文章编号:1672-3961(2012)06-0019-06 DOI:10.6040/j.issn.1672-3961.2012.06.004

一种基于聚类算法的缺陷语句定位技术

蔡虹,黄霞
(淮海工学院计算机学院,江苏 连云港 222005)

摘要:由于数据集的不一致,已有的基于频谱覆盖的缺陷定位方法之间的比较并不全面。本研究实现了现有的28种基于频谱覆盖的缺陷定位方法,并在同一数据集上加以比较。提出一种新的基于*k*-means聚类算法的缺陷定位技术,利用现有的多种方法计算出特征值,对数据集进行聚类并排序,给出一个新的语句的可疑度序列。实验结果表明:该方法可以取得比较好的结果,能够捕获到个别算法的优越性,较为有效地对程序中的缺陷进行定位。

关键词:缺陷定位;频谱覆盖;聚类;程序分析;可疑度
中图分类号:TP311 **文献标志码**:A

A spectrum-based fault localization method based on clustering algorithm

CAI Hong, HUANG Xia
(School of Computer Engineering, Huaihai Institute of Technology, Lianyungang 222005, China)

Abstract: The comparison of the existing spectrum-based fault localization methods is not comprehensive due to the difference of data set, and there is no single method best for all situations so far. Therefore, the existing 28 spectrum-based fault localization methods were implemented to evaluate spectrum-based methods in same data set. A new spectrum-based fault localization method, which utilized *k*-means algorithm, was proposed to obtain a new suspicious ranking of statements so as to improve the effectiveness of fault localization. The effectiveness and performance of this method were confirmed by means of the designed experiment, and the statements with accepted high suspiciousness in program was captured.

Key words: fault localization; spectrum-based coverage; clustering; program analysis; suspiciousness

0 引言

软件调试是一项极其艰巨复杂的任务。研究表明,改正软件中错误与漏洞的开销占到整个工程的80%[1],而找到错误所在是调试中最难的部分之一[2]。目前,自动化调试技术已经成为一种趋势,一个好的自动化缺陷定位技术对帮助调试者减少工作量具有重要意义。自动化缺陷定位方法主要通过分析成功测试用例运行和失败测试用例运行,以及二者之间在某种程序信息上的差异,并通过某种映射机制最终找到差异所在的最低抽象层次位置,这种最低抽象层次的差异就是缺陷所在的程序位置。程序位置粒度有大有小,典型的有语句、基本块、函数、分支、路径等。当前软件自动化调试技术较为活跃的研究集中在缺陷定位技术,其中频谱覆盖的方法是缺陷定位方法中最为重要的一类。基于程序谱的缺陷定位技术(spectrum-based fault localization, SFL)是目前最先进的缺陷定位方法之一,SFL方法使用不同的度量准则来评估一个程序位置包含错误的可能性大小,并且按包含错误的可能性大小对程序位置进行可疑度的排序,可疑性值越高就代表程序的该位置有缺陷的可能性越高。因此,度量方法的设计是SFL方法的关键问题之一。此外,SFL通过对单元覆盖和测试结果的分析来确定单元的可疑度,如果测试设计的合理,那么测试结果可以在很大

收稿日期:2012-03-20
基金项目:江苏省普通高校科研成果产业化项目(JH08-35)
作者简介:蔡虹(1977 -),女,江西南昌人,讲师,硕士,主要研究方向为软件分析技术. E-mail:rainbow521@gmail.com

程度上反映程序自身存在的问题,如何利用此信息成为基于频谱覆盖的缺陷定位方法的另一个关键所在。目前已有的 SFL 使用多种不同的方法来计算程序单元的可疑度[3-10],典型的 SFL 方法有 JARCCARD[11]、TARANTULA[12]、OCHIAI[13]等。众多方案的提出者均在自己的文章中和其他方法进行了比较,但是由于数据集的不一致,以及其他因素,导致比较结果可能不全面。此外,据相关研究表明,目前没有任何一个单一的 SFL 度量公式能够适用于所有情况[14]。因此,本研究实现了现有的 28 种基于频谱覆盖的缺陷定位方法,并在同一数据集上加以比较。并在此基础上提出了一种新的基于 *k*-means[15-16]聚类算法的缺陷定位技术。该方法利用现有的多种方法计算出特征值,对数据集进行聚类,并对其进行排序,给出一个新的语句的可疑度序列。这样就可以利用已有的多种方法的优势来捕捉那些具有公认的具有高可疑度的程序位置,从而提高错误定位的准确性。

