一种基于源代码分析的程序变化影响路径集的生成方法

郭丹丹 姜 瑛
(云南省计算机技术应用重点实验室 昆明 650500) (昆明理工大学信息工程与自动化学院 昆明 650500)

摘 要 在软件生命周期的任意阶段,均可能因为各种原因而导致软件发生变化。当软件发生变化时,必须对其进行回归测试,检查这些变化是否影响了软件原有的正常功能。为了提高回归测试的效率并降低成本,需要尽可能准确地确定软件变化影响的内容。在单元测试中,基于源代码语句分析了程序的变化影响范围,得到了程序的变化集和影响集,提出了生成影响路径集的算法。实验结果表明,该方法能有效产生程序变化影响的路径集,提高了回归测试的效率。

关键词 回归测试,源代码分析,变化影响路径集
中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.12.036

Generation Method of Path Set Affected by Program Change Based on Source Code Analysis

GUO Dan-dan JIANG Ying
(Yunnan Key Lab of Computer Technology Application, Kunming 650500, China)
(Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650500, China)

Abstract Any stage of the software life cycle may change software due to various reasons. When the software changes, it must be checked whether the changes affect the normal functions of the original software by using regression testing. In order to improve the efficiency and reduce the cost of regression testing, it is necessary to determine the content affected by software change accurately. This paper proposed a method to analyze the change scope of the programs based on source code statements during the unit testing. Then the impact and change sets can be gained. The generation algorithm of the paths impact set was presented. The experimental results show that this method can effectively generate the program path set affected by program change. And the efficiency of regression testing is improved.

Keywords Regression testing, Source code analysis, Path set of change impact

1 引言

随着软件规模日益增大,软件的演化和变更越来越频繁。软件变更是指在软件生命周期内,由于各种客观或主观原因,需要对软件系统的相关内容进行修改[1]。为了判断软件修改是否会影响软件质量或损坏软件原有正常功能,就必须对软件进行回归测试。在确定回归测试的范围时,如何根据变化内容进行变化影响分析、缩小测试范围、提高回归测试效率并降低测试成本等问题成为当前的研究重点。

变化影响分析是在变化执行后,进行变化传播和波动影响分析[2]。Pfleeger 和 Bohner 定义变化影响分析为"估计与变化相关的危险性,包括估计对资源、计划和进度的影响"[3]。Turver 和 Munro 定义变化影响分析为"估计某个模块中的变化对系统中其他模块的影响,确定一个变化影响到的范围以及提供一个复杂的衡量方法"[4]。Rothermel 等提出通过构建程序依赖图来表示数据依赖和控制依赖,并找出关于继承、封装和多态等构造关系的子类和其它类[5]。该方法主要从类粒度级进行静态变化影响分析,虽然花费代价相对较小,但是不够精确。刘震等人提出动态分析技术构造 Java 程序的动态调用图[6],采用 K-类方法后向切片计算影响集,较文献[5]精确,但不适用于语句粒度级。

基于路径的测试是软件测试的重要内容。文献[7]基于程序控制流图(Control Flow Graph,CFG)生成基本路径集,提出面向基本路径的软件测试。文献[8]通过构造类依赖图和基本消息流图,建立依赖分析模型,基于该模型得到测试路径,但该方法需借助 UML 模型,使用范围有限。文献[9]采用基于函数调用的路径覆盖生成技术,通过对软件源代码的静态分析并结合程序运行,获取动态执行路径,与静态路径进行匹配,从而判断测试用例是否覆盖了程序变更部分及受影响部分,但动态调用中需对函数前后进行插桩,增加了系统开销。

目前,变化影响分析方法使用较多的是基于程序依赖关系的分析,主要包括数据依赖和控制依赖。在单元测试中,本文基于程序源代码,分析方法或类内部的语句依赖关系,如循环结构,因此将程序的控制依赖分析作为主要考量,以便快速定位变化及分析影响。本文基于程序的语句粒度级,首先比

到稿日期:2015-02-01 返修日期:2015-04-02 本文受国家自然科学基金(60703116,61063006,61462049)资助。
郭丹丹 女,硕士,主要研究方向为软件测试、软件质量保证,E-mail:2009ly_dan@163.com;姜 瑛(1974-),女,博士,教授,主要研究方向为软件工程、软件测试、服务计算,E-mail:jy_910@163.com(通信作者)。

