文章编号：2096-1472(2016)-08-19-04

一种基于UML活动图的测试场景自动生成策略

曹 阳，刘正涛
（三江学院计算机科学与工程学院，江苏 南京 210012）

摘 要：传统的场景法在设计测试用例的过程中存在着构造场景困难、冗余度高、设计效率低下等问题。针对此问题，提出了一种基于UML活动图的测试场景自动生成策略。在建立活动流图模型后，采用改进的深度优先搜索算法获得路径集合，应用路径优化算法生成测试路径及测试场景。通过在商用的供应商协同平台的测试过程中应用该策略，验证了其有效性。实践结果表明，该策略较好的解决了循环工作流产生的路径爆炸问题，降低了测试场景的冗余度。

关键词：测试场景；活动流图；深度优先搜索；独立路径；自动生成
中图分类号：TP312 **文献标识码**：A

A Scheme for Test Scene Automatic Generation Based on UML Activity Diagram

CAO Yang,LIU Zhengtao
(*College of Computer Science and Engineering,Sanjiang University,Nanjing* 210012,*China*)

Abstract: In the process of designing test case through the traditional scene method,there are many problems,including scene construction difficulty,high redundancy and low design efficiency.To solve this problem,the paper proposes a scheme for test scene automatic generation based on UML activity diagram.On the basis of the activity flow graph,the improved depth-first search algorithm is adopted to obtain path collection,and the path optimization algorithm method is applied to generate test path and test scene.The effectiveness of this scheme has already been verified in a testing process of a commercial supplier collaboration platform.The practice results indicate that the scheme can effectively solve the problem of path explosion caused by cycle workflow and significantly reduce the redundancy of test scene.

Keywords:test scene;activity flow graph;depth-first search;independent path;automatic generation

1 引言(Introduction)

基于场景的测试用例设计方法[1]是一种重要的黑盒测试技术，其核心思想是通过分析软件需求，构建各种测试场景，并寻找测试场景与系统输入参数、特征状态的关联关系进行测试用例的设计。测试场景的构建是场景法的关键环节，传统的从软件需求规格说明中提取基本流、备选流进行测试场景构建的方法存在效率低下、执行困难等问题。因此，探索测试场景自动生成策略成为运用场景法设计测试用例的重要研究点之一。基于UML模型驱动测试用例自动生成是一种基于模型的软件测试技术，在自动生成测试用例方面有广泛的研究[2,3]。其中UML活动图用于表示系统业务的工作流程，被认为是最适合描述软件过程的模型，因此众多研究者在使用UML活动图生成测试场景方面做了一定的研究。周飞等提出将活动图转化为有向图，通过构建图的搜索树生成测试场景[4]；苏翠翠等提出了一种基于路径覆盖的测试场景生成算法[5]；Jena等提出了活动流图(Activity Flow Graph，AFG)的概念，设计了测试场景模型，并采用遗传算法生成测试场景[6]。这些研究为基于UML活动图构建测试场景提供了良好的思路，但是存在着以下问题：一是构建的模型在形式化定义上有所欠缺；二是未能实现测试场景的自动化生成；三是未考虑复杂测试场景中的循环工作流的执行与优化。

本文通过对活动流图进行形式化定义，给出了测试场景的自动生成策略，针对系统需求中存在循环工作流的情况提出了一种测试场景优化算法，显著降低了测试场景的冗余性，提高了测试设计效率。

2 基于活动流图的测试场景生成策略(Test scene generation based on activity flow graph)

2.1 活动流图的元素定义

由于UML是一种半形式化的建模语言，因此需要采用一种更好的可形式化表示的图来构造测试场景模型。活动流图由活动图转化而来，通过把活动图中的各种元素按照一定的规则映射而成[7]，其本质是一个有向图。图1表示了一个活动图与活动流图的映射关系。

基金项目：江苏省电力公司科技项目(No.J2014057)；横向课题“供应商管理平台与车间作业现场管理平台”(KH150025)；江苏省教育科学“十二五”规划2015年度高教重点课题(B-b/2015/01/084)；江苏省现代教育技术研究重点课题(2015-R-42743).