1 基于频谱的缺陷定位方法

程序行为特征,通常也被称为程序频谱,是程序执行特征的统计信息。REPS T 等人在调试解千年虫的问题时,首次提出程序频谱的概念[17]。为了揭示程序行为特征与程序错误之间的关系,HARROLD M 等人对此做了大量实验[18]。实验结果表明:程序出现异常的行为特征未必意味着代码中存在缺陷,但是错误的程序运行往往会表现出异常的行为特征。在基于行为特征对比的方法中,一般通过假设失败的测试执行会表现出异常的程序行为特征,所以成功执行和失败执行中的行为特征的差异可以用来指导错误定位。SFL 方法采用统计学的方法对程序运行信息进行分析,找出那些与失效尽量相关的位置或者提取出某种模型作为参照来定位缺陷。SFL 是一个动态程序分析方法,它对各程序位置与错误相关性进行计算。由于简单和统计的特征,SFL 独立于任何系统模型并且花销较低。SFL 利用的程序信息是来自成功运行和失败运行的程序频谱。成功的运行就是程序的输出符合预期,失败的运行就是程序的输出不符合预期。程序谱在程序运行时被收集,其中最典型的就是记录着各程序位置的覆盖情况。基于频谱的缺陷定位技术是基于统计方法中最先进的缺陷定位方法之一,其工作流程如下:首先是根据收集信息类型的需要,对源代码进行插桩并执行程序,收集执行信息。然后再判断每个测试用例的执行结果。接着解析执行信息,得到执行行为特征。再根据给定的模型,建模程序实体的怀疑度,即可能出错的程度。最后,以程序实体排名的方式给出定位结果,将各程序实体按照可疑度大小从大到小排列,供开发人员参考。按照使用的行为特征信息的种类和策略,程序实体按照粒度的大小总体可以分为 5 类:基于语句(statement)或基本块的(basic block)、基于谓词的(predicate)、基于方法的(method)、基于定义使用对或信息流的(definition-use pair)的方法。

2 基于聚类算法的缺陷语句定位方法

2.1 覆盖信息的收集

程序覆盖信息提供了程序运行的动态信息,具体说覆盖信息是指一个执行单元在程序的执行时被每个测试用例的覆盖情况。其中执行单元可以有多种定义,包括基本块、分支、语句、函数、类等等。另外还需收集每个测试用例执行的结果信息。本研究的覆盖信息指每个语句被每个测试用例的覆盖情况,以及这个测试用例的执行结果。收集的覆盖信息矩阵如表 1 所示。

表 1 覆盖信息矩阵

Table 1 Coverage information matrix

语句 &用例	T_1	T_2	T_3	T_4	T_5	a_{ef}	a_{ep}	a_{nf}	a_{np}
Statement_1	1	0	1	1	1	1	3	1	0
Statement_2	1	1	0	1	0	2	1	0	2
Statement_3	0	1	0	1	1	1	2	1	1
…	…	…	…	…	…	—	—	—	—
TestResult	1	1	0	0	0	—	—	—	—

