通过增大边际权重提高基于频谱的错误定位效率

谭德贵 陈 林 王子元 丁 晖 周毓明 徐宝文

(南京大学软件新技术国家重点实验室 南京 210093)

(南京大学计算机科学与技术系 南京 210093)

摘 要 基于频谱的错误定位技术通常利用覆盖信息来求出程序中每条语句的可疑度,并将语句按照可疑度降序排序以寻找错误语句.文中对已有的基于频谱的错误定位算法进行改进,将失败测试用例的边际权重引入到可疑度计算的过程中,即针对某一特定语句,令失败测试用例的权重随着其对该语句覆盖次数的增加而增大.实验结果表明,相对于其它方法,文中提出的方法对错误定位效率有一定的促进作用,即只需检查更少的语句即可找到出错位置.

关键词 软件测试;程序分析;错误定位;覆盖信息

中图法分类号 TP301 **DOI 号**: 10.3724/SP.J.1016.2010.02335

Spectra-Based Fault Localization by Increasing Marginal Weight

TAN De-Gui CHEN Lin WANG Zi-Yuan DING Hui ZHOU Yu-Ming XU Bao-Wen

[1)] (*National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing*, 210093)

[2)] (*Department of Computer Science and Technology, Nanjing University, Nanjing* 210093)

Abstract Spectra-based fault localization technique uses coverage information to calculate every statement's likelihood of having a bug. And then rank the likelihood in a decreasing order to find the faulty statement. This paper improves the spectra-base fault localization technique by increasing the marginal weight of the failed test cases. That means that as the number of the failed test case increases, the weight of the failed test case also increases. Comparing with reducing or sustaining the weight of covered statement's successful/failed test case, the experimental result shows that increasing the marginal weight of the covered statement's failed test case can promote the fault localization efficiency.

Keywords software testing, program analysis, fault localization, coverage information

1 引 言

错误定位技术因其重要性与困难性已经受到软件工程研究者的广泛关注,许多研究者提出了各种各样的错误定位方法,但都未能很好地解决软件的错误定位问题.调试是进行错误定位的最常见方法.该方法在程序中设置若干断点,之后运行失败测试用例,从断点处开始根据程序执行路径和状态定位错误语句.这种方法有两个很大的缺点:首先是程序的断点位置不容易确定;其次这种方法需要手工排查很多语句,当程序比较庞大时,这种方法

收稿日期:2010-08-22;最终修改稿收到日期:2010-09-29.本课题得到国家自然科学基金(90818027, 60873050)、国家"八六三"高技术研究发展计划专题项目基金(2009AA01Z147)、国家"九七三"重点基础研究发展规划项目基金(2009CB320703)和上海市科委重点实验室基金(09DZ2272600)资助.**谭德贵**,男,1984 年生,硕士研究生,主要研究方向为错误定位. E-mail: damly2008@163.com. **陈 林**,男,1979 年生,博士,主要研究方向为软件分析、软件重构. **王子元**,男,1981 年生,博士,主要研究方向为软件测试. **丁 晖**,男,1985 年生,博士,主要研究方向为软件分析. **周毓明**,男,1974 年生,博士,教授,博士生导师,主要研究领域为软件分析、度量. **徐宝文**,男,1961 年生,博士,教授,博士生导师,主要研究领域为程序设计语言、软件工程、并行与网络软件.

将会极其耗时.

鉴于在程序设断点手工排查方法存在的缺点,研究者提出了一系列自动化的错误定位方法.这些自动化方法可分为静态方法与动态方法.静态方法主要是利用程序的依赖关系、类型约束等程序信息来分析程序的可能出错点.动态方法主要是通过运行测试用例得到执行轨迹、覆盖信息等来进行错误定位.近年来我们结合以前在程序分析、度量与测试方面的工作,对错误预测和定位进行了初步的研究[1-7],并在此基础上对基于频谱的动态错误定位方法进行了探讨.这既要对程序进行分析,也需要输入测试用例对程序进行测试,是对以往错误定位工作研究的一个延伸.

由于实现容易,操作简单,基于频谱的错误定位是目前得到很大重视的一类自动化的动态错误定位方法:通过运行测试用例得到程序的各条语句被测试用例覆盖到的信息,然后利用覆盖信息计算出程序中每条语句的含错可疑度,并将相关语句按照含错可疑度降序进行排序以便定位错误语句.在实际的错误定位中,当测试用例个数发生变化时,它们对于语句可疑度的贡献是不一样的.针对这一情况,本文分析了在不同条件下测试用例权重对错误定位效果的影响,提出了增大失败测试用例的边际权重的错误定位方法.实验结果表明,相对于其它方法,本文提出的错误定位方法具有很好的效率.

