基于 UML 活动图模型的测试用例生成技术研究

张 楣 刘 超 孙昌爱
(北京航空航天大学 计算机科学与工程系)

摘 要:为了设计和生成系统工作流程的测试用例,引入测试大纲模型的概念,设计出从 UML(Unified Modeling Language)活动图模型到测试大纲模型,再到测试用例模型的三级转换过程和一组消除活动图模型中的非结构化特征、将其转化为模块化的测试大纲模型,以及把并发子过程实例化为一组典型测试场景的基本规则.还研究了针对每一个输入操作,选择基本输入数据集,并将其赋予测试大纲模型之上,从而构造测试用例模型的方法,以及最终基于测试用例模型生成完备的测试用例集合的方法.

关 键 词:计算机辅助测试;计算机设计自动化;软件工程;测试用例的自动生成;UML 活动图
中图分类号:TP 311.56
文献标识码:A **文 章 编 号**:1001-5965(2001)04-0433-05

在测试用例自动生成方法上,Tsai 等提出了从关系代数查询表示的规格说明中自动生成测试用例的方法[1],Weyuker 等提出了基于布尔规格说明的测试数据自动生成方法[2],工具 ConData[3] 采用基于协议规格语言(PSL)、结合多种不同的测试方法(等价划分等)生成测试数据.研究表明,采用有效的形式化方法来描述问题域,并选择适当的测试策略,是设计和生成完备的测试用例集合的关键环节.

在用例模型(Use Case Models)驱动的面向对象开发方法中,UML(Unified Modeling Language)活动图既是描述系统用例工作流程的重要建模工具,同时又是设计系统操作和功能实现方法的工具[4],[5].活动图本质上是一种自动机模型,它着重描述系统为完成指定功能或任务所必须执行的活动序列.而对于复杂的交互式和并发软件系统而言,各种复杂的操作流程无疑是测试的重要内容.因此,活动图成为软件功能测试,特别是面向操作流程的测试的重要依据.

1 基于 UML 活动图的测试用例设计

1.1 UML 活动图模型

UML 活动图描述了为实现系统用例所要进行的活动以及活动间的约束关系,即系统用例的操作规程.运用常规的集合概念,活动图模型可表示为

$$D_A = (A_D, T_D) \tag{1}$$

其中,A_D 称为活动图的结点集合;T_D 称为边集合.

与传统的控制流程图相似,活动图中包括多种不同类型的节点,分别表示初始结点(I_D)和终点活动(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).此外,它还包含了并发分支(K_D)和并发汇聚结点(J_D)、信号发送(G_s)和信号接受活动(G_r),以及对象结点(O_D).组合活动实际上是另一个嵌套的子活动图.在一个活动结点中,通常包含一个基本活动和若干个可以由特殊事件触发的动作.因此,结点集合可定义如下:

$$A_D = \{I_D, F_D, S_D, C_D, K_D, B_D, J_D, M_D, G_s, G_r, O_D\} \tag{2}$$

活动图中的迁移边也有多种类型,分为控制流关系(I_F)、消息流关系(S_F)和对象流关系(D_F).因此,迁移边集合的定义为

$$T_D = \{I_F, S_F, D_F\} \tag{3}$$

其中

收稿日期:2000-09-20
基金项目:国家 863 高技术计划资助项目(863-306-ZD02-03-7)
作者简介:张 楣(1976-),女,四川成都人,硕士生,100083,北京.

$$I_F : (A_D, F_L) \rightarrow A_D \quad (4)$$

$$S_F : (G_s, F_L) \rightarrow G_r \quad (5)$$

$$D_F : (A_D, F_L) \rightarrow O_D \quad \text{or} \quad (O_D, F_L) \rightarrow A_D \quad (6)$$

一般而言,这些边代表某种“流关系”,是活动对事件做出的响应.迁移的发生需要满足一定的约束条件,这种迁移条件可以用迁移边上的标识来描述,即

$$F_L = (E_L, C_L, A_L) \quad (7)$$

一个迁移标识 $f_1 \in F_L$ 表示当发生了某个事件($e \in E_L$)并且迁移条件($c \in C_L$)已经满足时将产生迁移.同时,在发生迁移时,可能会伴随着触发某个动作($a \in A_L$).如果在迁移边上带有“*”则表示迭代,即反复发生的事件将触发活动的反复执行.这种活动的反复执行可以是依次进行的,也可以是并发执行的.