其中 Statement_n 表示一个程序的一条语句,T_n 表示一个测试用例。$\langle \text{Statement}_1, T_1 \rangle = 1$ 表示在执行测试用例 T_1 时,语句 Statement_1 被覆盖;$\langle \text{Statement}_1, T_1 \rangle = 0$ 则代表未被覆盖到。TestResult 表示测试用例执行的结果,$\langle \text{TestResult}, T_1 \rangle = 0$ 表示测试用例 T_1 执行成功,$\langle \text{TestResult}, T_1 \rangle = 1$ 表示测试用例 T_1 执行失败。对于每个语句利用 4 个值$\langle a_{ef}, a_{ep}, a_{nf}, a_{np} \rangle$来表示执行的状态。下标中的第一字母表示该条语句是否被执行,其中 e 表示被执行到(executed),n 表示未被执行到(not executed)。下标中的第二字母指某个特定的测试用例的执行结果,f 表示测试用例失败(failed),p 表示测试用例成功(passed),一条语句相对于一个测试用例的一次执行,a_{ef}, a_{ep}, a_{nf}, a_{np}4 个值中,有且仅有一个会被赋值为 1,如 $a_{ef} = 1$ 表示当前语句被该测试用例所覆盖,而且这个测试用例执行结果为失败。表 1 中

这 4 个值是表示多个测试用例执行后信息的累加,如〈Statement_1, a_{ep}〉=3 代表在 T_1 到 T_5 的执行中,语句 Statement_1 被执行成功的测试用例覆盖了 3 次。想收集到此信息要对程序进行插桩,本方法利用开源工具 Codecover[19] 实现了对源程序的插桩,并收集相关信息,提取覆盖矩阵。而测试用例的输入则由 Java 的单元测试工具 Junit 来完成。

2.2　可疑度计算方法

基于频谱覆盖的方法主要利用收集到的〈a_{ef}, a_{ep}, a_{nf}, a_{np}〉进行计算,SFL 有很多不同计算程序位置可疑值的公式,典型的 SFL 计算方法有 Ochiai[5]、Jarccard[6]、Tarantula[7] 等,它们的计算公式分别如下:

Ochiai 的公式

$$S_{\text{Ochiai}}=\frac{a_{ef}}{\sqrt{(a_{ef}+a_{nf})(a_{ef}+a_{ep})}},$$

Jarccard 的公式

$$S_{\text{Ochiai}}=\frac{a_{ef}}{\sqrt{(a_{ef}+a_{nf})(a_{ef}+a_{ep})}},$$

Tarantula 的公式

$$S_{\text{Tarantula}}=\frac{\dfrac{a_{ef}}{a_{ef}+a_{nf}}}{\dfrac{a_{ef}}{a_{ef}+a_{nf}}+\dfrac{a_{ep}}{a_{ep}+a_{np}}}。$$

由于受篇幅影响这里不再把 28 种频谱计算公式一一列出,具体实现的 28 种算法请参见实验分析。此外,由于这 28 种方法度量所需的基本信息是相同的,不同的部分主要在于可疑度的计算方法,而 SFL 方法的复杂度主要在于基本信息的收集。因此,在实现 28 种基于频谱覆盖的缺陷定位方法时,只需要完成一次的基本信息收集,而实现 28 种度量公式的计算和最后的聚类方法实现并不会带来过高的算法复杂度。

2.3　聚类算法

聚类就是按照事物间的相似性进行区分和分类的过程,在这一过程中没有指导者,因此是一种无监督的分类。聚类分析则是用数学方法研究和处理所给定对象的分类。聚类方法首先提取样本特征,把原始样本作为输入,并决定使用哪些特征来描述样本的本质性质和结构。完成特征提取后,通过执行聚类算法,获得聚类谱系图。聚类的输入是一个样本矩阵,一个样本通过映射成为特征变量空间中的点。聚类算法的目的就是获得能够反映 N 维空间中这些样本点的最本质的"簇"的性质。聚类方法最后需要选取合适的分类阀值,选定阀值以后,就能够从聚类谱系图上直接看出分类方案。基于频谱覆盖的算法主要是通过多个测试用例执行后,收集程序被每个测试用例覆盖的信息以及测试用例执行的结果,即〈a_{ef}, a_{ep}, a_{nf}, a_{np}〉,目的都是计算语句出错的可能性,所以对正确的语句可疑度计算,应相差不多,但是由于侧重角度的不同,同样的错误可疑度值在不同的计算公式下,可能会有比较大的差异,而这些差异可以更好进行缺陷定位。