2 相关工作

经过人们的努力,基于频谱的错误定位方法在不断取得进步.Harrold 等人实证研究了程序频谱与程序行为之间的关系,论证了通过研究运行失败测试用例得到的频谱信息与运行成功测试用例得到的频谱信息之间的差异性可为定位出错语句提供帮助[8].

Renieris 和 Reiss 等按最接近执行原则,用覆盖信息来度量语句的接近程度,根据接近程度的差异来寻找错误语句[9].对于不在差集里的错误,这种方法也可以通过构造程序依赖图来定位错误.

Jones 等人设计实现了基于频谱的错误定位工具 Tarantula,该工具通过计算各语句成功覆盖与错误覆盖的比率来给各个语句着色,根据颜色来区分语句的可疑度[10].实验表明,Tarantula 工具所用的错误定位方法的效率要优于其它很多基于频谱的错误定位方法[11].

为了进一步提高基于频谱的错误定位方法的效率,人们还从多个方面对其进行了改进.

一方面,人们考虑了测试用例的冗余对错误定位算法精度和效率的影响.如果测试用例选择不当,会导致冗余测试用例具有相同或类似的覆盖信息,这些冗余的覆盖信息可能最终降低错误定位的精度和效率.针对这一问题,Yu 等人研究了在不影响测试结果的情况下如何约简测试用例以提高错误定位效率的问题[12].Hao 等人用 SAFL 技术来对冗余覆盖信息进行判断并对冗余测试用例进行约简[13].此后,Hao 等人又提出了三种测试用例约简策略.利用这三种策略,开发者只需要选择一个有代表性的测试输入子集来进行结果检查与错误定位[14].为了弥补自动化错误定位方法灵活性不足的缺点,Hao 还提出了一个基于测试信息的交互式错误定位方法.该方法认为先前交互式步骤产生的信息可以为当前的交互式步骤提供可疑语句的排序[15].

另一方面,人们还考虑了与可疑度有关的因素在计算语句可疑度时的权重.Wong 等人提出,随着更多测试用例的运行,成功测试用例的权重应分阶段逐渐减小[16].他们使用函数 $f(k)$ 来计算如何缩小成功测试用例权重.对 $f(k)$ 做如下定义:

$$f(k)=\begin{cases}k, & k=0,1,2\\ 2+(k-2)\times 0.1, & 3\le k\le 10\\ 2.8+(k-10)\times\alpha, & k\ge 11\end{cases}$$

其中,k 表示语句被成功覆盖的次数,0.1,α 均是权重缩小因子.语句被成功覆盖是指输入成功测试用例时该语句被一次或者多次执行;语句被失败覆盖是指输入失败测试用例时该语句被一次或者多次执行.从 $f(k)$ 的定义中可以看出,成功测试用例的权重是按区间逐步缩小的.计算语句的可疑度时,主要是根据执行该语句的失败测试用例个数的增加会增大语句的可疑度,而执行该语句的成功测试用例个数的增加会减小该语句的可疑度.因此,语句的可疑度为执行该语句的失败测试用例的权重之和减去执行该语句的成功测试用例的权重之和.此后,他们还提出失败测试用例的权重也应分区间逐步减小[17].语句的可疑度为执行该语句的失败测试用例权重之和减去执行该语句的成功测试用例权重之和.实验结果表明,Wong 的上述两种方法均可在一定程度上提高错误定位效率[16-17].

Wong 等人考虑了测试用例权重对错误定位效率的影响,然而他们的方法也面临一些问题:首先,权重的区间不容易选择;其次,权重在各区间上的缩

小因子不能自动调节.

3　增大边际权重算法

为了解决 Wong 等人提出的方法所面临的问题,我们提出了一种改进的基于频谱的错误定位方法.该方法通过放大失败测试用例的边际权重来计算语句的可疑度.即针对某一特定语句,令失败测试用例的权重随着其对该语句覆盖次数的增加而增大.

