基于 UML 活动图生成系统测试场景的方法①

王志坚 金 春 (河海大学 计算机及信息工程学院 江苏 南京 210098)

摘 要： 旨在研究运用统一建模语言(UML)活动图生成测试场景的方法。首先对 UML 活动图进行了形式化定义，确定了一系列覆盖准则。然后，根据覆盖准则制定活动图的一般处理思路，针对活动图中并发结构提出采用信号量和嵌套分割的方法进行处理。该方法有效控制了测试场景集的数量，为 UML 活动图的自动化测试提供了系统的、有效的、可行的方法。

关键词： 活动图；测试场景；统一建模语言；覆盖准则；完全并发块

Design of Testing Scenario Generation Based on UML Activity Diagram

WANG Zhi-Jian, JIN Chun

(College of Computer & Information Engineering, Hehai University, Nanjing 210098, China)

Abstract: This paper mainly studies how to generate test scenarios from UML activity diagram. Firstly, the activity diagram is defined formally, and coverage rules are set down. Then, based on the general thought of dealing with activity diagram with UML activity diagram, a novel method is proposed by adopting semaphore and nested partition to process the concurrent structure of activity diagram for some special elements of activity diagram. It reduces the number of test scenario effectively and also makes automatic generating test case from UML activity diagram become systemic, effective and feasible.

Keywords: activity diagram; test scenario; unified modeling language; coverage rules; fully concurrent block

统一建模语言(Unified Modeling Language，UML)是一个通用的可视化建模语言，用于对软件进行描述、可视化处理、构造和建立软件系统制品的文档。UML 模型在软件测试方面有着很大的应用潜力，其最大优点就是它比其他形式化方法具有更为广泛的适用性。在软件开发的各个阶段，通过将系统的分析、设计和实现等工件转化为 UML 的规格说明，并把它们作为测试需求的直接来源，驱动测试的整个过程。

1 引言

UML 活动图是 UML 中描述系统动态行为的工具之一，它展现了活动可能发生的序列，专注于系统(操作、对象)的动态视图[1]。它对系统(操作、对象)的功能建模特别重要，并强调对象间的控制流程。模型借用了流程图、状态转换图、工业工程工作流程图和 Petri 网的思想。基于活动图的系统测试方法是进行软件测试的重要途径之一[2]。已有的测试用例自动生成方法很多，文献[3]提出了基于关系代数形式化说明的自动生成测试用例的方法；文献[4]提出了基于布尔形式化说明的测试数据自动生成方法；文献[5]结合等价划分等多种方法及基于协议的形式化语言(Protocol Specification Language, PSL)产生了工具 ConData，用于生成测试用例。

测试场景是与待测试软件的执行相对应的一个活动场景，由一系列活动按照一定的顺序组成，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，测试用例包括测试场景和测试数据，因此生成测试场景是生成测试用例的前提，从活动图中提取测试场景对由 UML 活动图生成测试用例有重要的意义。

本文在文献[6]的基础上研究运用 UML 活动图模型生成测试场景的方法，改进算法之后提高了测试场

① 收稿时间:2009-08-05;收到修改稿时间:2009-09-08

景生成的完整性，提高了测试效率。这种方法不仅适合于软件的系统测试，同时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。

2 活动图的形式化定义

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。为了从活动图中自动生成测试场景，首先需要通过形式化的方法定义活动图。

运用常规的集合概念，活动图可表示为

$D=(A, O,T, C,F,a_I,a_F)$

其中 $A=\{a_1,a_2,a_3,\cdots\}$表示活动状态有限集，称为活动图的结点集合；

$O=\{o_1,o_2,o_3,\cdots\}$表示对象有限集；

$T=\{t_1,t_2,t_3,\cdots\}$表示转换有限集，称为边集合；

$F\in(A\times T\times C\rightarrow A)\cup(A\times T\times C\rightarrow O)\cup(O\times T\times C\rightarrow A)$表示活动状态或对象与转化之间的流关系。

