文章编号：2095-2163(2019)02-0251-04 中图分类号：TP311.52 文献标志码：A

基于程序谱的方法级别错误定位技术

徐 迟，苏小红，王甜甜
（哈尔滨工业大学 计算机科学与技术学院，哈尔滨 150001）

摘 要：自动化的软件错误定位是软件调试过程中的一个热点问题，旨在更高效地发现软件中的错误。然而，目前的软件错误定位技术大多是针对语句级别的定位，而语句级别的定位精度不是很高，一旦定位失败，程序员就会很难发现软件中的错误。为了进一步地提高软件错误定位的效率，本文提出了一种基于程序谱的方法级别错误定位技术。方法级别的错误定位相比于语句级别的错误定位，其优势在于，方法级别定位的效率会更高，可以首先确定 bug 语句存在于待测程序的某个方法中，然后通过人工调试的方法进一步确定 bug 语句的具体位置。本文首先提出了一种基于程序谱的方法级别错误定位方法，随后，对本文的方法进行了实验分析。实验表明，本文方法能够有效提高方法级别错误定位的精度。

关键词：错误定位；程序谱；方法级别

Spectrum-based function-level fault location technology

XU Chi, SU Xiaohong, WANG Tiantian
(School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China)

【Abstract】 Automatic software fault location is a hot issue in the software debugging process, aiming to find errors in the software more efficiently. However, most current software fault location technologies are aimed at the statement-level locating, and the statement-level locating accuracy is not very high. Once the location fails, the programmers can hardly find errors in the software. In order to further improve the efficiency of software fault location, this paper proposes a spectrum-based function-level fault location method. The advantage of function-level fault location compared to statement-level fault location is that method-level locating is more effective. It is first determined that a bug statement exists in a function of the test program, and then the specific location of the bug statement is manually determined. This paper first proposes a spectrum-based function-level fault location method. And the method of this paper is analyzed experimentally. Experiments show that this method can effectively improve the accuracy of function-level fault location.

【Key words】 fault location; spectrum; function-level

0 引 言

目前基于程序谱的错误定位方法[1]有很多，例如 Tarantula 方法[2]、Ochiai 方法[3]等等，其中 Tarantula 方法也经常用来作为实验的对比方法。然而，这类方法的可疑度计算公式中的参数，其实并不适合用来进行方法级别的定位。本文接下来首先探讨了基于程序谱的错误定位相关问题，然后分析了 Tarantula 方法在方法级别定位上的不足之处，并且给出了 Optimal Ranking 错误定位方法。最后，通过实验分析，验证了本文的方法能够提高方法级别定位的效率。

1 错误定位相关问题

软件错误定位的目的就是为了找到待测程序中存在的 bug，待测程序通常表示为 P，P 中的程序实体可以表示为：

$$P = \{p_1, p_2, p_3, \cdots, p_m\}, \quad (1)$$

其中，m 表示为 P 中的程序实体数量，本文中，程序实体都是指待测程序中的方法。$p_i(1 \leqslant i \leqslant m)$ 表示 P 中的第 i 个方法。

待测程序 P 的测试用例集表示为：

$$T = \{t_1, t_2, t_3, \cdots, t_n\}, \quad (2)$$

其中，待测程序 P 的测试用例集 T 中的测试用例数量为 n，$t_i(1 \leqslant i \leqslant n)$ 表示 T 中的第 i 个测试用例。同时，用 T_F 表示执行失败的测试用例集合，T_P 表示执行成功的测试用例集合。

使用测试用例去执行待测程序得到的结果表示为：

$$R = \{r_1, r_2, r_3, \cdots, r_n\}, \quad (3)$$

其中，n 表示测试用例的个数，$r_i(1 \leqslant i \leqslant n)$ 表

基金项目：国家自然科学基金(61672191)；“十三五”国家重点研发计划课题(2017YFC0702204)。

作者简介：徐 迟(1992-)，男，硕士研究生，主要研究方向：软件错误定位；苏小红(1966-)，女，博士后，教授，博士生导师，主要研究方向：软件工程、智能信息处理与信息融合、图像处理等；王甜甜(1980-)，女，博士，副教授，主要研究方向：程序分析、计算机辅助教学。