3.1　算法设计

在利用频谱信息计算语句的可疑度时,执行某语句的失败测试用例和成功测试用例担当了不同的作用:失败测试用例增加该语句的可疑度,而成功测试用例减少相应的可疑度.Wong 等人的方法则启发我们,当测试用例个数变化时,考虑相应的频谱权重能改变可疑度计算的精度,从而提高错误定位效率.不妨从以下几个方面来考察测试用例个数变化时权重对错误定位效率的影响.

当失败测试用例个数相对于正确测试用例个数比较稀缺时,语句被失败测试用例覆盖的概率要比被成功测试用例覆盖的概率低,所以语句增加一个失败测试用例覆盖比语句增加一个成功测试用例覆盖对可疑度的影响更大.为了抵消这种语句正确覆盖与错误覆盖概率不一致时使用相同权重对可疑度的不良影响,只有让覆盖该语句的失败测试用例的权重大于成功测试用例的权重.并且随着覆盖该语句的失败测试用例个数的增加,在剩余的测试用例中失败测试用例的个数就会减少,则其再一次被失败测试用例覆盖的概率就更低.因此,下一个失败测试用例对可疑度的权重会更大.

同理,当失败测试用例个数比成功测试用例个数大得多时,则失败测试用例的权重比成功测试用例的权重小,且成功测试用例的权重逐步增大.

通过以上的分析可知,测试用例的权重应该逐步地增大.Wong 的计算测试用例的权重恰好与我们的分析过程相反.应该把 Wong 算法中第 k 个测试用例的权重与倒数第 k 个测试用例的权重相调换.由于这种调换对测试用例的总权重不产生影响,因此,调换后语句的可疑度并不会变化.

此外,边际增大测试用例权重可以避免区间选择不当的问题.相对于 Wong 的权重缩小因子,采用增大测试用例边际权重的方法有一个权重增大因子.在权重增大因子的选择上,为了能够让权重增大因子适应不同的测试集,我们让权重增大因子随着测试集中成功测试用例与失败测试用例之间比例的变化而变化.采用增大失败测试用例权重而非成功测试用例权重是因为采用增大失败测试用例权重更能体现出每增加一个失败测试用例时语句可疑度的变化.

在用语句的覆盖信息表进行可疑度计算时,首先求出每条语句在 n 个测试用例中的成功覆盖次数以及失败覆盖次数.第 j 条语句在 n 个测试用例中的失败覆盖次数为

$$s_f(j)=\sum_{i=1}^{n}C_{i,j}\times r(i),$$

第 j 条语句在 n 个测试用例中的成功覆盖次数为

$$s_s(j)=\sum_{i=1}^{n}C_{i,j}\times(1-r(i)),$$

在上述两式中,i 表示第 i 个测试用例,j 表示第 j 条语句.当第 i 个测试用例覆盖了第 j 条语句时,$C_{i,j}$ 取 1,否则取 0;当第 i 个测试用例成功覆盖时 $r(i)$ 取 0,失败覆盖时取 1;n 表示测试用例个数.

接着就要计算失败测试用例的权重.首先计算失败测试用例的权重因子.第 j 条语句的失败测试用例的权重增大因子 v 计算公式为

$$v=\begin{cases}0, & \sum_{i=1}^{n}r(i)=0\\ \sum_{i=1}^{n}(1-r(i))\Big/\sum_{i=1}^{n}r(i), & \sum_{i=1}^{n}r(i)\neq 0\end{cases}$$

当语句的错误覆盖次数由 k 增大到 $k+1$ 时,其第 $k+1$ 个失败测试用例的权重 $mw(k+1)$ 的计算公式为

$$mw(k+1)=(k+1)v,$$

其中,$mw(0)=0$.

接着就可以得到覆盖第 j 条语句的失败测试用例总权重的计算公式:

$$f_f(s_f(j))=\sum_{i=0}^{s_f(j)}mw(i).$$

对于每条语句,该语句被错误覆盖的次数越多,则该语句的可疑度越大.与此相反,对于每条语句,该语句被正确覆盖的次数越多,则该语句的可疑度越小.因此,语句的可疑度应该与被错误覆盖次数正相关,与被正确覆盖次数反相关.据此,在对失败测试用例权重进行扩大后,我们提出了一种计算语句可疑度的公式:

RA-1:$f_f(s_f(j))-s_s(j)$.

此外,语句的可疑度也可能与被错误覆盖次数成正比,与被正确覆盖次数成反比.据此,在对失败

测试用例权重进行扩大后，我们提出了另一种计算语句可疑度的公式：

RA-2：$(f_f(s_f(j))+0.001)/(f_f(s_f(j))+s_s(j))$.