C 为相应转换上的控制条件；

$a_I\in A$ 表示初始活动状态,$a_F\in A$ 表示终止活动状态；

活动状态集 A 中的活动状态有多种类型:初始活动(Initial)aI 和终点活动(Final)aF、基本活动(Activity)和组合活动(Component Activity)，条件分支(Branch)及其汇聚结点(merge)。此外，它还包含了并发分支(Fork)和并发汇聚结点(Join)，信号发送(Signal Sender)和信号接受活动(Signal Receiver)，以及对象结点(Object)。组合活动就是一个嵌套子活动的活动。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。活动图中的迁移边也有多种类型，流关系 F 分为控制流关系(Control Flow)和对象流关系(Object Flow)、信号流关系(Signal Flow)。

3 覆盖准则

软件测试的充分性直接依赖于测试用例集合的质量—有效性和覆盖率。白盒测试中的覆盖准则有语句覆盖、分支覆盖、条件覆盖等，是为保证测试的充分性而制定的。黑盒测试是相对功能点来说的，同样需要确定一定的覆盖准则来支持充分测试。这里的覆盖准则与白盒测试类似，因为采用的 UML 活动图与程序流程图从顺序关系上看非常相似。

提出功能测试的 5 个基本覆盖测试准则：

准则 1(结点覆盖准则) :图上的所有控制点至少要执行过 1 次。

准则 2(转换覆盖准则) :图上的所有转换边至少要执行过 1 次。

准则 3(逻辑路径覆盖准则) :图上的所有路径至少要执行过 1 次。在运用此项测试准则时,必须对图中的循环作一定的限定，比如限定只考核两种情况,即不进入循环体和进入循环体 1 次或 1 次以上。经此简化后的路径称作逻辑路径或基本路径。

准则 4(代表值覆盖准则):对于每一个控制点以及控制点上的每一个控制量,依据一定的测试策略选定一个有限的输入值集合,称作代表值集合,则每个控制点的代表值集合中的每一个值至少要被测试过 1 次。

准则 5(信号流覆盖准则)：

生成测试场景时候至少包含三种可能的场景。

(1) 信号流发生场景，即发送活动与接收活动同时出现，发送信号达到发送信号的各种条件，且接收活动接收到了信号并做出相应的反应。

(2) 发送方不发送信号的场景，发送方未达到发送信号条件或者触发了其他活动而没有发送信号。

(3) 发送方发送信号而接收方接收不到场景，例如网路中断是最常见的原因。这时我们关心发送方与接受方的后续操作，特别是同步信号流的发送方是一直等待，还是超时重发；异步信号流的接受方没有收到信号会有什么不同的操作流程，这些是设计人员最容易忽略的，但确是测试者最应该测试的，因此这种场景最为重要。

满足五个覆盖准则生成的测试场景对系统的测试具有重要意义。

4 信号流处理

信号一般是一个反复等待、查询的过程，信号的发送与接收并没有必然的流程关系。

因此信号流处理方法是：信号发送活动与信号接收活动间的前后关系放进约束条件池，并记录此发送接收对，以便生成包含信号流的场景。

5 生成测试场景算法

介绍测试场景生成算法之前先介绍几个概念：

完全的并发块：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的并发块称为完全并发块。

嵌套分割：针对嵌套的完全并发块要分割处理，即是先处理内部嵌套完全并发块活动序列生成。然后整个内部完全并发块活动序列当作一个活动点，再处理外部完全并发块。

针对完全并发块要使用信号量的方法进行生成测试序列。既是使用排除法，首先把并发块中的结点(活动结点和分叉和聚合结点)进行全排列，然后根据约束条件使用信号量方法来进行过滤掉不合理的测试场景。在生成测试场景时候，先分析压缩后的活动图生成初步的测试场景，然后将被压缩的并发活动依据并发块处理算法进行实例化，再还原到测试场景中，这样就可以得到完整的测试场景。