收稿日期：2018-06-05

示第 i 个测试用例的执行结果，r_i 的取值有 2 种可能，1 或者 0。如果 $r_i = 0$，就表示第 i 个测试用例的执行结果为成功；如果 $r_i = 1$，就表示第 i 个测试用例的执行结果为失败。

用矩阵 $\boldsymbol{M}$ 来表示每个测试用例的覆盖信息，其数学表述如下：

$$\boldsymbol{M} = (\boldsymbol{M}_{ij})_{1 \leqslant i \leqslant m,\ 1 \leqslant j \leqslant n}. \tag{4}$$

其中，$\boldsymbol{M}$ 为一个 m 行 n 列的矩阵，$\boldsymbol{M}_{ij}$ 表示测试用例 t_j 执行程序实体 p_i 的情况。如果 $\boldsymbol{M}_{ij} = 1$，就表示测试用例 t_j 执行了程序实体 p_i；如果 $\boldsymbol{M}_{ij} = 0$，就表示测试用例 t_j 没有执行到程序实体 p_i。

覆盖信息矩阵 $\boldsymbol{M}$ 和结果向量 $\boldsymbol{R}$ 共同组成了记录程序谱信息的 0－1 矩阵，0－1 矩阵将作为本文研究的错误定位方法的输入。本文研究中选取的示例程序如图 1 所示。

```
void main(int a, int b, int c)
{
  f1( );
  if(a==0)
    f2( );
  else
  {
    if(b!=0)
    {
      f3(c);//含有 bug 语句的方法
    }
    if(a+b==5)
    {
      f4( );
    }
    if(a>=b)
      f5( );
  }
  f6( );
}
```

(a)一段含有 bug 语句的代码

(a) The code containing bug statements

```
void f3(int c)
{
  if(c>0)
  {
    printf("%d",-c);
    //正确语句应该是
    printf("%d",c);
  }
  else
  {
    printf("%d",-c);
  }
}
```

(b)含有 bug 语句的方法 f_3

(b) f_3 method containing bug statements

图 1 示例程序

Fig. 1 Sample program

研究中针对图 1 的示例程序，给出了 6 组测试用例进行测试得到的 0－1 矩阵，其最终直观展示可见表 1。

表 1 示例程序的 0－1 矩阵

Tab. 1 0－1 matrix of sample program

	t_1 {0,1, －1}	t_2 {2,3,0}	t_3 {4,2, －1}	t_4 {1,2, －2}	t_5 {3,1,2}	t_6 {3,2, －2}
f_1	1	1	1	1	1	1
f_2	1	0	0	0	0	0
f_3	0	1	1	1	1	1
f_4	0	1	0	0	0	1
f_5	0	0	1	0	1	1
f_6	1	1	1	1	1	1
result	0	1	0	0	1	0

0－1 矩阵中的每一列表示测试用例 t_i 是否调用了程序中的方法，最后一行表示测试用例 t_i 的执行结果。

2 Optimal Ranking 错误定位方法

2.1 方法设计流程

Optimal Ranking 方法的整体设计流程如图 2 所示。首先基于待测程序的抽象语法树，进行方法级别的程序插桩。然后使用测试用例去执行插桩后的程序得到方法的调用序列。接下来根据方法的调用序列，统计方法的调用信息，使用 Optimal Ranking 的可疑度数学公式来计算每个方法的可疑度。最后对每个方法根据可疑度的大小进行降序排列，得到方法可疑度的排序列表，用于对此后定位结果的研究与分析。