在这两个计算公式中，RA-1 可以看作是 Wong 算法的改进.这两个公式都是保证可疑度与语句的错误覆盖次数正相关，与语句的正确覆盖次数负相关.RA-2 之所以引入 0.001 是为了避免 $f_f(s_f(j))$ 为零时 $suspect(j)$ 都为零.

3.2 算法应用举例

本小节通过一个实例来详细介绍如何使用增大边际权重算法来计算可疑度.语句的覆盖信息收集完后将得到一个程序覆盖信息表(如表 1 所示)[18].该表的一行代表一个测试用例的代码覆盖情况；除最后一列外，每列代表一条语句在各个测试用例中被覆盖的情况.如第 i 行 j 列表示第 j 条语句在第 i 个测试用例中的覆盖情况.每次测试时某条语句若覆盖到则用 1 表示，未覆盖到则用 0 表示.最后一列表示各个测试用例成功还是失败，成功用 0 表示，失败用 1 表示.实际进行语句可疑度的计算时，覆盖信息表可用一个二维数组 $a[ROW][Collum]$ 来保存.当 $j<Collum-1$ 时，$a[i][j]$ 表示 t_i 与 s_j 对应的数值，也就是运行第 i 个测试用例时第 j 个语句的覆盖情况.如 $a[3][4]=0$ 表示第 4 条语句在运行第 3 个测试用例时没有覆盖到.当 $j=Collum-1$ 时，$a[i][Collum-1]$ 表示第 i 个测试用例成功与否.$a[i][Collum-1]=0$ 表示程序执行该测试用例产生的结果正确；$a[i][Collum-1]=1$ 表示程序执行该测试用例产生的结果错误.在这个覆盖信息表中，有 5 个失败测试用例，15 个成功测试用例.语句的成功覆盖次数与失败覆盖次数之和 $s_f(j)+s_s(j)<=20$.这是因为当运行某个测试用例时，若语句 j 没有被覆盖到，则无论该测试用例失败覆盖程序还是成功覆盖程序，$s_f(j)$，$s_s(j)$ 都不会增加.所以 $s_f(j)+s_s(j)$ 表示的只是各条语句被覆盖的次数而非程序被覆盖的次数.

表 1 程序覆盖信息表

	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	r
t1	1	1	0	0	0	1	1	0	1	1	0
t2	1	1	1	1	0	0	1	1	1	0	0
t3	0	1	0	0	0	1	0	0	1	1	0
t4	1	0	0	0	1	0	1	1	1	1	1
t5	0	1	1	0	0	0	0	1	1	0	0
t6	1	1	0	1	1	0	1	0	1	1	0
t7	1	0	1	0	0	0	1	1	1	0	0
t8	0	1	0	1	1	0	1	0	1	1	0
t9	1	0	1	0	1	0	1	1	1	1	1
t10	0	1	0	1	0	0	0	1	1	0	0
t11	1	1	1	1	0	0	1	1	1	0	0
t12	1	0	1	0	1	1	0	1	1	1	1
t13	1	1	1	1	0	0	1	0	1	0	0
t14	0	0	1	1	1	1	1	0	1	1	0
t15	1	1	0	0	0	0	1	1	0	1	1
t16	0	0	0	1	1	0	1	1	0	0	0
t17	0	1	1	0	0	0	1	0	0	1	0
t18	1	1	1	0	1	0	0	0	0	1	0
t19	1	0	0	1	1	1	0	1	1	0	1
t20	1	1	1	0	0	1	0	0	0	1	0

使用 RA-1 所给的公式计算表 1 中各条语句的可疑度，各语句依可疑度由高至低的排列顺序为 s8、s1、s5、s10、s9、s7、s6、s3、s4、s2.具体结果如表 2 所示.

表 2 用放大后的语句失败覆盖次数减去语句成功覆盖次数的方法所得到的可疑度

	$s_f(j)$	$s_s(j)$	$suspect(j)$
s1	5	8	37
s2	1	12	−9
s3	2	9	0
s4	1	8	−5
s5	4	5	25
s6	2	4	5
s7	3	10	8
s8	5	6	39
s9	4	11	19
s10	4	8	22

使用 RA-2 所给的公式计算表 1 中各条语句的可疑度，各语句依可疑度由高至低的排列顺序为 s8、s5、s1、s10、s9、s6、s7、s3、s4、s2.具体结果如表 3 所示.