UML 活动图生成测试场景的算法(ATOS)Rational Rose 扩展接口(REI)从 Rational Rose 的规约文件中提取 UML 活动图信息。下面给出针对测试覆盖准则从活动图转化到测试场景的算法(Activity diagram To Scenario)。

算法描述如下：

(1)人工检测 UML 活动图的正确性，完整性；

(2) 将活动图中的并发活动压缩为一个活动结点；

① 新建一个活动为空的测试场景。

② 将初始活动(Initial)结点 a_1 加入到测试场景。

③ 沿控制流 F 开始进行遍历。

```
if (next 结点 is Activity (ai))
{
  if(TScenario 包含 ai)
    {
     TScenario 构造结束;
       Return;
    }
else
  {
    将 ti，ci，next ai 加入 TScenario
  }
}
If(next 结点 is Branch)
{
```

将 TScenario 复制成 N-1 个新的 TScenario，TScenario 走其中一条分支，将 t(i+1) ，c(i+1) ，a(i+1) 加入 TScenario。

```
}
If(next 结点 is Fork)
{
If(TScenario 包含 Fork(ai))
 {
   TScenario 构造结束;
     Return;
  }
 Else
 {
     将 ti，ci，下一个 ai 加入 Tscenario，继续沿着控制流遍历直到找到并发汇聚结点 aj,将 aj 加入 TScenario;
     goto ③;
  }
};
```

④ 将未构造结束的测试场景，按照③继续进行直到所有场景结束；

⑤ 将所有场景执行，检查是否存在有对象流的结点。如果存在则找到对象后一活动结点 a_i，将对象作为 ai 的输入转换条件保存到转换中。因为活动图中的对象是前一个活动的输出，后一个活动的输入，对象到达某个状态后，必须由下一个活动接收进行下一步的处理；

⑥ 采用信号量方法解决并发块的实例化问题，首先将并发块中的分支活动结点分别组合为一个活动结点，简化活动图。然后将并发块中所有结点(活动结点和分叉、聚合结点)全排列，生成全部的测试场景(包括合理和不合理的)。针对每一个结点定义一个信号量，初始信号量是该活动结点的入迁移个数，每个结点在场景出现的时候必须满足的条件是，该活动点的信号量是 0，同时将该结点所有出迁移对应的活动结点的信号量分别减 1。并且用户可以自己输入活动图无法提取的约束条件，根据约束条件对活动场景进行过滤，过滤出去不合理的测试场景。将并发结点和聚合结点从活动序列中去除，最后把合并后的条件分支结点还原到测试场景中，生成整个并发块的测试场景；

⑦ 将并发块的合理测试场景再还原到整个活动图的测试场景中，从而可以得到完整的测试场景。

6　删除不必要场景

通过前边算法得到了多个场景。但是如果整个测试场景没有设计到系统，对于测试系统而言，这样的测试场景不需要，因此为了提高测试效率需要将其去掉。

删除不必要场景的算法描述如下:

选择一个测试场景遍历。建立测试对象数组 Object　B[]并初始化，设置布尔变量 flag=false

```
For(int n=0; n < TScenario.length();n ++)
{
For(int i=0;i< TScenario[n].length();i++)
{
  flag=false ;
  If(ai∈B[])
    {
      flag=true;
      return;
    }
}
  If(flag= =false)
  {
    delete TScenario[n];
  }
};
```