较修改前后程序静态结构，确定变化范围，然后分析变化波及的范围并生成变化影响路径集，以缩小回归测试的范围，提高测试效率。

本文第 2 节介绍程序的变化范围；第 3 节给出一种基于程序源代码的变化影响分析方法；第 4 节讨论利用影响集得到回归测试影响路径集；第 5 节是实验对比和结果分析；最后是结论以及下一步工作展望。

2 变化范围

近年来，程序变化研究主要通过分析程序中的相应关系找到实体与实体间的依赖关系。对于需求和设计层次，往往需要借助抽象模型执行变化影响分析，这些技术不需要访问源代码，但是不如基于源代码的变化影响分析技术成熟。因此，基于源代码的变化影响分析技术在识别软件变化影响中占有很大优势[10]。

在变化影响分析中，变更的粒度直接决定了影响的精确度，同时也决定了回归测试的范围。文献[10]中将变化粒度由粗到细划分为文本变更请求、文件变更、类变更、方法变更、属性变更、语句变更，并分别统计了不同粒度级的相关研究工作。Petrenko 等人经过验证后指出变化元素的粒度越细，其影响结果就越精确[11]。

3 变化影响分析

程序 CFG 主要体现了程序的控制结构，便于直观分析结点之间的变化影响。

定义 1 CFG 是具有唯一的入口和出口结点的有向图 $G=(V(G),E(G))$。其中 $V(G)=\{v_0,\cdots,v_n\}$ 是一个非空集合，集合元素为 CFG 结点，对应程序中的每条语句；$E(G)=\{(v_i,v_j)|v_i,v_j\in V(G)\}$ 是图中有向边的集合，反映了语句间的控制流关系[12]。

CFG 的生成可通过深度遍历程序语法树，并对语法树进行修剪生成抽象语法树(Syntax Tree, AST)[12]。程序 CFG 反映了程序语句间的依赖关系，当程序的源代码发生变化时，其程序结构也随之发生变化。因而，通过扫描原程序和变化后程序，比较变化前后的程序源代码可以得出变化的内容，映射到控制流图中，即可得到变化结点的集合。

定义 2 用 $CS(Change\ Set)=\{v_p|v_p\in V(G),p=1,2,\cdots,n-1\}$ 表示变化集。其中 v_p 表示变化结点，不包含入口结点 v_0 和出口结点 v_n。

程序中某条语句的修改会对其他有依赖关系的语句产生影响，这就是波动效应[11]。对应到 CFG 中，若变化结点 v_i 与 v_j 邻接，则 v_i 一定影响 v_j。若 v_i 经过多条有向边与 v_j 连接，则 v_i 受波动效应也会影响 v_j。

定义 3 用 $IS(Impact\ Set)=\{v_q|v_q\in V(G),q=1,2,\cdots,n-1\}$ 表示受变化集 CS 中结点影响的所有结点集。其中 v_q 表示影响结点，不包含入口结点 v_0 和出口结点 v_n。

基于以上定义，为了更直观地分析各个结点之间的邻接关系，使用邻接矩阵表示 CFG，将其转化成的可达矩阵可分析图中任意两个结点间的影响关系。

定义 4 可达矩阵[2]定义为：$P=(p_{ij})_{n*n}$。

其中，$p_{ij}=\begin{cases}1, & \text{从 } v_i \text{ 到 } v_j \text{ 存在通路}\\ 0, & \text{从 } v_i \text{ 到 } v_j \text{ 不存在任何通路}\end{cases}$

在 CFG 中，由于结点变化引起的波动效应经一条通路中的相邻结点传播，因此受变化影响的结点可通过寻找通路中变化结点的相邻结点得到，直至找到程序出口结点时，影响结束。本文提出由变化集 CS 得到影响集 IS 的算法描述如图 1 所示。

```
输入：变化集 CS，邻接矩阵 A=a[i][j]
输出：影响集 IS
{
    /*利用 Warshall 算法生成可达矩阵 P*/
    P=GetKDMatrix(A)
    /*由变化结点 vp 开始遍历可达矩阵中值为 1 的结点，加入到
     影响集 IS 中*/
    for each(vp∈CS){
      for (int m=0;m<=n-1;m++){
      if( P[vp.getIndex()][m]==1){//判断 vp 所在行受影响的
    结点
        for each(vp∈V(G)){
          if(i==m) {
            vi.isImpact=true;//设置结点属性为受影响，加入到
            IS 集合中
            add vi into IS;
            }
          }
        }
      }
    }
}
```

