基于 UML 活动图的测试场景生成方法研究①

Study on Method for Generating Test Scenarios Based on UML Activity Diagram

樊 鑫 舒 坚 刘琳岚 梁旗军 （南昌航空大学 计算机学院 江西 南昌 330063）

摘 要：本文研究运用统一建模语言（UML）活动图生成测试场景的方法，提出采用 McCabe 的基路径方法生成测试场景，首先对活动图进行压缩，采用基路径寻找算法找出其中的基本路径，并发活动实例化，替换找出的基本路径，形成完整的路径，据此生成相应的测试场景。文中对 UML 活动图的语法和语义进行了形式化定义和描述，详细描述了从压缩后的活动图中寻找基本路径的算法，对并发活动进行了约束。该方法所产生的基本路径相对独立，提高了测试效率。

关键词：测试场景 基路径 活动图 统一建模语言

1 引言

UML（Unified Modeling Language，统一建模语言）的易用性和规范性为软件测试技术的发展提供了契机。活动图是 UML 中用于描述业务流程的重要工具，它提供了一系列的图形符号来支持对业务流程的建模，并提供对条件、并行和反复的逻辑行为的支持[1]。UML 活动图本质上是一个流程图，可以用于描述活动到活动的控制流，还可以定义参与到活动中的对象及其角色、状态和属性的变化。基于 UML 活动图生成测试场景和依据测试场景自动生成测试用例的方法不仅适合于软件系统的测试，也适用于软件设计阶段对软件需求和设计模型的测试和验证[3]。

测试场景是与待测试软件的执行过程相对应的一个活动背景，它描述了系统的典型活动过程。待测试软件的各种功能经过细化，可以表示为具体的测试场景，生成测试场景是生成测试用例的前提。目前，一般通过定义一系列的测试覆盖准则并利用这些测试覆盖准则对 UML 模型进行遍历寻找测试场景。北京航空航天大学刘超等人提出的 PIEF（Program Interactive Execution Flowchart，程序交互执行流程图）及其基本覆盖测试准则为评判交互式软件功能测试的覆盖情况提供了一种客观的和规范化的度量准则[2]，在此基础上，他们进一步细化了针对活动图的控制点、转移边、逻辑路径和代表值的覆盖准则，并制定了在生成场景时对活动图信号流和对象流的覆盖准则[3]，南京大学王林章等人根据路径覆盖的准则，设计了一个从活动图到测试场景再到测试用例的自动生成工具 UMLTF[4]。

本文研究运用 UML 活动图生成测试场景的方法，提出采用 McCabe 的基路径方法寻找活动图的基本路径，对并发活动进行实例化，得到完整路径，从而生成测试场景。

2 活动图的形式化定义

UML 活动图主要包括活动状态、转移、转移条件、并发活动、分支汇聚结点等基本元素。为了更好地描述这些元素，对 UML 活动图进行如下定义[5]。

定义 1 一个活动图是一个六元组 D = (W,JF,BM,T,F,C)：

(1) W 表示非空的有限活动集，w_0 表示唯一初始活动，wn 表示唯一结束活动；

(2) JF 是并发分支与汇聚结点集；

(3) BM 是条件分支与汇聚结点集；

(4) T 是非空的有限转移集；

(5) C 是有限条件转移表达式，ci 对应 ti；

(6) $F \subseteq (W \times T \times C) \cup (T \times C \times W)$，表示活动与转移之间的流关系。

定义 2 设 D = (W,JF,BM,T,F,C) 是一个活动图，

① 基金项目 国家自然科学基金项目（60773055）

©《计算机系统应用》编辑部 http://www.c-s-a.org.cn

CW 为 D 的当前活动状态,对于所有的：

(1)$'t=\{w\in W\mid(w,t)\in F\}$、$t'=\{w\in W\mid(t,w)\in F\}$,分别表示 t 的前集和后集；

(2)enabled(CW)表示可以从 CW 触发的转移集,$enabled(CW)=\{t\mid t\in$满足$c(t)$条件的$CW\}$；

(3)fired(CW)表示某时刻只能从 CW 触发的转移集,$fired(CW)=\{t\mid t\in enabled(CW)\ and\ (CW-'t)\cap t'=\phi\}$,当 t 被触发,新的活动状态$CW'=(CW-'t)\cup t'$。如果满足条件的转移不唯一,可以选择其中任何一个未触发的转移；

(4)ep 表示 D 运行时的一个可执行路径,它是活动和转移的序列,$ep=CW_0\xrightarrow{t_0}CW_1\xrightarrow{t_1}\cdots\xrightarrow{t_{n-1}}CW_n$,$CW_0=\{w_0\}$,$CW_n=\{w_n\}$,$CW_i$ 为当前活动,$t_i=fired(CW_i)$,$i\geq 0$；$CW_i=\{(CW_{i-1}-'t_{i-1})\cup t'_{i-1}\}$,$i\geq 1$。