本研究则在实现已有的 28 种频谱覆盖方法的基础上,将其作为特征值对语句进行聚类,认为语句节点最多的类为正确的类,计算每个节点到正确类中心节点的距离,即为可疑度,以此排序并输出给用户。具体流程如下:

(1)处理 28 种基于频谱覆盖方法的结果:对于一条语句 S_n,每种基于频谱覆盖方法计算出的可疑度值 M_1,$M_2\cdots M_{28}$ 均为 S_n 的属性值,将 28 种方法的可疑度值连同语句,一起存入 CSV 文件中。

(2)聚类是以语句为节点,各方法的可疑度值为属性,采用 k-means 方法聚类。

(3)找出最大子类就是统计每个子类节点的数目,节点最多的类,即为主类。

(4)找出主类的中心点。

(5)计算各节点到主类中心节点的欧式距离。

(6)按照距离值从大到小排序,并显示结果。

由于工程的代码量可能会比较大,出于时间效率的因素,本研究采用的聚类算法为 k-means 算法。选择所有频谱覆盖方法计算的可疑度值为属性值。用欧氏距离计算,并排序。

k-means 算法的实现过程如下:

Input 为点集 P,

Output 为 k-means 问题的解值。

Step 1　从 P 中随机选取 k 个点作为初始的中心点。

Step 2　对于 P 中每个点,计算该点与每个中心点的距离,将该点分配给最近的中心点。

Step 3　重新计算每个新聚类的均值,即质心。

Step 4　若聚类的中心不再变化,则返回划分结果,否则转 Setp2。

在结构性聚类方法中,选择测量的距离是重要的一步。在聚类方法中,若使用 n 个指标特征变量来描述样本,那么就可以把每个样本点看作 n 维空间的一个点,进而使用某种距离来表示样本点之间的相似性,距离较近的样本点性质较相似,距离较远

的样本点差异较大。不同的距离度量可用来检测不同结构的数据子集。常见的距离函数如表2所示。在本研究中选用较为常用的Euclid距离来度量样本的相似性。

表2 常见距离函数
Table 2 Some popular distance functions

距离名称	距离函数	特点功能
Minkowaki	$D_q(X,Y)=\{\sum_i \lvert X_i-Y_i\rvert^q\}^{\frac{1}{q}}$	对应 $1\leqslant p\leqslant\infty$ 为族距离测度
Euclid	$D_2(X,Y)=\{\sum_i \lvert X_i-Y_i\rvert^2\}^{\frac{1}{2}}$	对应 $P=2$ 的 Min 距离
Hamming	$D_1(X,Y)=\{\sum_i \lvert X_i-Y_i\rvert\}$	对应 $p=1$ 的 Min 距离
Maximum	$D(X,Y)=\max_{i=1,2\cdots n} \lvert X_i-Y_i\rvert$	对应 $p=\infty$ 的 Min 的距离
Mahalanobis	$D(X,Y)=(X-Y)^{\mathrm{T}}*\sum^{-1}*(X-Y)$	$\sum$为正定矩阵总体分布协差估计量