1.2 测试用例设计过程

一般而言,测试用例是指依据某个特定的测试要求设计的用于驱动被测软件从某一状态(如初始状态)执行到另一个预定状态(如终止状态)所需的外部设置、操作指令和输入数据的集合.在大型复杂软件的测试实践中,特别是对于分布式和交互式系统的测试,测试用例通常包括一个特定的测试场景和与之相对应的一组输入数据(包括操作指令、输入数值和初始化设置值等)两个部分.测试场景描述了系统的一个特定的执行流程.依据一定的测试准则设计的所有测试场景的集合组成一个测试大纲.输入数据则是指在测试场景中对于那些请求输入的交互操作赋予的具体输入值.这些值可以是初始设置中的一个环境变量的值,也可以是某个外部对象发送给本系统的一个操作指令或者数据值.为此,测试用例的设计通常应当分为两个部分,即基于系统活动图模型的测试大纲的设计和基于输入约束条件的输入数据的设计.

不言而喻,活动图的基本测试覆盖准则应当包括结点覆盖测试、迁移边覆盖测试、关键路径覆盖测试以及对应于所有输入操作的输入数据空间的覆盖测试等覆盖准则[6].然而,由于活动图存在非结构化和并发特性,以及复杂的输入数据类型、多种输入方式和非单一的输入对象等因素,使得在根据活动图来设计测试大纲和测试数据时,需要有效地解决以下几个关键问题:

1) 活动图的模块化划分方法,以解决复杂活动图中非结构化部分的测试大纲设计和描述问题;

2) 并发流程的实例化方法,即根据测试要求来设计用于测试并发流程的测试大纲和测试用例的问题;

3) 消息流与对象流的测试问题.

为此,提出一种基于测试大纲模型的测试用例生成方法,其基本过程如图1所示.该框架强调依据UML活动图模型,首先将活动图转换成一个具有良好的模块化结构的测试大纲模型,然后再生成测试大纲.

与此同时,依据活动图中对于各种交互操作的描述,获取各个交互操作的输入数据空间定义,依据一定的测试准则设计生成基本测试数据集.基本测试数据集应当包括合法数据、非法数据、边界数据等.然后再依据已经生成的测试大纲和基本测试数据集构造测试用例模型,并依据一定的优化组合策略来设计和生成最终的测试用例集合的基本方法.

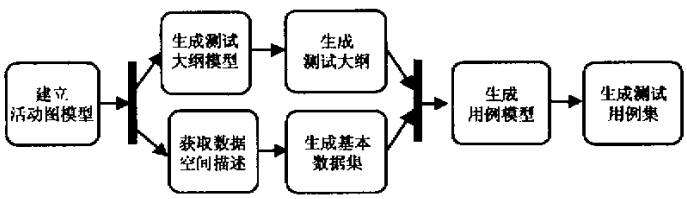

图1 基于UML活动图的测试用例自动生成模型

2 测试大纲模型

2.1 测试大纲模型定义

对于一个给定的活动图 $D_A = (A_D, T_D)$,它的测试大纲模型为

$$D_O(D_A) = (A_O, L_O, T_O) \quad (8)$$

其中 $A_O = \{I_O, F_O, S_O, C_O, \text{NULL}\}$

$L_O = \{\text{OR}, \text{AND}, \text{COR}, \text{CAND}\}$

$T_O : (\{A_O, L_O\}, F_{LO}) \rightarrow \{A_O, L_O\}$

$F_{LO} = \{L \mid L \text{ 是迁移边上的标识}\}$

A_O 是活动的集合,其中,I_O, F_O, S_O, C_O 分别是 A_D 的 I_D, F_D, S_D, C_D 或者是它们的子集;L_O 称作

逻辑关联结点的集合,这些关联结点描述了一组相关联的场景之间的逻辑关系,包括异或关联(OR),与关联(AND),并发或关联(COR),并发与关联(CAND)等;T_O 是结点之间的迁移边的集合.直观地讲,异或关联结点带有一个迁入边和若干个迁出边,每一条迁出边表示一个不同的、相对独立的场景,即每一个场景只覆盖其中一个迁出边.与关联结点有多个迁入边和一个迁出边,每一条迁入边表示一个不同的、相对独立的场景,即每一个迁入边所代表的场景都包含其迁出边以后的场景片断.异或关联结点和与关联结点通常是成对出现的.并发或关联与异或关联相似,但是各条迁出边代表若干个并发的场景片断,它们会在同一个场景中出现,而先后顺序是不确定的.并发与关联和与关联相似,各条迁入边代表若干个并发的场景片断,但是并发与关联结点表示这些并发场景都结束时才会移向以后的场景.迁移边上带有 F_{IO} 中的一个标识,表示活动之间发生迁移的约束条件,包括无条件(NULL)迁移.