定义 3 设 D=(W,JF,BM,T,F,C)是一个活动图,TS 是 D 的测试场景,$ts\in TS$。TS 是一系列活动和条件转移,$ts=CW_0\xrightarrow{[c_0]t_0}CW_1\xrightarrow{[c_1]t_1}\cdots\xrightarrow{[c_{n-1}]t_{n-1}}CW_n$ 其中 $CW_0=\{w_0\}$,$CW=\{w_n\}$,CW_i 为当前活动,$t_i=fired(CW_i)$,$i\geq 0$；$CW_i=\{(CW_{i-1}-'t_{i-1})\cup t'_{i-1}\}$,$i\geq 1$。

3 基路径方法

向量空间的基是相互独立的一组线性向量,基"覆盖"整个向量空间,使得该空间中的任何其他向量都可以用基向量来表示。因此,一组基向量在一定程度上可表示整个向量空间的本质:空间中的一切都可以用基表示,并且如果一个基元素被删除了,则这种覆盖特性也会丢失。基对测试的潜在意义是:如果可以把程序看做是一种向量空间,则这种空间的基就是需要测试的关键元素的集合:如果基没有问题,则可以希望能够用基表达的一切都是没有问题的[6]。

1976 年,McCabe 提出了一种基路径的测试方法,他认为强连接图的圈数量就是图中线性独立环路数的数量[7]。强连接图中圈数量的公式为 V(G)=e−n+p,其中 e 表示边数,n 表示结点数,p 表示连接区域数。

采用基路径方法对 UML 活动图进行分析。UML 活动图中一定存在初始结点和终止结点,如果从终止结点到初始结点引一条有向边,则 UML 活动图成为一个强连接图,此时的圈数就是图中线性独立环路的数量。对于图中的每一条独立环路,若删除从终止结点引向初始结点的有向边,则线性独立环路成为从活动图中初始结点开始到终止结点结束的线性独立路径,即为基本路径,这时每一条基本路径表示的就是一个测试场景。

4 采用基路径方法生成测试场景

4.1 基于 UML 活动图生成测试场景的步骤

UML 活动图中通常包含一系列的并发子过程。采用基路径方法生成测试场景时,为了不忽略这些活动的并发特性,需要对活动图进行加工。

UML 活动图中的各种并发总是从并发分支结点开始,结束于并发汇聚结点,满足单入口单出口的性质。因此,可以将包括这些并发活动的部分压缩为一个结点。在生成测试场景时,先分析压缩后的活动图基路径以生成初步的测试场景,然后将被压缩的并发活动实例化,再还原到测试场景中,这样就可以得到完整的测试场景。

基于 UML 活动图生成测试场景的步骤描述如下：

Step 1 人工检测 UML 活动图的正确性、完整性；

Step 2 将活动图中的并发活动压缩为一个活动结点；

Step 3 采用基路径寻找算法,找出压缩后的活动图所包含的基本路径；

Step 4 将 Step 2 中被压缩的并发活动实例化,并替换 Step 3 找出的基本路径,形成完整的路径；

Step 5 根据 Step 4 产生的完整路径,生成相应的测试场景。

4.2 将活动图中的并发活动压缩为一个活动结点

某数字电视系统中业务处理活动图如图 1 所示,它描述了数字电视基本业务的工作流程,包含了一些基本的活动图元素。

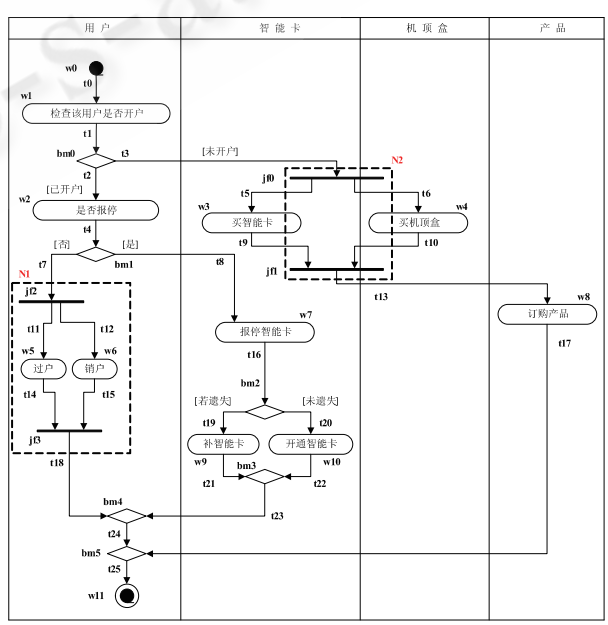

图 1　某数字电视系统中业务处理活动图

©《计算机系统应用》编辑部 http://www.c-s-a.org.cn

根据定义 1,活动图模型为 D = (W,JF,BM,T,F,C),其中活动集合为 W = {w_0,w_1,…,w_{11}};转移(边)集合为 T = {t_n,t_1,…,t_{25}};条件分支汇聚结点集合为 BM = {bm_0,bm_1,bm_2,bm_3,bm_4,bm_5};并发分支汇聚结点集合为 JF = {jf_0,jf_1,jf_2,jf_3};条件集合为 C = {c_0,c_1,c_2,c_3,c_4,c_5},其中 c_0 表示[未开户],c_1 表示[已开户],c_2 表示[否],c_3 表示[是],c_4 表示[若遗失],c_5 表示[未遗失]。