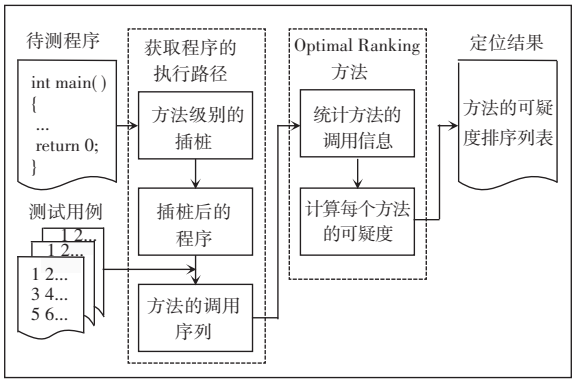

图 2 方法的整体流程图

Fig. 2 The overall flow chart of the method

2.2 方法级别定位和语句级别定位的区别

方法级别定位和语句级别定位主要的不同之处，就在于程序的执行路径和其相对应的执行结果上。下面将以图 1 中的示例程序来进行阐释说明。

图 1(a)所示的是一段含有 bug 语句的代码，bug 语句所在的方法为 f_3。图 1(b)中，给出了含有 bug 语句的方法 f_3 的具体功能。可以看到，f_3 方法的第 5 行代码出现了错误，程序的本意是想输出 c 的值，但是却误写成了输出 $-c$ 的值。

如果采用的是语句级别的错误定位方法，那么给定测试用例 t 的前提下，该研发代码片段只有执行到了 f_3 的第 5 行语句的时候，测试用例 t 的执行结果才可能是失败(不考虑偶然正确性测试用例的前提下)。

而对于方法级别的错误定位来说，研究中插桩的粒度是方法级别，因此程序的执行路径不会考虑到方法中具体的某一条语句是否执行。那么，当使用测试用例 t 执行这段代码时，如果没有执行到 f_3

中的第 5 行错误语句，那么测试用例 t 的执行结果就是成功的。这样，根据基于覆盖的错误定位方法的理论，被成功的测试用例执行到的程序体应该更少的怀疑，同时想定位到的包含 bug 语句的 f_3 方法的怀疑度就会降低。而研究中设定的理想情况是，只要执行到了包含 bug 语句的方法 (f_3)，那么测试用例的执行结果应该就是失败的。

例如，图 1(a) 中的程序输入测试用例 $t = \{2,1,-1\}$。显然这个测试用例的执行结果是成功的。对于语句级别的定位来说，测试用例 t 并没有执行 f_3 中第 5 行的错误语句；而对于方法级别的定位来说，测试用例 t 却调用了 f_3。这样，因为程序执行了 f_3，且得到了正确的输出结果，那么 f_3 的怀疑度明显就会降低。

2.3　可疑度计算公式

根据 2.2 节所述，在方法级别的错误定位上，测试用例虽然调用了包含错误语句的方法，但实际上却没有执行到具体的错误语句，所以就会导致研究拟设的执行结果是失败的测试用例，而实际的执行结果却是成功的。这样必定会影响错误定位的精度。

此时，研究是否可以换个角度，考虑那些没有被失败测试用例 T_f 执行到的方法。因为这些方法没有被失败测试用例 T_f 执行到，就意味着，导致测试用例执行失败的错误语句，一定不会在这些方法中。因此，可以给这些方法赋一个比较低的可疑度值。

基于上文分析，研究给出了针对方法级别错误定位的 Optimal Ranking 方法（简称 O 方法）。O 方法的可疑度的计算公式为：

$$O(a) = \begin{cases} -1, & if\ a_{nf} > 0, \\ a_{np}, & otherwise. \end{cases} \tag{5}$$

其中，a 表示待测程序中的某个方法。方法 a 的第一个下角标表示 a 是否被测试用例执行，n 表示没被执行，e 表示被执行。方法 a 的第二个下角标表示测试用例的执行是否成功，p 表示成功，f 表示失败。因此 a_{nf} 就表示没有调用到方法 a 的失败测试用例个数，a_{np} 表示没有执行到方法 a 的成功测试用例个数。