2.2　活动图模型转化为测试大纲模型

对任意的一个活动图 $D_A = \{A_D, T_D\}$,运用下列变换规则来构造其测试大纲模型 $D_O = (A_O, L_O, T_O)$.

规则 1　结点规则.D_A 中的所有初始结点、终结点、活动和组合活动结点均属于 A_O,有下列关系:

$$I_O = I_D \qquad F_O = F_D$$
$$S_O = S_D \qquad C_O = C_D$$

规则 2　串行迁移规则.如果 t 是某个 $A_1 \in A_D$ 的唯一的迁出边,则 $t \in T_O$.

规则 3　分支结点规则.对于结点 $A_1 \in A_D$,如果它有多条迁出边 $\{(A_1, L_{1,i}) \to A_i \mid i = 2,3,\cdots,n\}$,则增加一个逻辑关联结点 YOR,并将原 A_1 的迁出边集合变为 $\{(A_1', \text{NULL}) \to \text{YOR}\} + \{(\text{YOR}, L_{1,i}) \to A_i' \mid i = 2,3,\cdots,n\}$,其中:

1) 如果 A_1 是一个条件分支结点,即 $A_1 \in S_D$,则对应的测试大纲模型中 $A_1' = \text{NULL}$(因而可以省略),YOR = OR;

2) 如果 A_1 是一个并发分支结点,即 $A_1 \in F_D$,则对应的测试大纲模型中 $A_1' = \text{NULL}$(因而可以省略),YOR = COR;

3) 如果 A_1 是某种活动结点,即 $A_1 \in \{J_D, F_D, B_D, C_D\}$,则对应的测试大纲模型中 $A_1' = A_1$,YOR = OR.

规则 4　会聚结点规则.对于结点 $A_n \in A_D$,如果它有多条迁入边 $\{(A_i, L_{i,n}) \to A_n \mid i = 1,2,\cdots,n-1\}$,则增加一个逻辑关联结点 YAND,并将原 A_n 的迁入边集合变为 $\{(A_i', L_{i,n}) \to \text{YAND}:A_n' \mid i = 1,2,\cdots,n-1\} + \{(\text{YAND}:A_n', \text{NULL}) \to A_n'\}$,其中:

1) 如果 A_n 是一个汇聚结点,即 $A_n \in M_D$,则对应的测试大纲模型中 $A_n' = \text{NULL}$(因而可以省略),YAND = AND;

2) 如果 A_n 是一个并发汇聚结点,即 $A_n \in J_D$,则对应的测试大纲模型中 $A_n' = \text{NULL}$(因而可以省略),YAND = CAND;

3) 如果 A_n 是某种活动结点,即 $A_n \in \{J_D, F_D, B_D, C_D\}$,则对应的测试大纲模型中 $A_n' = A_n$,YAND = CAND.

规则 5　迭代规则.如果在迁移边上带有“*”,表示出现迭代,则将迭代过程分解为两种场景,即不执行迭代体和至少执行一次迭代体,并使用逻辑关联结点 YOR 将这两种场景连接在一起,其中 YOR 可以是 OR,或者 JAND.通常可以将 YOR 直接转换为 OR,即将其串行化.

在测试大纲模型中,实际上所有不含输入操作的活动和动作都可以用空活动(NULL)取代,以简化模型,突出交互过程.

显而易见,依据上述规则可以将一个活动图分解为一些简单的模块.每个模块描述了活动图中以某个活动或某一组活动为中心的控制流片段,它可以用一个代表性的(入口或出口)活动作为这个模块的标识.在测试大纲模型中,称每个模块为一个测试大纲单元.AND 和 CAND 中的标识符标明了单元之间的联系.从测试的角度看,单元中的每条路径代表着针对这个单元的一个典型的测试场景.

同样,消息流和对象流也可以被视为一些特定的测试场景.因此,可以首先运用下列规则将消息流和对象流转换为等效的控制流,然后运用规则 1 ~ 规则 5 生成测试大纲模型.