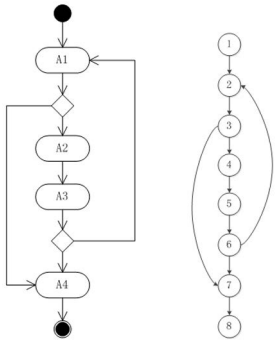

图1 活动图与活动流图

Fig.1 Activity diagram and activity flow graph

定义1：活动流图用一个四元组$AFG(V,E)=\{V(AFG), E(AFG), v_s, VE\}$表示。其中$V(AFG)=\{v_1,v_2,\cdots,v_N\}$表示图中所有结点的集合，N为活动流图中所有结点的数量；$E(AFG)=\{e_1,e_2,\cdots,e_M\}$表示图中所有边的集合，M为活动流图中所有边的数量；v_s为活动流图中唯一的起始结点；$VE=\{ve_1,ve_2,\cdots,ve_n\}$表示活动流图中终结结点的集合，n为所有终结结点的数量。边可以由一个有序结点对表示，即$\forall e\in E(AFG),\exists v_i,v_j\rightarrow e=(v_i,v_j)$。

定义2：路径集合是活动流图中所有从起始结点到终结结点由边连接而成的结点序列，记为$P(AFG)=\{p_1,p_2,\cdots,p_{np}\}$，np为活动流图中路径的总数。路径可以由组成该路径的边的有序集合表示，记为$\forall p\in P(AFG),p=\{e_1,e_2,\cdots,e_{ne}\}$，ne为路径p包含的边数。

根据活动图与活动流图的映射关系，一条路径表示系统用例从开始到结束的执行流程，对应着一个测试场景，路径集合则对应测试场景集合，构建测试场景的问题就转化为获取活动流图路径集合的问题。对于较复杂的系统需求，所构建的活动图往往包含循环工作流；循环会导致路径的组合爆炸，对活动图中所有可能的路径进行穷尽测试是无法达到的[8]。为了得到所有路径集合，需要对循环工作流进行合理处理。循环可展开成为无限长的路径序列，为了控制路径序列的长度，限定测试场景中每个循环只展开一次[9]。同时，路径集合中的路径之间存在大量相同的边，使其对应的测试场景之间存在较多相同的测试工作流，从而导致设计出的测试用例存在冗余，因此需要对路径集合进行优化，降低冗余度。

定义3：循环回路是某路径中含有一组边，且由这组边形成的一个闭合回路，即$\exists e_i,e_{i+1},\cdots,e_j\in p\rightarrow e_i=(v_i,v_{i+1})\wedge e_j=(v_{j-1},v_i)$；循环回路集合表示为C(AFG)。

定义4：独立路径是至少存在一条路径集合中其余路径不包含的边，且不含循环回路或包含的各个循环回路至多执行一次的路径。独立路径表示为：$\exists p\in P(AFG)\rightarrow(\exists e\in p\rightarrow e\notin(P(AFG)-p)\wedge(c\in C(AFG)\wedge\forall c\in p\rightarrow count(c)\leq 1))$。

活动流图中所有的独立路径组成的集合称为独立路径集合，记为$Indep_P(AFG)$。独立路径集合为路径集合的一个子集，即$Indep_P(AFG)\subseteq P(AFG)$。

2.2 测试场景生成策略的设计

根据独立路径的定义，独立路径集合能够覆盖活动流图中所有的边，且保证每条路径中循环回路至多执行一次；对应的测试场景集合即为优化的测试场景集合，实现了采用较少的测试场景覆盖全部的工作流，同时解决了循环回路的执行问题。基于活动流图的测试场景生成策略如下：

(1)根据系统需求创建活动图。

(2)将活动图转化为活动流图。

(3)基于改进的深度优先搜索算法获取活动流图的路径集合。

(4)采用路径优化算法获取活动流图的独立路径集合。

(5)根据独立路径集合生成测试场景。

3 关键算法的设计与实现(Design and implementation of the key algorithm)

3.1 基于深度优先策略的路径搜索算法

深度优先搜索算法(Depth First Search，DFS)是一种经典的图遍历算法，可以用于获取图的路径集合。在基于UML活动图生成测试场景的研究中，多为面向有向无环图的获取路径集合的DFS算法[10,11]。但有向无环图不包含循环回路，因此针对本文的研究内容，提出了一种改进的深度优先搜索算法，以解决循环回路的问题，具体思想如下：

(1)起始结点v_s设置为已访问，将其入栈。

(2)获取栈顶元素v的相邻结点集合W。

(3)对集合W进行遍历，对于未入栈且未被访问的相邻结点$w\in W$，将其入栈，并标记为已访问，同时将w作为栈顶元素，重新执行第(2)步。

(4)对于已入栈且已访问的相邻结点$w\in W$，说明存在循环回路，判断w在栈中出现的次数是否小于该结点的出度；如果出现次数小于出度，则说明当前循环回路为第一次执行，将w入栈，同时将w作为栈顶元素，重新执行第(2)步；否则，则访问下一相邻结点。