3 实验分析

目前提出的部分基于频谱覆盖的缺陷定位方法对于测试用例有特殊要求,如Overlap方法中要求a_{ef}, a_{nf}, a_{ep}均不可为0,否则依照此公式得出的值没有意义。而且提出的算法公式本身往往都有一些局限性,要求测试用例的覆盖面达到特定程度才能使用。由于本研究实现目前28种算法的目的之一是比较各算法在同一个测试集上的效率高低,所以不去考虑算法测试用例必须满足的前提,为方便比较,需要对这些算法做一些特殊处理。在使用“Ample”、“AnderbergMethod”、“Dice”、“HarmonicMean”、“JaccardMethod”、“Kulczynski”、“M2”、“Ochiai”、“Overlat”、“RussellRao”、“RogersTanimoto”、“Rogot”、“SDice”、“SimpleMatching”、“Sokal”、“Tarantula”、“Zoltar”等方法计算时,如出现分母为0的情况,则将计算结果赋值为9.99(按照以上方法计算出的结果均小于9.99)当按照“Goodman”、“Fleiss”、“GeometicMean”、“ArithmeticMean”、“Cohen”、“Scott”等方法计算时,如出现分母为0且分子大于等于0的情况,则将计算结果赋值为9.99;如出现分母为0且分子小于0的情况,则将计算结果赋值为-9.99(按照以上方法计算出的结果在-9.99到9.99之间)。对于“Hamann”方法,如出现分母为0且分子大于等于0的情况,则将计算结果赋值为99.99;如出现分母为0且分子小于0的情况,则将计算结果赋值为-99.99(按照以上方法计算出的结果在-99.99到99.99之间)。

本程序的实验对象是一个插件开发工程,是实现基于聚类的缺陷定位方法的子工程。共包含35个类、1 800行代码。该子工程需要的语句被多个测试用例的执行信息已由经过修改过的Codecover收集完成,并存入数据库。本程序共植入9个错误,表3列出了错误的具体信息,植入的错误分为7种,部分类型植入两个错误。测试代码共包括20个测试类,49个测试用例,其中成功的测试用例32个,失败17个,没有error产生。具体的实验结果如表4所示。

表3 植入错误列表
Table 3 The detail of faults injection

序号	出错语句内容	错误种类	所在位置(*.java:line)	正确版本
S_1	answer=answer1+answer2;	赋值错误	Ample:15	answer=answer1-answer2;
S_2	return 8.88;	返回值错	Dice:15	return 9.99;
S_3	if(aef*anp<anf*aep){	分支条件、判断错误	Cohen:11	if(aef*anp>=anf*aep){
S_4	if(aef+anp-aep-anf<0){	分支条件、判断错误	Hamann:11	if(aef+anp-aep-anf>=0){
S_5	answer=2*aef/(aef+anf+aep);	类型转换错误	Dice:14	answer=(double)2*aef/(aef+anf+aep);
S_6	for(int i=0;i<=a/2;i++){	循环边界值错误	KMeansClustering:111	for(int i=0;i<=a;i++){
S_7	FilePath+="s";	文件读写异常	FileIO: 32	去掉
S_8	return null;	空指针异常	MethodsMatrix:95	return this.methods;
S_9	methods[29]=0;	数组越界	MethodsMatrix:103	去掉

表4是植入的9种错误在每种方法中的排序情况,以及方法的一些附加信息。

其中,S_n表示错误语句序列,NA表示不能用此算法计算的节点数量,“—”表示本条语句无法用对应方法计算。如〈Ample,S_1〉值为17,表示经过Ample方法排序后,错误语句S_1被排到第17位,〈Am-

berg,NA〉= 522 表示本程序中有 522 条语句用 Amberg 无法计算(会出现除数为 0 或其他异常)。〈Overlat,S_2〉值为"—"表示错误语句 S_2 用 Overlat 方法不可计算。