从公式(5) 中可以看到，O 方法的可疑度计算公式相比于其它的可疑度计算方法，使用了 a_{nf} 和 a_{np} 这 2 个在其它方法中并未居于重要位置的变量。如果所有没执行到方法 a 的测试用例中有失败的测试用例，即 $a_{nf} > 0$，就可给这些方法的可疑度赋值为 -1。此外，如果 2 个方法的 $a_{nf} = 0$，则这两者一定有相同的 a_{ef}，因为 $a_{nf} + a_{ef}$ 是所有失败测试用例的总数。同理，$a_{np} + a_{ep}$ 是所有成功测试用例的总数。分析至此可以推知，对于包含错误语句的方法，只有小部分执行到此方法的测试用例是成功的，即 a_{ep} 的值很小，因此 a_{np} 的值就会相对较大。所以对于那些 a_{nf} 值不为 0 的方法，用 a_{np} 来表示其对应的可疑度是非常合理的。

2.4　实例分析

研究中设计选取的示例程序使用 Tarantula 方法和 O 方法的方法级别错误定位结果可见表 2。可以看到，使用 Tarantula 方法计算方法的可疑度，包含错误语句的方法 f_3 的可疑值并没有排在第一位；而是用 O 方法计算方法的可疑度，f_3 方法的可疑度值则是最大的。显然，对于示例程序，O 方法的定位效果要优于 Tarantula 方法的定位效果。接下来将详尽分析为何会得到这样的结果。

首先，研究中将给出 Tarantula 方法的可疑度计算公式的数学表述如下：

$$Tarantula(a) = \frac{\dfrac{a_{ef}}{a_{ef} + a_{nf}}}{\dfrac{a_{ep}}{a_{ep} + a_{np}} + \dfrac{a_{ef}}{a_{ef} + a_{nf}}}. \tag{6}$$

其中，各符号表示的含义和公式(5) 中各符号的含义相同。

表 2　示例程序的方法级别定位结果对比

Tab. 2　Comparison of function-level location results for sample program

	t_1 {0,1,−1}	t_2 {2,3,0}	t_3 {4,2,−1}	t_4 {1,2,−2}	t_5 {3,1,2}	t_6 {3,2,−2}	Tarantula	O
f_1	1	1	1	1	1	1	0.500	0
f_2	1	0	0	0	0	0	0	−1
f_3	0	1	1	1	1	1	0.571	1
f_4	0	1	0	0	0	1	0.633	−1
f_5	0	0	1	0	1	1	0.500	−1
f_6	1	1	1	1	1	1	0.500	0
result	0	1	0	0	1	0		

仔细观察表 2 中的数据可以发现，t_3、t_4、t_6 这 3 组测试用例虽然调用了方法 f_3，但是因为没有执行 f_3 中的错误语句，因此，这 3 组测试用例得到的执行结果都是正确的，这样就会导致 a_{ep} 的值变大，Tarantula 方法计算的可疑度值就会变小。而理想情况下，研究希望 t_3、t_4、t_6 这 3 组测试用例的执行结果都是失败的，这样 a_{ef} 的值就会变大，Tarantula 方法计算的可疑度值也会变大。此消彼长，就会导致 Tarantula 方法计算的 f_3 方法的可疑值变小。

如果使用 O 方法计算可疑度，分析得知 f_2、f_4、f_5 这 3 个方法的 a_{nf} 值都是大于零的，因此根据公式(5)，就将被赋予最小的可疑度值。而 f_3 方法的 a_{nf} 值是等于零的，因此随即会被赋予更大的可疑度值。