图 1 变化影响分析算法

4 影响路径集生成

CFG 是源代码的直接映射，如果源代码发生变化，将会在 CFG 中体现。为了确定变化及其波及范围，需要对变化后的程序结构进行分析，并且生成程序变化后的测试路径集。但是对于路径数目增加较多，尤其是程序中环路较多的情况，遍历 CFG 重新生成测试路径集时将耗费更大的代价。根据第 3 节提出的变化影响分析方法，确定变化集 CS 和影响集 IS 后，只需在原有程序基本路径集的基础上生成覆盖所有变化集 CS 和影响集 IS 中结点的影响路径集，不需要对所有路径重新测试。

4.1 程序路径集

程序的一条路径是指程序中顺序执行的一个语句序列，由 CFG 中包含入口和出口结点的一个结点序列组成。如果程序中出现多个循环，路径数目可能急剧增加，为解决这一问题，必须对循环机制进行简化以限制循环的次数。文献[13]提出只考虑执行时进入循环体 1 次和 0 次的情况。而为了更好地体现路径中的循环执行，需再增加循环体执行 2 次的路径。

本文在程序 CFG 的基础上，通过计算程序环路复杂度，采用文献[14]的方法，生成程序的基本路径集 Paths。程序中若含有循环结构，需要增补循环体执行 2 次的路径。生成增补路径集的算法描述如图 2 所示。

```
输入:基本路径集 Paths
输出:增补路径集 addPaths
{
    foreach(path∈Paths){//遍历基本路径集中每条路径
      foreach (v_i∈path){//遍历每条路径中的每个结点
        if(v_i is not WhileEntry){//判断该结点是否是循环入口结点
          add v_i into List<V(G)> pt_0;//v_i 加入到结点集 pt_0 中
        }else if(v_i is WhileEntry and v_i. next is TrueBranch){//判
        断该结点是否是真分支结点
          while(v_i. next!=v_i){//v_i 邻接结点是否是 v_i 本身
          add v_i. next into List<V(G)> pt_1;//v_i 邻接结点加入到
          结点集 pt_1 中
            }
          }else{
        add v_i into List<V(G)> pt_2;
          }
        add pt_0,pt_1,pt_1,pt_2 into addPath;//合并后生成循环 2 次
        的结点序列
        add pt_0,pt_2 into addPath;//合并后生成循环 0 次的结点序列
        }
    add addPath into addPaths;//去掉重复路径,生成增补路径集
    }
}
```

图 2　增补路径算法

4.2　影响路径集

影响路径是指从入口结点开始经过若干影响结点到达出口结点的路径[15]，是在基本路径集基础上生成的覆盖变化影响结点的路径。

定义 5　用 $IPath(v_0, v_n)$ 表示一条影响路径。其中 v_0 和 v_n 分别代表入口结点和出口结点，$v_i(i\in(0,n))$ 必定满足 $v_i\in CS\cup IS$。

程序的变化影响是按照程序的控制关系传递的，体现在 CFG 中是按照结点之间依赖关系自上而下进行传递，受变化结点影响的结点位于从变化结点到出口结点的通路中。因此，在生成影响路径集时，可以先找到从入口结点 v_0 到变化结点 v_p 的通路，然后与变化结点 v_p 到出口结点 v_n 的通路合并成为一条影响路径 $IPath$。生成影响路径集的算法描述如图 3 所示。

```
输入:基本路径集 Paths,增补路径集 addPaths
输出:影响路径集 IPaths
{
  add Paths and addPaths into List<List<V(G)>>Paths_all;
  //Paths_all 集合存放程序路径集;
    for each(Paths_all){
      for each(v_j∈Paths_all. get(i)){
        if(v_j∈CS){
          int k =j;
          while(k<path. size()){
            add path. get(k) into List<V(G)>Path1;//Path1 集合
            存放由变化结点到出口结点的结点序列
              k++;}
    for(int m=0;m<j;m++){
      add path. get(m) into List<V(G)>Path2;//Path2 集合存放由
      入口结点到变化结点的结点序列
          }
        }
      }
    }
    IPaths=Path2×Path1;//将 Path2 中路径与 Path1 中序列做笛卡
    尔积,去掉重复路径,生成 IPaths
}
```