所有测试场景执行完毕，剩下的测试场景就是与系统测试相关的测试场景。

7　并发块处理方案

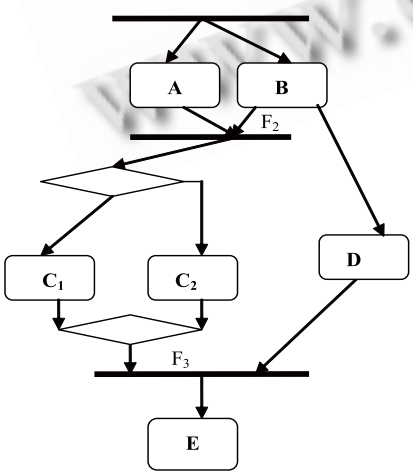

图 1　并发块

如图 1 所示是一个典型的活动图中的并发块，通过本文对并发块的处理算法对图 1 进行实验验证，通过验证充分证明了本文算法的充分性和合理性。

① 首先将分支结点 C1，C2 合并为 C 结点。

② 将完全并发块中所有结点进行全排列

ABCDF_1 F_2 F_3; ABDC F_1 F_3 F_2; BACD F_2 F_3 F_1; BADC F_3 F_2 F_1…等 5040 个全排列等价于 24 个 ABCD 全排的序列。

③ 初始化各个活动点的信号量，A 的信号量为 1，B 为 1，C 为 1，D 为 1，F_1 为 0，F_2 为 2，F_3 为 2。

④ 过滤不合理的活动序列

序列 F1AB F2C F3D，F_1 先执行，然后 A 的信号量减去 1 变为 0，B 的信号量减去 1 变为 0，然后 A(信号量为 0)行，F_2 的信号量减 1，变为 1，B 执行(信号量为 0)F_2 的信号量减 1 变为 0，D 的信号量减 1 变为 0，接着执行 F_2(2 信号量为 0)此时 C 的信号量减 1 为 0，此时 F_3 的信号量减 1 为 1，C 执行，F_3 的信号量减去 1 变为 0，F_3(3 信号量为 0)行，此时 D 的信号量为 0 所以合法。D 执行。活动序列 F_1AB F_2C F_3D 为合法的并发块测试序列。依此算法过滤每个序列。

⑤ 将过滤后的序列去除分叉和聚合结点生成合理测试序列为 ABCD，ABDC，BACD，BADC，BDAC。

⑥ 最后将 C 分别替换为 C_1, C_2 生成合理序列 AB C_1D， ABD C_1，BA C_1D，BAD C_1，BDA C_1，AB C_2D，ABD C_2，BA C_2D，BAD C_2，BDA C_2。

综上所述，最后并发块生成有效测试场景为 10 个，过滤掉无效测试场景 38 个，完全排除了不合理的测试场景，大大提高了测试场景生成的效率和有效性。

8　结语

通过对 UML 活动图的形式化描述以及对覆盖准则的定义，对测试场景生成进行研究，从理论上分析活动图的并发特征给测试场景集合带来的数量爆炸问题。在分析的同时引入信号量和嵌套分割处理并发块的方法，减少测试场景的具体思路和算法。试验证明，加了过滤的算法消除了多数无效的测试场景，从而有效控制了测试场景集的数量，为下一步自动化生成测试用例和进一步提高算法性能奠定了基础。

(下转第 200 页)

(上接第 188 页)

参考文献

1 徐宏喆,陈建明,等.UML 自动化测试技术.西安:西安交通大学出版社, 2006. 21－22.

2 梁义芝,王延章.UML 活动图的形式语义及分析.计算机工程与应用, 2003,39(18):28－31.

3 Tsaiwt, Volovik D, Tkeefe TF. Automated test case generation for programs specified by relational algebra queries. IEEE Transactions on Software Engineering, 1990,16(3):316－324.

4 Weyuker E, Goradia T, Singha.Automatically generating test case data from a Boolean specification. IEEE Transactions on Software Engineering, 1994, 20(4):353－363.

5 Martins E, Sabiao SB. ConData: A tool for automating specification based test case generation for communication system. IEEE Los A lam itos. 2000. 1060-3425.

6 姜树明,赵燕清,等.基于 UML 活动图的测试用例生成.山东科学, 2007,20(5):43－47.