表 3 用放大后的语句失败覆盖次数除以语句成功覆盖次数与失败覆盖次数之和的方法所得到的可疑度

	$s_f(j)$	$s_s(j)$	$suspect(j)$
s1	6	10	0.849
s2	2	13	0.200
s3	2	11	0.500
s4	2	10	0.273
s5	4	8	0.857
s6	2	7	0.692
s7	3	11	0.643
s8	6	8	0.882
s9	4	13	0.732
s10	4	10	0.789

通过对这两种方法的结果进行分析可以看出.通过放大 $s_f(j)$ 的方法来计算可疑度时，$s_f(j)$ 对 $suspect(j)$ 的影响要大于 $s_s(j)$，并且 $s_f(j)$ 的影响程度由所有测试用例成功覆盖次数与失败覆盖次数之商 v 以及语句 j 的失败覆盖次数 $s_f(j)$ 来决定.

4 实 验

为了检验上述算法的效果，本文选用西门子程

序集作为实验对象对算法的效果进行验证①.由于之前的很多基于频谱的错误定位方法使用了西门子程序集.本实验使用西门子程序集是为了更好地与之前的错误定位方法进行比较.程序的特殊性会对实验结果产生影响,比如,当程序的分支很少时,使用基于频谱的错误定位方法进行错误定位会造成很多语句的覆盖情况一样,这将导致很多语句的可疑度一样.此时,基于频谱的错误定位方法效果将会不明显.西门子程序集共含有 7 个程序,每个程序均存在一个正确版本以及若干个不同的错误版本,各个程序的基本信息如表 4 所示.

表 4　西门子程序集基本信息

程序	描述	错误版本数	源代码行数	可执行代码行数
print_tokens	词法分析程序	7	565	204
print_tokens2	词法分析程序	10	510	202
replace	模式代换	32	563	274
schedule	优先级调度	9	412	166
schedule2	优先级调度	10	307	146
tcas	高度区分	41	173	74
tot_info	信息估量	23	406	139

首先使用 gcc3.3.1 对正确版本以及各个错误版本的程序进行编译,之后使用 gcov 收集代码的覆盖信息,最后利用不同的错误定位算法对收集到的覆盖信息进行处理,从而得到错误定位结果.在实验中,我们将本文提出的算法与 Wong 等人提出的方法以及 Jones 等人提出的 Tarantula 方法进行比较.

4.1　实验设计

本实验对西门子程序集中的 7 个程序的各个错误版本都进行了实验.当测试用例运行时,gcov 记录测试用例运行时的语句覆盖情况.程序中永远不会执行的语句属于非执行语句.当测试用例运行结束之后,gcov 将会生成记录覆盖信息的文件.

把输入测试用例后各个错误版本的输出结果与正确版本的输出结果进行比较以确定错误版本每个测试用例是成功还是失败.若输出结果一样则认为输入的测试用例是成功测试用例,否则认为是失败测试用例.然后对可疑代码的覆盖信息生成一个矩阵,接着把测试结果附于矩阵的最后一列就得到一个覆盖信息表,最后对得到的覆盖信息表进行计算以得到各条语句的可疑度.

当对各条语句的可疑度进行排序时,如果几条语句的可疑度一样,则他们的排序顺序将根据它们在代码中的行号来决定:行号小的则其可疑度排在前面;行号大的则其可疑度排在后面.因为不同可疑语句的行号在与其可疑度相同的语句的行号排序中可能排在任何的位置,即不同可疑语句的位置是随机的.有的可疑语句可能排在前面,有的可疑语句可能排在后面,有的可疑语句可能排在中间.所以在可疑语句比较多时,即使根据行号来排序也还是会得到很好的随机效果.本实验并不比较各种算法的最好情况与最坏情况.使用最好最坏情况的好处是可以根据最好最坏情况的差别大小,来了解算法的区分度.若最好最坏情况的差别较大,则说明错误语句与很多正确语句的可疑度一样,若最好最坏情况的差别较小则说明较少的正确语句与错误语句的可疑度一样.然而使用最好与最坏情况进行比较只在理论上可行,在实际中是不可行的.因为在未知哪条语句存在错误的情况下,确定存在错误的语句在可疑度一样的几条语句中排在最前还是最后是不可能的.