图 3　生成影响路径算法

回归测试时对以上方法生成的影响路径集进行测试，可以有效缩小回归测试的范围。

5　实验

为了验证本文方法的有效性，利用 Eclipse 开发了一个生成程序影响路径集的原型工具，并进行了大量实验，下面主要选取 2 个实验进行说明。

5.1　实验 1

首先，通过实验比较了基本路径集与影响路径集的测试效果。下面以一个简单的 JAVA 程序为例进行说明，该程序功能是取任意 3 个数的最大值。在程序中手动植入 4 个错误，错误对象包括参数名、参数类型及逻辑错误。根据文献[12]的方法对源程序构建 AST 并进行遍历，通过可视化工具 Graphviz 生成了程序的 CFG，如图 4 所示。

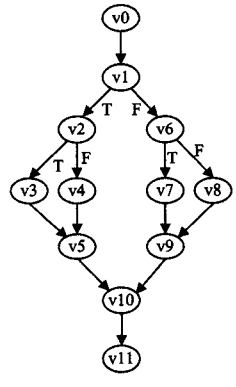

图 4　程序 CFG 图

通过对 CFG 遍历，生成 4 条程序基本路径集。例如，修改了程序的源代码，使用第 2 节中的方法找到 $CS=\{v_6, v_7\}$，分析得到 $IS=\{v_8, v_9, v_{10}\}$。从入口结点生成到变化结点的通路 $\{v_0, v_1, v_6\}$，从变化结点到出口结点的通路 $\{\{v_6, v_7, v_9, v_{10}, v_{11}\}, \{v_6, v_8, v_9, v_{10}, v_{11}\}\}$，最终得到影响路径集，如表 1 所列。

表 1　基本路径集与影响路径集

基本路径集	影响路径集
Path1: $v_0, v_1, v_2, v_3, v_5, v_{10}, v_{11}$	IPath1: $v_0, v_1, v_6, v_7, v_9, v_{10}, v_{11}$
Path2: $v_0, v_1, v_2, v_4, v_5, v_{10}, v_{11}$	IPath2: $v_0, v_1, v_6, v_8, v_9, v_{10}, v_{11}$
Path3: $v_0, v_1, v_6, v_7, v_9, v_{10}, v_{11}$	
Path4: $v_0, v_1, v_6, v_8, v_9, v_{10}, v_{11}$	

借助已开发的原型工具自动生成初始测试用例集，保存到 XML 文档中。实验过程中分别用初始测试用例集运行基

本路径集及影响路径集，根据动态执行的路径筛选测试用例。分析了测试用例的查错率、执行时间等，结果如表2所列。

表2 基本路径集与影响路径集的测试结果

序号	植错数	基本路径数	影响路径数	基本路径集			影响路径集		
				测试用例数	查错率	执行时间(ms)	测试用例数	查错率	执行时间(ms)
1	4	4	2	216	0.528	47	127	0.886	16
2	5	4	3	216	0.879	31	203	0.931	15
3	6	4	2	216	0.880	31	203	0.901	15
4	6	4	3	216	0.879	32	177	0.936	15
5	5	4	2	216	0.532	31	128	0.887	16
6	5	4	3	216	0.877	32	178	0.940	15
7	5	4	3	216	0.885	31	204	0.931	16
8	6	4	3	216	0.528	31	101	0.886	16
9	5	4	2	216	0.417	31	103	0.855	16
10	5	4	3	216	0.473	31	115	0.745	16

从上述对比结果可以看出，只要不改变程序的结构，程序基本路径集通常是相对固定的，而修改内容会直接导致影响路径集发生变化。统计10次实验的平均结果可以看到，针对影响路径进行回归测试，测试用例的数量减少了29.9%，执行时间减少了52.1%，而查错率提高了29.4%。

5.2 实验2

其次，通过实验对本文方法与文献[6]的方法进行了比较。文献[6]基于Java程序的动态调用图，采用K-类方法后向切片计算影响集。实验中分别对2个Java程序进行了分析，其中TreeNode程序包含3个类16个方法，ATM2程序包含4个类23个方法。通过修改程序的一些内容植入相同的错误数，然后用本文方法和文献[6]的方法分别执行216个测试用例，并统计了这两种方法的查错率及执行时间，结果如表3所列。

表3 本文方法和文献[6]方法的实验结果

程序	序号	采用方法	植错数	查错率	执行时间(ms)
TreeNode	1	本文方法	4	0.750	16
		文献[6]方法	4	0.750	16
	2	本文方法	8	0.875	17
		文献[6]方法	8	0.625	16
ATM2	1	本文方法	4	0.750	17
		文献[6]方法	4	0.750	17
	2	本文方法	8	0.875	18
		文献[6]方法	8	0.750	17

