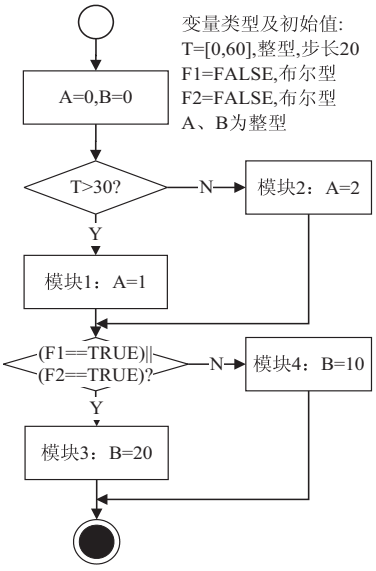

图 3 控制流模型

分析:基于控制流的活动图模型,根据自动识别表达式的类型并生成满足 MC/DC 覆盖要求的测试用例集,并对多组输入数据进行交叉组合测试。

(2)实例 2:基于数据流的活动图模型及用例(见图 4 和表 2)。

表 2 测试用例集

用例	测试输入	预期输出	说明
用例 1	T=0	T0=0, T1, T2	输入域边界值
用例 2	T=1	T0=1, T1, T2	输入域边界值
用例 3	T=20	T0=20, T1, T2	–
用例 4	T=40	T0=40, T1,T2	–
用例 5	T=50	T0=50, T1, T2	输出域边界值
用例 6	T=51	T0=51, T0, T2	输出域边界值

续表 2

用例	测试输入	预期输出	说明
用例 7	T=60	T0=60, T0, T2	-
用例 8	T=80	T0=80, T0, T2	-
用例 9	T=99	T0=99, T0, T2	输入域边界值
用例 10	T=100	T0=100, T1, T2	输入域边界值

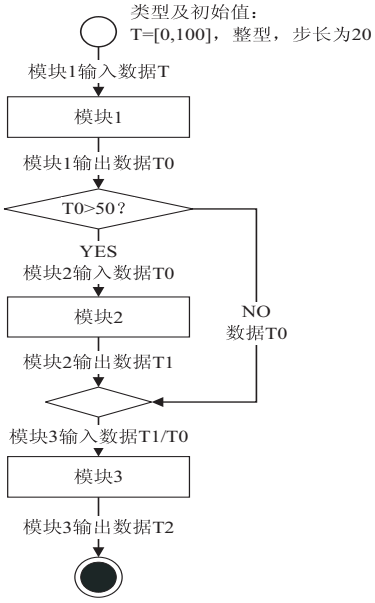

图 4　数据流模型

分析:基于数据流的活动图模型,能够根据设定的外部数据类型、范围和步长完整的遍历整个测试输入域,有效覆盖模型中的全部路径,快速建立输入和输出数据的对应关系。

4　结束语

比对了两种建模语言及方法,根据应用领域软件对象的多种特性进行比较,选择 SysML 活动图作为建模语言,采用需求结构覆盖和数据表达式覆盖两种覆盖准则,建立测试场景,从控制流和数据流的角度,对模型中的表达式进行约束求解,生成满足测试要求的测试数据,对前后序列的数据进行组合,形成测试用例集,并采用两种类型的典型模型进行实例化说明。后续将继续研究模型验证的方法,保证模型的正确性,实现模型驱动测试在航天领域的工程化应用。

参考文献:

[1] 粘新育. 基于 UML 活动图模型的测试用例生成方法的研究[D]. 济南:山东大学,2007.

[2] 王克丽. 复杂信息系统流程验证及统一建模平台实现研究[D]. 太原:太原理工大学,2016.

[3] 赵　莹,赵　川,黄　苾,等. BPMN2.0 过程模型的语义和分析[J]. 计算机科学,2018,45(11A):558-563.

[4] TEIXEIRA F A D, SILVA G B. EasyTest: an approach for automatic test cases generation from UML activity diagrams [C]//14th international conference on information technology: new generations (ITNG 2017). Florestal: [s. n.],2016.

[5] 王文浩,毕文豪,张　安,等. 基于 MBSE 的民机系统功能建模方法[J]. 系统工程与电子技术,2021,4(12):1-10.

[6] 汪玉泉,闻立杰,闫志强. 基于对齐的 BPMN2.0 模型符合性检测算法[J]. 计算机研究与发展,2017,54(9):1920-1930.

[7] 王松锋. SysML 行为图到 Petri 网的转换研究[D]. 郑州:解放军信息工程大学,2012.

[8] YIN Yufei, XU Yiqun, MIAO Weikai, et al. An automated test case generation approach based on activity diagrams of SysML[J]. International Journal of Performability Engineering,2017,13(6):922-936.

[9] FRIEDENTHAL S, MOORE A, STEINER R. A practical guide to sysML, the systems modeling language[M]. Morgan Kaufmann Publishers, Elsevier Inc,. 2012:205-250.

[10] 张　楣,刘　超,孙昌爱. 基于 UML 活动图模型的测试用例生成技术研究[J]. 北京航空航天大学学报,2001,27(4):433-437.

[11] DUMAS M, HOFSTEDE A H M. UML activity diagrams as a workflow specification language [C]//4th international conference on the unified modeling language, modeling languages, concepts, and tools. Toronto, Canada: [s. n.],2001: 76-90.

[12] GEAMBASU C V. BPMN VS. UML activity diagram for business process modeling[J]. Accounting and Management Information Systems,2012,11(4):637-651.

[13] KIM H, KANG S, BAIK J, et al. Test cases generation from UML activity diagrams[C]//Eighth ACIS international conference on software engineering, artificial intelligence, networking, and parallel/distributed computing (SNPD 2007). Qingdao, China: IEEE,2007:556-561.

[14] 祝玉芬,刘　超. 基于活动图模型测试剖面的测试用例生成方法[J]. 计算机工程,2003,29(21):45-47.

[15] WANG Linzhang, YUAN Jiesong, YU Xiaofeng, et al. Generating test cases from UML activity diagram based on gray-box method[C]//Proceedings of the 11th Asia-Pacific software engineering conference (APSEC'04). Busan, South Korea: IEEE,2004:284-291. L