由于代码的错误是未知的,当使用本实验的排序方法时,可疑度一样的错误语句与正确语句的排序先后应是随机的.即错误语句可能位于最前,也可能位于最后,还可能位于其他的位置.使用本实验的排序方法是为了使得结果更好地接近平均情况.

在对语句的可疑度进行排序后,就得出每个算法在进行错误定位时所需查找的执行语句个数 a(a 表示有 $a-1$ 条正确语句的可疑度排在错误语句的可疑度前面).假设该版本的总执行语句数为 b,则通过计算 $(a/b)\times 100\%$ 得出进行错误定位时所需查找的执行语句数占总执行语句数的百分比,即排查效率.把 0～100%进行离散化得到 20 段区间.每个区间的百分数表示排查效率 $(a/b)\times 100\%$ 在该区间上的个数占总个数的百分比.很明显,对于各个程序的各个版本来说,$(a/b)\times 100\%$ 越小,则所使用的算法在进行错误定位时效率就越高.

4.2　实验结果

本实验使用西门子程序集提供的测试用例来进行测试.为了更好地比较各个算法的效率,在测试用例的选择上,我们从 testplans-bigcov,testplans-bigrand提供的一系列测试用例集里各选取一个测试用例集作为输入.此外,我们还选择了西门子提供的所有测试用例集 universe 作为输入.testplans-bigcov 里的测试集是从测试总集中选出来的一个子

① The Siemens Suite. January 2007. http://www-static.cc.gatech.edu/aristotle/Tools/subjects

集,这个子集能够覆盖所有的分支路径,但覆盖每个分支路径上的测试用例个数是随机的. testplans-bigrand 是一个测试用例个数与 testplans-bigcov 相等但测试用例属于随机选取的测试用例集. 对语句可疑度进行排序时,给不被覆盖的语句的可疑度赋予一个极小的值,以使被覆盖语句的可疑度大于不被覆盖的语句可疑度. 各个算法的实验结果如图 1～图 3 所示. 其中,我们将 Wong 等人在文献[16]中所提出的缩小语句成功测试用例权重的算法记为 Wong-3,将 Wong 等人在文献[17]中提出的既缩小成功测试用例的权重也缩小失败测试用例的权重的算法记为 Wong-4,而 RA-1、RA-2 则分别表示本文所提出的两种可疑度计算公式.

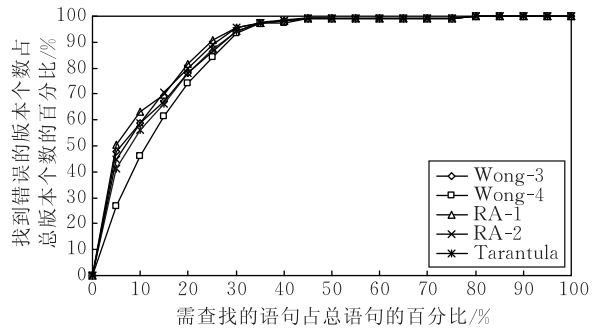

图 1 5 个算法在 testplans-bigrand 测试用例集中的效率比较图

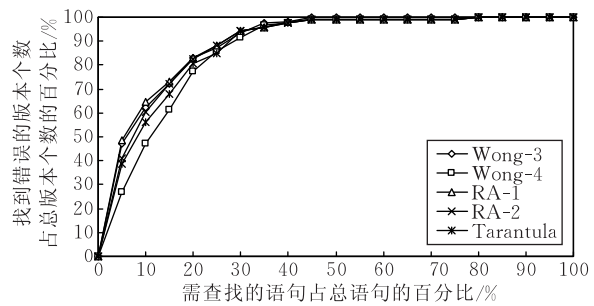

图 2 5 个算法在 testplans-bigcov 测试用例集中的效率比较图

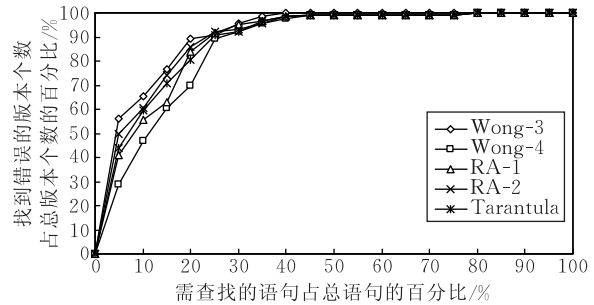

图 3 5 个算法在 universe 测试用例集中的效率比较图

根据图 1 所示结果,RA-1 计算公式得到的效率最高,Wong-3 次之,之后是 RA-2,Tarantula,Wong-4. 根据图 2 所示结果,使用 RA-1 计算公式得到的效率最高,Wong-3 次之,RA-2,Tarantula 以及 Wong-4 的两个算法相对较差. 从图 1、图 2 中我们可以发现一个异常的情况,根据图 3 所示结果,Wong-3 效果最好,RA-2 次之,之后依次是 Tarantula,RA-1,Wong-4.