分析实验1和实验2可以看出，本文方法从语句粒度级出发，利用CFG分析可得到全部受变化影响的路径，能够快速有效地识别语句粒度级修改，不会因动态插桩而有所遗漏。与文献[6]相比，本文方法在单位时间内平均查错率较高，粒度级更细，且系统消耗也不高，能够缩小回归测试范围，提高回归测试效率。

结束语 在路径测试的基础上，本文利用基于源代码的变化影响分析技术得到重新需要测试的影响路径，缩减了回归测试的范围。但是，本文仅分析了程序方法或类内部的语句粒度级变化及其影响，层次范围有限。在后续研究中，将继续研究多粒度的变化影响分析以及相应的测试用例优化筛选。

参 考 文 献

[1] 刘志清. 变更管理系统的研究和实现[D]. 吉林：吉林大学，2009
Liu Z Q. Change Management System Research and Implement [D]. Jilin: Jilin University, 2009

[2] 王映辉，张世琨，刘瑜，等. 基于可达矩阵的软件体系结构演化波及效应分析[J]. 软件学报，2004，15(8)：1107-1115
Wang Y H, Zhang S K, Liu Y, et al. Ripple-Effect Analysis of Software Architecture Evolution Based on Reachability Matrix [J]. Journal of Software, 2004, 15(8): 1107-1115

[3] Pfleeger S L, Bohner S A. A framework for software maintenance metrics: Software Maintenance[C]// Proceedings Conference on IEEE. 1990: 320-327

[4] Turver R J, Munro M. An early impact analysis technique for software maintenance[J]. Journal of Software Maintenance: Research and Practice, 1994, 6(1): 35-52

[5] Rothermel G, Harrold M J. Selecting regression tests for object-oriented software: Software Maintenance[C]// Proceedings of International Conference on IEEE. 1994: 14-25

[6] 刘震，缪力. 基于动态调用图的Java程序修改影响分析技术[J]. 湖南师范大学(自然科学学报)，2011，34(6)：26-30
Liu Z, Miao L. Java Programs Change Impact Analysis Based on Dynamic Call Graph[J]. Hunan Normal University(Natural Science), 2011, 34(6): 26-30

[7] 赵家玉. 面向基本路径的软件测试研究[J]. 电脑知识与技术，2011，7(11)：2583-2586
Zhao J Y. Research on Software Testing for Basic Path[J]. Computer Knowledge and Technology, 2011, 7(11): 2583-2586

[8] 陈树峰. 基于UML模型的依赖分析在回归测试中的研究与应用[D]. 南京：南京航空航天大学，2010
Chen S F. Research and Application of Dependence Analysis Based on UML Model in Regression Test [D]. Nanjing: Nanjing University of Aeronautics and Astronautics, 2010

[9] 张志华，牟永敏. 基于函数调用的路径覆盖生成技术研究[J]. 电子学报，2010，38(8)：1808-1811
Zhang Z H, Mu Y M. Research of Path Coverage Generation Techniques Based Function Call Graph [J]. Acta Electronica Sinica, 2010, 38(8): 1808-1811

[10] Li B, Sun X, Leung H, et al. A survey of code-based change impact analysis techniques[J]. Software Testing, Verification and Reliability, 2013, 23(8): 613

[11] Petrenko M, Rajlich V. Variable granularity for improving precision of impact analysis[C]// IEEE 17th International Conference on Program Comprehension(ICPC'09). IEEE, 2009: 10-19

[12] 秦利勇. 白盒构件测试数据自动生成方法研究与实现[D]. 昆明：昆明理工大学，2012
Qin L Y. Research and Implementation on Method of White-box Component Test Data Automatic Generation [D]. Kunming: Kunming University of Science and Technology, 2012

[13] 周晓波. 构件回归测试方法研究与实现[D]. 昆明：昆明理工大学，2012
Zhou X B. Research and Implementation for Method of Component Regression Testing [D]. Kunming University of Science and Technology, 2012

[14] 丁振国，郭强. 基于程序控制的路径测试技术研究[J]. 电子科技，2008，21(12)：53-56
Ding Z G, Guo Q. Research on the Program Control Based Path Coverage Testing Technique[J]. Electronic Technology, 2008, 21(12): 53-56

[15] 陈树峰，郑洪源. 面向对象软件的依赖性分析与回归测试[J]. 计算机应用，2009，29(11)：3110-3113
Chen S F, Zheng H Y. Dependence analysis and regression testing of object-oriented software[J]. Journal of Computer Applications, 2009, 29(11): 3110-3113