图 1 中所有结点的集合为,将图中并发活动 w5、w6 压缩为一个结点 N_1,并发活动 w_3、w_4 压缩为一个结点 N_2,则压缩后的活动图模块为 D = (W′,JF′,BM′,$T'_{W\cup BM}$,F′,C′),结点集合为,转移(边)集合为 T′ = {t_n,t_1,…,t_{25}};活动集合为 W′ = {w_0,…,w_2,w_7,…w_{11},N_1,N_2},BM′ = {bm_0,…,bm_5},JF′为空集,C′ = {c_0,c_1,c_2,c_3,c_4,c_5}。

4.3 采用基路径方法找出基本路径

采用基路径方法找出基本路径算法的基本思想是:首先选择一条尽可能多的经过活动图条件分支结点的基线路径 ep_0,ep_0 起始于初始活动 w_0,终止于结束活动 w_0;然后,回溯该路径,依次在每个分支结点处作一次翻转,即取另一条转移边 t′,按照深度优先方法遍历活动图至终止结点 w_n,产生一条新的路径 ep_1,重复上述步骤,直至所有分支结点都翻转一次,算法结束。具体算法描述如下:

算法 Generate_Basic_Path

输入 :the selected path ep_0 ;(基线路径 ep_0,始于 w_0,止于 w_n)

输出 :basic paths set Paths ;(活动图中的基本路径集 Paths)

Algorithm Begin

For each bm node in ep_0//将 ep_0 中的每一个条件分支结点 bm 依次入结点队列 node_queue;

Enqueue(node_queue,bm)

End For

Enqueue(path_queue, ep_0)//将基线路径 ep_0 加入路径队列 path_queue;

Put ep_0 in set Paths

//分别取出队列 node_queue 和 path_queue 的队头元素;

bm = Dequeue(node_queue)

ep = Dequeue(path_queue)

Do begin//当两队列 node_queue 和 path_queue 都不为空时,开始查找基本路径;

If fetch t ∈ ep in ep and t ∈ fired(bm) then

get t′ from set {fired(bm) – t}

End if//取出结点 bm 的另一条转移边 t′

ep′ = DFS_traverse(D′,ep,bm,t′)//从分支结点 bm′,以转移边 t′,按照深度优先的算法遍历活动图 D′,得到一条新的路径 ep′;

Put ep′ in set Paths

For each bm′ node in ep′//依次查找新路径 ep′ 的每个分支结点 bm′,若 bm′≠bm 且也不在结点队列 node_queue 中,则将结点 bm和 ep都入队;

If exist bm′ ∈ ep′ and bm′ not in node_queue and

bm′≠bm then

Enqueue(node_queue, bm′)//将结点 bm′ 和 ep都分别入队列;

Enqueue(path_queue, ep′)

End if

End for

bm = Dequeue(node_queue)

//本次翻转结束后,从结点队列中再次取出新的分支结点 bm;

If bm not in ep then

ep = Dequeue(path_queue)//若 bm 不在当前的路径 ep 上,则从路径队列再次取新的路径 ep

Until bm is null and ep is null

return Paths//返回基本路径集 Paths;

Algorithm End

对图 1 所示的活动图,执行上述算法,可以得到 4 条基本路径:ep_0、ep_1、ep_2、ep_3,如表 1 所示。依据强连接图中圈数公式 V(G) = e – n + p,若从 w_{11} 引一条边到 w_0,则活动图 D成为强连接图。此时 e 的值为 19,n 的值为 16,p 的值为 1,则 V(G)的值为 4,即为上述算法所找出的基本路径数目。

4.4 并发活动的实例化

如果并发流程中的活动可以按任意的顺序排列,对于包含多个并发活动的流程,可能出现活动组合数量爆炸的问题。假设有 n 个进程并发,每个进程包含的活动数目为 mn,则生成的基本路径数为 $\left(\sum_{i=1}^{n} m_i\right)!\Big/\left(\prod_{i=1}^{n} m_i!\right)$。本文从以下四个方面对并发活动加