规则 6　同步消息流的转换规则.当两个活动 A_1 和 A_2 之间存在同步消息 $M_{1,2}$ 发送和接受时,则将此消息流视同为控制流,触发迁移的事件为 SendSyncto($M_{1,2}$).同时,再增加一个反向的携带返回消息 Return 的控制流,即 A_2 到 A_1 的迁移,其触发事件为 Return($M_{2,1}$).

规则 7 异步消息流的转换规则.当两个活动 A_1 和 A_2 之间存在异步消息 $M_{1,2}$ 时,则将消息流转换为 A_1 和 A_2 上的两个回授控制流,即指向自己的一个迁移边,A_1 上的事件为 Sendto(A_2, $M_{1,2}$),A_2 上的事件为 Sendfrom(A_1, $M_{1,2}$).

规则 8 对象流的转换规则.当两个活动 A_1 和 A_2 之间存在对象流 $D_{1,2}$ 时,则将对象流转换为 A_1 和 A_2 上的两个回授控制流,A_1 上的事件为 Sendto(A_1, $D_{1,2}$),A_2 上的事件为 Sendfrom(A_2, $D_{1,2}$).

如果活动图是结构化的,即对于所有的分支结点都有一个唯一的与之相对应的聚集结点,规则 3 与规则 4 将会成对应用,因此通过适当合并 YAND 结点可以方便地构成 AND-OR 树.

2.3 测试大纲生成

测试大纲(Test Outlines)是对测试大纲模型的实例化,其中测试大纲模型中的并发流程通常需进一步实例化,即需要确定各并发子流程之间各个活动的执行顺序.如果并发子流程的测试顺序是可控的,通常可以运用下列规则来选择并发流程的执行顺序.

规则 9 异步规则.当各并发子流程 $\{f_i\}$ 之间不存在消息收发时,即相互独立,则可以在每个子流程中的活动顺序不被颠倒的前提下,任选一个执行顺序,依次以串行方式执行这些活动.显然,依次执行各子流程是一种最简单的串行执行方式.为了提高对不同情况的测试覆盖,每次重复执行此并发流程时,可以采取优先选择尚未测试过的或者测试频度最低的执行顺序的方式,或其它优先级加权策略.

规则 10 同步规则.当两个并发子流程 f_1 和 f_2 中存在两个活动 A_1 和 A_2,它们之间存在同步消息 $M_{1,2}$ 时,则 A_1 排在 A_2 之前的执行序列被视为合法的测试场景,否则被视为是非法的测试场景.

实际测试中,并非任意一种执行顺序都是可能出现的.此外,通常也不可能穷尽测试所有可能的执行顺序.因此,合理的测试策略是:

1) 弱覆盖策略.至少一种执行顺序被测试;

2) 强覆盖策略.选定若干条期望测试的执行顺序,称为关键测试场景,要求所有关键测试场景都必须被测试到.为此通常需要建立一定的测试环境,通过对测试条件的设置来控制执行的特定顺序.

3 测试用例模型及测试用例的生成

3.1 测试用例模型

针对被测软件的交互活动流程,测试用例由一个描述这个流程的特定的测试场景和在此场景中依次出现的所有交互输入的数据值序列两个部分组成.因此,测试用例模型可表示为

$$D_T = (D_O, N_T) \tag{9}$$

D_O 是测试大纲模型,它清晰地描述了测试场景的集合.N_T 是对应测试大纲模型中所有输入操作的输入数据集合.

对于每一个交互输入,测试数据的选择应当考虑以下几点:

1) 合法的输入空间,即所有可能的合法输入数据的集合.针对活动图中每一个特定的输入操作,根据其合法输入数据集合的大小,可以将其分为实际测试可穷尽和不可穷尽的两类.实际测试可穷尽是指依据一定的测试准则,在给定的测试资源(包括人力、时间等)条件下,有可能进行穷尽测试.比如,对于一个含有若干个选项的选单应当被视为是实际测试可穷尽的,而对于某个温度参数,其输入值可以是[1,10]之间的一个实数,考虑到取值精度可以是小数点后若干位,因此可以被认为是实际不可穷尽的.一个输入数据集合是实际测试可穷尽的,则它必然是有限的;

2) 非法的输入空间,即所有可能出现的非法输入数据的集合.对于一个特定的输入操作,非法输入数据集合同样分为实际测试可穷尽和不可穷尽的两类.由于在很多情况下可能的非法输入集合是实际测试不可穷尽的,因此一般需要适当界定有必要进行测试的非法输入集合,称作待测试非法输入集合.比如,对温度参数的输入,界定待测试的非法输入集合为[-1 000,1)和(10,1 000];

3) 对输入的各种约束,包括等价类的划分、多个输入之间的依赖关系等.