(5)如果W集合中不存在步骤(3)(4)中的两类相邻结点，则将W集合中的每个结点标记为未访问，将栈顶元素v出栈。

(6)当栈顶元素为终结结点，即W为空时，将栈中的结点按照逆序排列构成的路径加入路径集合P(AFG)中，并弹出栈顶元素。

改进的深度优先搜索算法的具体实现如下：

```
v_s.visited = true;
v = v_s;
void dfs(Graph g,Node v){
push v into Stack;
W={w|adjacent node of v};
if W is not empty
then
foreach w in W
if w not in Stack and w.visited==false
```

```
                then
                w.visited=true;
                v=w;
                push v into Stack;
                dfs(g,v);      //以当前结点为起始结点，递归调用算法
                else if w in Stack and w.visited==true
                                then
    if occurrence number of w is less than out-degree of w;
        //判断循环回路是否是第二次执行
                then
    v=w;
    push v into Stack;
    dfs(g,v);
        else
    continue;
        end
        else
        pop v out of stack;
    w.visited=false;
        end
        end
        else
    add the reverse sequence of node in Stack to P(AFG);
    //当前结点为终结节点，将栈中结点逆序存入路径集合
        pop v out of stack;
        end
        }
```

3.2 独立路径集合获取算法

在路径集合中根据独立路径的定义进行路径筛选，可以得到独立路径集合。独立路径集合的获取算法思想如下：

(1)任选一条路径作为独立路径，初始化独立路径集合。

(2)对剩余路径进行遍历，逐条验证其是否为独立路径，将独立路径加入独立路径集合。

(3)对第(1)步中得到的独立路径集合中的每条路径再次进行遍历，移除不满足独立路径条件的路径。

独立路径集合的获取算法具体实现如下：

```
void acquireIndepPath(Collection P(AFG))
    p0=radom(P(AFG));
    put p0 into Indep_P(AFG);
    foreach p in P(AFG)-Indep_P(AFG)  //对剩余路径进行遍历
    if p is independent path  //判断是否满足独立路径的条件
    then
        add p into Indep_P(AFG);
    else
        fetch next path in P(AFG);
    end
    end
    foreach p in Indep_P(AFG)  //对独立路径集合进行二次筛选
    if p is not meet the condition of independent path
    then
remove p from Indep_P(AFG);
    end
    end
```

4 应用实例(Examples of application)

供应商协同平台是一个将企业采购管理系统与供应链管理系统对接形成线上流程的一体化平台，为企业拉动与上游客户之间在订货协作、商品推介、库存查看、资金支付、物流查询、渠道沟通等业务环节的紧密协作。平台中包含众多业务流程，其中“采购物资”流程是一个较为复杂的系统需求，包含若干审批环节。以该流程为测试对象验证本文提出的测试场景自动生成策略的有效性。根据系统需求，构建“采购物资”流程的活动图如图2所示。

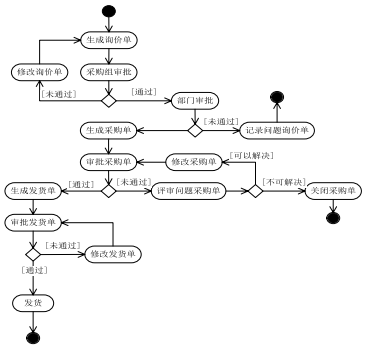

图2 采购物资的活动图

Fig.2 Activity diagram of purchasing goods

将活动图中的各元素映射为活动流图中的结点与边，得到活动流图如图3所示。

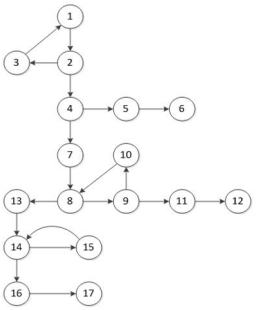

图3 采购物资的活动流图

Fig.3 Activity flow graph of purchasing goods

根据系统需求定义，$v_s=1$，$VE=\{6,12,17\}$，应用改进的深度优先搜索算法，得到对应的路径集合如表1所示。

表1 路径集合

Tab.1 Path collection

编号	路径构成
P1	1−2−4−5−6
P2	1−2−3−1−2−4−5−6
P3	1−2−4−7−8−9−11−12
P4	1−2−3−1−2−4−7−8−9−11−12
P5	1−2−4−7−8−9−10−8−9−11−12
P6	1−2−3−1−2−4−7−8−9−10−8−9−11−12
P7	1−2−4−7−8−13−14−16−17
P8	1−2−3−1−2−4−7−8−13−14−16−17
P9	1−2−4−7−8−13−14−15−14−16−17
P10	1−2−3−1−2−4−7−8−13−14−15−14−16−17
P11	1−2−4−7−8−9−10−8−13−14−16−17
P12	1−2−3−1−2−4−7−8−9−10−8−13−14−16−17
P13	1−2−4−7−8−9−10−8−13−14−15−14−16−17
P14	1−2−3−1−2−4−7−8−9−10−8−13−14−15−14−16−17