©《计算机系统应用》编辑部 http://www.c-s-a.org.cn

以约束[7]：

(1) 测试人员定制约束条件：如活动 wi 必须在活动 wj 之前发生，表示为 $w_i > w_j$；

(2) 对活动图的动作分类，按照活动的重要性选择组合时的顺序；

(3) 制定活动的权值，在生成测试场景时，权值大的活动优先选择；

(4) 测试人员制定生成测试场景的数量。

对于上述活动图，其并发活动部分 N_1 和 N_2 分别可以产生 2 条路径，即 N_1 可以产生路径 w_5w_6、w_6w_5，N_2 可以产生路径 w_9w_{10}、$w_{10}w_9$。

表 1　活动图图 1 中基本路径的查找过程

序号	翻转点	路径	结点队列	路径队列	当前结点	当前路径
1		ep_0：$w_0t_0w_1t_1bm_0t_2w_2t_4bm_1t_7N_1t_{18}bm_4t_{24}bm_5t_{25}w_{11}$	bm_0，bm_1	ep_0		
2	bm_0	ep_1：$w_0t_0w_1t_1bm_0t_3N_2t_{13}w_8t_{17}bm_5t_{25}w_{11}$	bm_1		bm_0	ep_0
3	bm_1	ep_2：$w_0t_0w_1t_1bm_0t_2w_2t_4m_4bm_1t_8w_7t_{16}bm_2t_{19}w_9t_{21}bm_3t_{23}bm_4t_{24}bm_5t_{25}w_{11}$	bm_2	ep_2	bm_1	ep_0
4	bm_2	ep_3：$w_0t_0w_1t_1bm_0t_2w_2t_4bm_1t_8w_7t_{16}bm_2t_{20}w_{10}t_{22}bm_3t_{23}bm_4t_{24}bm_5t_{25}w_{11}$			bm_2	ep_2

将路径 w5w6、w6w5 和 $w9w_{10}$、w_10w9 分别替换 N_1 和 N_2，还原上述四条基本路径 ep_0、ep_1、ep_2、ep_3，可以得到完整的 6 个测试场景，如表 2 所示。

5　结束语

本文对 UML 活动图的语法和语义进行了形式化定义和描述，介绍了 McCabe 的基路径方法，结合实例，详细介绍了采用基路径方法从 UML 活动图中提取测试场景的方法，该方法产生的基本路径相对独立，已应用于多个实际项目的测试工作，提高了测试效率，具有良好的实用性。

表 2　某数字电视系统中业务处理活动图的测试场景

场景编号	期望的活动序列
ts_1	w_0 t_0 w_1 t_1 [已开户] t_2 w_2 t_4 [不报停] t_7 (w_5 w_6) t_{18} t_{24} t_{25} w_{11}
ts_2	w_0 t_0 w_1 t_1 [已开户] t_2 w_2 t_4 [不报停] t_7 (w_6 w_5) t_{18} t_{24} t_{25} w_{11}
ts_3	w_0 t_0 w_1 t_1 [未开户] t_3 (w_9w_{10}) t_{13} w_8 t_{17} t_{25} w_{11}
ts_4	w_0 t_0 w_1 t_1 [未开户] t_3 (w_10w_9) t_{13} w_8 t_{17} t_{25} w_{11}
ts_5	w_0 t_0 w_1 t_1 [已开户] t_2 w_2 t_4 [已报停] t_8 w_7 t_{16} [若遗失] t_{19} w_9 t_{21} t_{23} t_{24} t_{25} w_{11}
ts_6	w_0 t_0 w_1 t_1 [已开户] t_2 w_2 t4 [已报停] t_8 w_7 t_{16} [未丢失] t_{20} w_{10} t_{22} t_{23} t_{24} t_{25} w_{11}

参考文献

1　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. 北京航空航天大学学报，2001，8(4)：433 – 437.

2　刘超. 程序交互执行流程图及其测试覆盖规则. 软件学报，1998，9(6)：458 – 463.

3　刘敏，金茂忠，刘超. 基于 UML 活动图模型生成测试场景的设计. 计算机工程与应用，2002，12(38)：122 – 124.

4　Wanglinzhang，Yuan J iesong，Yu xiaofeng，Hu jun，Li Xuandong，Zheng Guoliang. Generating Test Cases from UML Activity Diagrams Based on Gray – Box Method. In：11th Asia – Pacific Software Engineering Conference (APSECp04). Washington：IEEE Computer Socrety，2004. 284 – 291.

5　牟凯，顾明. 基于 UML 活动图的测试用例自动生成方法研究. 计算机应用，2006，4(4)：844 – 846.

6　Paul C. Jorgensen. Software Testing：A Craftsman's approach，CRC Press，INC. 2003，(7)：131 – 134.

7　McCabe，Tom. A Software Complexity Measure. IEEE Trans. Software Eng.，1976，2(6)，308 – 320.

©《计算机系统应用》编辑部 http://www.c-s-a.org.cn