在活动图模型中应当对工作流程中的每个输入操作给出明确的定义,包括输入操作的类型,值域的定义,输入者(人或其它系统),约束条件等.这些输入操作可以是一些输入活动或者是来自外部的触发事件.依据这些输入描述,可以抽取出针对每个输入操作的数据空间的定义,包括其合法数据空间、非法输入空间和一些重要的约束.在此基础上可以依据一定的测试策略设计和选择基本测试数据集合,即针对每一个输入操作,依据一定

的测试策略选定的一个实际测试可穷尽的数据集合.这些策略包括等价类划分、边界值分析等,也包括各种典型实例,比如,易引发错误的输入等.例如,对于上述温度输入参数,其合法取值区间为[1,10],待测试非法输入集合为[-1000,1),(10,1000],依据等价类划分原则,应当在每个区间中至少选定一个代表值,如 {0,3,100},根据边界测试策略,需在边界附近选择一组值,比如抽取{-1000,0.9,1,1.1,9.9,10,10.1,1000},从而得到一个由 11 个数据组成的基本测试数据集合:{1,1.1,3,9.9,10} + {-1000,0,0.9, 10.1,11,1000}.

3.2　测试用例生成

一个测试用例实际上是测试用例模型中的一个场景的实例,它是模型中各种基本元素的一种组合.这些基本元素包括模型中的所有测试大纲单元,每个单元中的所有活动结点和迁移边,以及每个输入操作的基本测试数据集合中的每一个输入值.测试用例的生成就是选定一个测试场景,并且针对场景中的每一个输入操作,从它的基本测试数据集合中选择一个输入值.测试的充分性则体现在测试用例模型中的每一个基本元素是否都被测试过.在构造测试用例的过程中,选择基本元素的基本原则为:

1) 覆盖原则.每个基本元素至少被选中一次;

2) 优先原则.重点测试的元素需优先选中,并且应当被多次选中.

4　结 束 语

基于 UML 活动图的测试用例生成技术的研究,避免了活动图的非结构化特性和并发特征给测试场景的设计和描述带来的问题,从而为面向操作流程的测试提供了重要的依据.其中基于 UML 活动图的测试大纲设计和生成技术得到设计实例验证.

参 考 文 献

[1] Tsai W T, Volovik D, Tkeefe T F. Automated test case generation for programs specified by relational algebra queries[J]. IEEE Trans on Software Engineering, 1990, 16(3): 316 ~ 324.

[2] Weyuker E, Goradia T, Singh A. Automatically generating test case data from a Boolean specification[J]. IEEE Trans on Software Engineering, 1994, 20(4): 353 ~ 363.

[3] Eliane Martins, Selma B Sabiao. ConData: a tool for automating specification-based test case generation for communication system [A]. In: IEEE Los Alamitos[C], 2000. 1060 ~ 3425.

[4] James Rumbaugh, Ivar Jacobson, Grady Booch. The unified modeling language reference manua[M]. An imprint of Addison Wesley Longman Inc, 1998.

[5] 刘　超,张　莉.可视化面向对象建模技术——标准建模语言 UML 教程[M].北京:北京航空航天大学出版社,1999.

[6] 刘　超.程序交互执行流程图及其测试覆盖规则[J].软件学报,1998,9(6): 458 ~ 463.

Automated Test Case Generation Based on UML Activity Diagram Model

ZHANG Mei　LIU Chao　SUN Chang-ai

(Beijing University of Aeronautics and Astronautics, Dept. of Computer Science and Engineering)

Abstract: In order to design and generate test cases of workflows, test outline model is defined. The three-tier process to transform an activity diagram into its test outline model and then into its test case model, as well as the test coverage criteria, is introduced, including a set of basic rules, by which unstructured flows is eliminated to modularize test outline units, and concurrency flows can be instantiated. The principles on the instantiated input data for each input operation in the system and the creation of the test case model by assigning the instantiated set of input data to the outline model, and then methods to generate the complete set of test cases based on the model are discussed.

Key words: computer-aided test; computer design automation; software engineering; test data auto-generation; UML activity diagram