3 实验分析

为了进一步验证在方法级别的错误定位上 O 方法的定位效果，本文使用了 Siemens 测试集[4]进行实验，并且将 Tarantula 方法作为本文实验的对比方法。实验的评价指标选择的是成功定位百分比(*PSD*)[5]，表示通过检查待测程序中特定百分比的方法，成功检查到包含错误语句方法的程序占总程序的百分比。研究推得其计算公式为：

$$PSD(a\%) = \frac{successful\_findbug\_num}{program\_total\_num}. \quad (7)$$

其中，$a\%$ 表示检测待测程序中 $a\%$ 的方法；*program_total_num* 表示待测程序的总数；*successful_findbug_num* 表示所有待测程序中，可疑度排名前 $a\%$ 的方法中可以找到包含错误语句的方法的程序数。

基于如上分析，仿真运行得到在 Siemens 测试集上，本文的方法和 Tarantula 方法在方法级别错误定位上的定位精度折线图的展示效果如图 3 所示。横轴表示的是检查的方法百分比，纵轴表示的是检查相应百分比的方法时，成功定位到包含错误语句方法的程序数占总程序数的百分比。研究中，测试集的方法数都小于 20，因此横轴从 10%开始。从图 3 中可以看到，检查相同百分比的方法时，O 方法相比于 Tarantula 方法，能够定位到更多的错误。此外，O 方法想要定位到全部的错误需要检查 90%左右的方法，而 Tarantula 方法则需要检查全部的方法。

综合如上实验分析结果，可以得出如下研究结论：在方法级别的错误定位上，O 方法的定位效果相比于 Tarantula 方法的定位效果要更好一些。

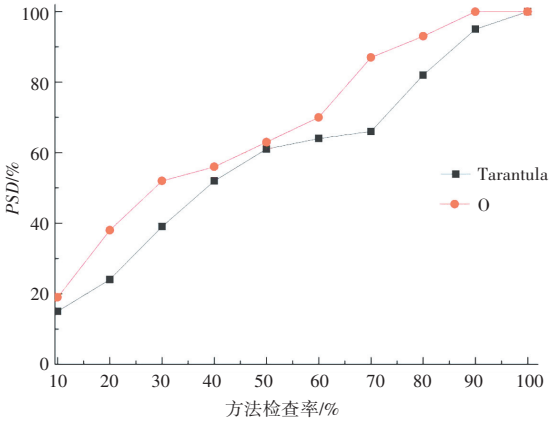

图 3 方法级别错误定位结果

Fig. 3 Results of function-level fault location

4 结束语

针对语句级别定位效率不高的问题，本文提出了方法级别的错误定位方法。并且通过分析基于程序谱的方法级别错误定位和语句级别错误定位的不同，进一步重点说明了有些错误定位方法的可疑度计算公式不适合应用在方法级别的错误定位上。最后通过实验验证了 O 方法在方法级别定位上有相对更好的定位效果。

此外，除了基于程序谱的方法级别错误定位方法，某些非程序谱的错误定位方法也可以应用于方法级别的错误定位上，这些均有待未来的深入探讨与研究。

参考文献

[1] 曹鹤玲，姜淑娟，鞠小林. 软件错误定位研究综述[J]. 计算机科学，2014，41(2)：1-6，14.

[2] JONES J A，HARROLD M J. Empirical evaluation of the tarantula automatic fault-localization technique[C]//Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering. Long Beach，CA，USA：ACM，2005：273-282.

[3] ABREU R，ZOETEWEIJ P，VAN GEMUND A J C. On the accuracy of spectrum based fault localization[C]//Proceedings of Testing：Academic and Industrial Conference-Practice and Research Techniques. Washington，DC，USA：IEEE Computer Society，2007：89-98.

[4] HUTCHINS M，FOSTER H，GORADIA T，et al. Experiments on the effectiveness of data flow and control-flow-based test adequacy criteria[C]// Proceedings of the 16th International Conference on Software Engineering. Sorrento，Italy：IEEE，1994：191-200.

[5] NAISH L，LEE H J，RAMAMOHANARAO K. A model for spectra-based software diagnosis[J]. ACM Transactions on Software Engineering & Methodology，2011，20(3)：1-32.