西门子实验的结果表明,与缩小覆盖语句的失败测试用例权重或者同时缩小覆盖语句成功测试用例和失败测试用例权重的方法相比,通过放大覆盖语句的失败测试用例权重的方法对错误定位效果有一定的促进作用. 当用例数增加,并且有较多冗余时,我们的方法效率没有提高.

4.3 实验讨论

本文算法的有效性是通过西门子实验来验证的. 由于西门子程序集的各个程序只有几百行的代码,因此我们的技术在更大的程序里进行错误定位是否也会得到很好的效果还有待进一步的研究.

在基于频谱的错误定位中,采用的都是对覆盖信息进行处理得出语句的可疑度. 如果失败测试用例的个数太少,很容易造成很多语句具有相同的失败覆盖次数与成功覆盖次数. 这将会导致错误语句与很多正确语句具有相同的可疑度,并最终降低各语句可疑度的区分度.

因为本文算法的核心思想是增大失败测试用例的权重,所以失败测试用例对实验结果的影响很大. 在这种情况下,失败测试用例权重因子的选择就显得至关重要. 由于成功测试用例对可疑度也是有影响的,在考虑失败测试用例的权重因子时,成功测试用例的影响也要进行相应的考虑. 有鉴于此,我们选择的权重因子是成功测试用例个数与失败测试用例个数的比率. 这说明权重因子随着成功测试用例与失败测试用例比率的增大而增大,随着成功测试用例与失败测试用例比率的减小而减小. 这与我们在第三节中对成功测试用例与失败测试用例的出现概率对权重影响的分析是相一致的. 这样还可以使得权重因子随着测试用例集的变化而自动变化.

在使用覆盖信息来计算语句的可疑度时,我们不仅要知道测试用例集中成功测试用例与失败测试用例的个数,还要知道每条语句被成功覆盖的次数与被失败覆盖的次数. 并且每条语句被成功覆盖的次数与被失败覆盖的次数未必与测试用例集中成功测试用例与失败测试用例的个数相等. 在多错误版本中,不同的失败测试用例可能覆盖不同的错误语句. 这时即使失败测试用例的个数比较多,覆盖每条

错误语句的失败测试用例个数也可能会很少.而我们的权重因子是根据测试用例集中成功测试用例个数与失败测试用例个数的比率得出的,因此,在多错误版本里我们的权重因子可能会失效.

5 结　论

本文对基于频谱的错误定位技术进行了研究,将失败测试用例的边际权重引入到可疑度计算的过程中,即针对某一特定语句,令失败测试用例的权重随着其对该语句覆盖次数的增加而增大.在这一理论的基础上,本文提出了两种新的错误定位计算公式 RA-1 和 RA-2.实验结果表明,与缩小覆盖语句的失败测试用例权重或者同时缩小覆盖语句成功测试用例和失败测试用例权重的方法相比,通过放大覆盖语句的失败测试用例权重的方法对错误定位效果有一定的促进作用.

在未来的工作中,我们还将致力于对本文提出的算法进行改进.一方面,在计算边际权重时,有必要考虑其它的权重因子,并对其效果进行进一步的研究和检验.另一方面,通过实验我们发现测试用例数量的增多对查找可疑语句的效率可能会有反作用,因此,我们试图通过对测试用例进行约简来提高查找可疑语句的效率.