根据独立路径集合的获取算法，得到独立路径集合如表2所示。将独立路径表示的结点序列映射至图1中，生成的测试场景如下：

测试场景1：Start—生成询价单—采购组审批未通过—修改询价单—生成询价单—采购组审批通过—部门审批未通过—记录问题询价单—End。

测试场景2：Start—生成询价单—采购组审批通过—部门审批通过—生成采购单—审批采购单未通过—评审问题采购单(可以解决)—修改采购单—审批采购单未通过—评审问题采购单(不可解决)—关闭采购单—End。

测试场景3：Start—生成询价单—采购组审批通过—部门审批通过—生成采购单—审批采购单通过—生成发货单—审批发货单未通过—修改发货单—审批发货单通过—发货—End。

表2 独立路径集合

Tab.2 Independent path collection

编号	路径构成
P1	1−2−3−1−2−4−5−6
P2	1−2−4−7−8−9−10−8−9−11−12
P3	1−2−4−7−8−13−14−15−14−16−17

通过对实例的应用结果进行分析，提出的测试场景生成策略解决了以下问题：

(1)实现了根据活动流图自动化获取路径集合。

(2)解决了循环回路带来的复杂性问题；图2中包含的三个循环回路在路径中仅执行一次。

(3)通过路径优化，将14个测试场景优化为三个测试场景，降低了测试冗余性。

(4)优化后的测试场景能够使系统工作流达到100%的测试覆盖率，具有较好的测试充分性。

5 结论(Conclusion)

本文提出了一种测试场景的自动生成策略，根据UML活动图，给出了活动流图的严格定义，通过改进的深度优先搜索算法及路径优化算法，解决了测试场景中的循环回路问题与测试冗余问题。同时该策略能够实现测试场景的自动生成，并在一个商业系统测试过程中成功应用，证明了测试结果具有较好的充分性，测试效率得到了显著提高。

参考文献(References)

[1] Anand S,et al.An Orchestrated Survey of Methodologies for Automated Software Test Case Generation[J].Journal of Systems and Software,2013,86(8):1978−2001.

[2] Utting M,Pretschner A,Legeard B.A Taxonomy of Model−based Testing Approaches[J].Software Testing Verification and Reliability,2012,22(5):297−312.

[3] Shirole M,Kumar R.UML Behavioral Model Based Test Case Generation:a Survey[J].ACM SIGSOFT Software Engineering Notes,2013,38(4):1−13.

[4] 周飞,杨根兴,蔡立志.基于UML的测试用例生成方法研究[J].计算机应用与软件,2009,26(2):107−110.

[5] 苏翠翠,王晓军.基于UML活动图的测试用例生成方法研究[J].计算机技术与发展,2010,20(8):49−51.

[6] Jena A K,Swain S K,Mohapatra D P.A Novel Approach for Test Case Generation from UML Activity Diagram[C].Issues and Challenges in Intelligent Computing Techniques (ICICT),2014 International Conference on.IEEE,2014:621−629.

[7] Kundu D,Samanta D.A Novel Approach to Generate Test Cases from UML Activity Diagrams[J].Journal of Object Technology,2009,8(3):65−83.

[8] 杨鹤标,李云平.基于UML活动图的功能测试场景生成方法[J].计算机工程,2011,37(2):55−57.

[9] 陈鑫,等.一种面向列车控制系统中安全攸关场景的测试用例自动生成方法[J].软件学报,2015,26(2):269−278.

[10] Sharma C,Sabharwal S,Sibal R.A Survey on Software Testing Techniques Using Genetic Algorithm[J].International Journal of Computer Science Issues,2014,10(1):381−393.

[11] 李浩,陈锋.基于遗传算法的UML活动图测试用例优化研究[J].现代电子技术,2015,38(19):117−120.

作者简介：

曹 阳(1982−)，男，硕士，讲师.研究领域：软件测试与验证，数据挖掘.

刘正涛(1975−)，男，博士，副教授.研究领域：数据库应用.