参 考 文 献

[1] Zhou Y, Xu B, Leung H. On the ability of complexity metrics to predict fault-prone classes in object-oriented systems. Journal of Systems and Software, 2010, 83(4): 660-674

[2] Zhou Y, Leung H. Empirical analysis of object-oriented design metrics for predicting high and low severity faults. IEEE Transactions on Software Engineering, 2006, 32(10): 771-789

[3] Chen Z, Xu B, Nie C. A detectability analysis of fault classes for Boolean specifications//Proceedings of the 2008 ACM Symposium on Applied Computing. Fortaleza, Ceara, Brazil, 2008: 826-830

[4] Zhang X, Xu B, Chen Z, Nie C, Li L. An empirical evaluation of test suite reduction for Boolean specification-based testing//Proceedings of the 2008 the 8th International Conference on Quality Software. Oxford, UK, 2008: 270-275

[5] Chen Z, Chen T Y, Xu B. A revisit of fault class hierarchies in general Boolean specifications. ACM Transactions on Software Engineering and Methodology, accepted, to appear

[6] Nie Changhai, Leung Hareton. The minimal failure-causing schema of combinatorial testing. ACM Transactions on Software Engineering and Methodology, accepted, to appear

[7] Nie Changhai, Leung H K N, Xu Baowen. A survey of combinatorial testing. ACM Computing Survey, accepted, to appear

[8] Harrold M J, Rothermel G, Wu R, Yi L. An empirical investigation of program spectra//Proceedings of the ACM SIGPLAN/SIGSOFT Workshop Program Analysis for Software Tools and Eng (PASTE'98). Montreal, Quebec, Canada, 1998: 83-90

[9] Renieris M, Reiss S P. Fault localization with nearest neighbor queries//Proceedings of the 18th IEEE International Conference on Automated Software Engineering. Montreal, Canada, 2003: 30-39

[10] Jones J A, Harrold M J, Stasko J. Visualization of test information to assist fault localization//Proceedings of the 24th International Conference on Software Engineering. Orlando, FL, USA, 2002: 467-477

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

[12] Yu Y, Jones J A, Harrold M J. An empirical study of the effects of test-suite reduction on fault localization//Proceedings of the 30th International Conference on Software Engineering. Leipzig, Germany, 2008: 201-210

[13] Hao D, Zhang L, Pan Y, Mei H. On similarity-awareness in testing-based fault localization. Automated Software Engineering, 2008, 15(2): 207-249

[14] Hao D, Xie T, Zhang L, Wang X, Sun J, Mei H. Test input reduction for result inspection to facilitate fault localization. Automated Software Engineering, 2010, 17(1): 5-31

[15] Hao D, Zhang L, Mei H, Sun J. Towards interactive fault localization using test information//Proceedings of the 13th Asia-Pacific Software Engineering Conference (APSEC 2006). Bangalore, India, 2006: 277-284

[16] Wong W E, Qi Y, Zhao L, Cai K Y. Effective fault location using code coverage//Proceedings of the 31st IEEE Computer Software and Applications Conference. Beijing, China, 2007: 449-456

[17] Wong W E, Debroy V, Choi B. A family of code coverage-based heuristics for effective fault location. The Journal of Systems and Software, 2010, 13(3): 188-208

[18] Wong W E, Wei T, Qi Y, Zhao L. A crosstab-based statistical method for effective fault localization//Proceedings of the 2008 International Conference on Software Testing, Verification, and Validation. Lillehammer, Norway, 2008: 42-51

TAN De-Gui, born in 1984, M. S. candidate. His research interests focus on fault localization.

CHEN Lin, born in 1979, Ph. D.. His research interests include software analysis, software refactoring.

WANG Zi-Yuan, born in 1982, Ph. D.. His research interests focus on software testing.

DING Hui, born in 1985, Ph. D.. His research interests focus on software analysis.

ZHOU Yu-Ming, born in 1974, Ph. D., professor, Ph. D. supervisor. His research interests include empirical software engineering, especially on software metrics, software evolution, and program understanding and analysis.

XU Bao-Wen, born in 1961, Ph. D., professor, Ph. D. supervisor. His research interests include teaching and research on programming language, software engineering, parallel and network software, acquisition technique on knowledge and information etc.

Background

This paper is supported by the National Natural Science Foundation of China under grant Nos. 90818027, 60873050, the National High Technology Research and Development Program (863 Program) of China under grant No. 2009AA01Z147, the National Basic Research Program (973 Program) of China under grant No. 2009CB320703, the Key Laboratory Funding of Science and Technology Commission of Shanghai Municipality (09DZ2272600). These projects mainly research on the following fields: Program analysis, Software Testing. This paper focuses on the research of the Spectra-Based Fault Localization by Increasing Marginal Weight to improve the effectiveness of fault localization. That means that as the number of the failed test case increases, the weight of the failed test case also increases. The experimental result shows that increasing the marginal weight of the covered statement's failed test case can promote the fault localization efficiency.