表 4　错误定位结果比较

Table 4　Comparison results of various methods

方法 &错误语句	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	NA
Ample	17	47	49	50	5	7	12	1	2	—
Amberg	16	21	23	24	5	7	12	1	2	522
Arithmetic	16	21	23	24	5	7	12	1	2	—
Cohen	16	21	23	24	5	7	12	1	2	—
Dice	16	21	23	24	5	7	12	1	2	—
Euclid	16	19	21	22	5	7	12	1	2	—
Fleiss	16	21	23	24	5	7	12	1	2	—
Geometic	21	15	17	18	5	7	12	1	2	522
Goodman	16	21	23	24	5	7	12	1	2	—
Hamann	16	19	21	22	5	7	12	1	2	—
HammingEtc	16	19	21	22	5	7	12	1	2	—
Harmonic	16	23	17	18	5	7	12	1	2	—
Jaccard	22	21	23	24	5	7	12	1	2	—
Kulczynski	14	15	17	18	5	7	12	1	2	522
M2	16	21	23	24	5	7	12	1	2	—
Ochiai	16	21	23	24	5	7	12	1	2	522
Ochiai2	16	21	23	24	5	7	12	1	2	522
Overlap	2	—	—	—	—	—	—	—	—	536
RogTanimoto	16	19	21	22	5	7	12	1	2	—
Rogot	21	15	17	18	5	7	12	1	2	522
RussellRao	6	23	25	28	9	11	18	1	2	—
Scott	16	21	23	24	5	7	12	1	2	—
SDice	6	25	27	28	11	13	18	1	2	—
SimMatching	16	19	21	22	5	7	12	1	2	—
Sokal	16	19	21	22	5	7	12	1	2	—
Tarantula	21	3	9	12	5	10	17	1	2	522
Wong	16	19	21	22	5	7	12	1	2	—
Zoltar	21	25	17	18	5	7	12	1	2	594
Clustering	6	21	21	22	5	7	12	1	2	—

从表中可以看出基于聚类的方法总体上可以取得比较好的结果,能较为有效地对程序中的缺陷进行定位。此外,聚类方法可以屏蔽掉属性值方法存在的某些不稳定因素,进而更多的利用各种方法对于缺陷定位的优势之处。比如本试验中有部分缺陷定位方法会有除数为 0 的情况,即使当这个比例很大的时候,聚类算法仍然不受影响,可以输出较好的结果。另外,从表中也可以看到,如果一个语句被大多数算法公认为可疑度最高,那么他在聚类算法中也会有高可疑度。例如,错误语句 6、错误语句 8、错误语句 9 和错误语句 5 在绝大部分的错误定位方法中排名都较为靠前,那么在聚类方法中的排名也较为靠前。即聚类算法能获得众算法公认的可疑度高的语句,另外由于聚类算法最后的可疑度排序时依靠距离的,所以如果某个方法在某些语句上有特别的优势,聚类算法虽不能因此将语句置为可疑度最高,但也能大大提高此语句的可疑度。例如,对于错误语句 1、错误语句 2,28 种方法中个别方法在错误定位中较有优势,则在聚类方法中的错误排名也相对总体获得了较大的提升。即聚类算法可以捕获到个别算法的优越性。

4　结论

当前,缺陷定位方法有很多种,基于频谱覆盖的方法是其中比较重要的一类,但由于数据集的不一致,已有的基于频谱覆盖的缺陷定位方法之间的比较并不全面。本研究在 Eclipse 环境下利用 JAVA 语言实现了现有的 28 基于频谱覆盖方法,并在同一数据集上进行比较。此外,本研究还提出了一种新的基 *k*-means 聚类算法的缺陷定位技术,该方法以每个语句为节点,利用现有的多种方法计算出特征值,对数据集进行聚类,并采取一定算法排序,给出一个新的语句的可疑度序列。实验结果表明,该方法可以取得比较好的结果,能较为有效地对程序中的缺陷进行定位,并可以捕获到个别算法的优越性。基于聚类的算法有一定的优势,相信通过较大规模完备的测试,可以验证算法的有效性,并且对比较其他基于频谱覆盖的方法也会有很大的帮助。

参考文献:

[1] COLLOFELLO J, WOODFIELD S. Evaluating the effectiveness of reliability-assurance techniques [J]. Journal of Systems and Software, 1989, 9(3):191-195.

[2] VESSEY I. Expertise in debugging computer programs [J]. International Journal of Man-Machine Studies: A process analysis, 1985, 23(5):459-494.

[3] HAO D, ZHANG L, PAN Y, et al. On similarity-awareness in testing-based fault localization [J]. Automated Software Engineering, 2008, 15(2):207-249.

[4] WONG W E, QI Y. An execution slice and inter-block data dependency-based approach for fault localization [C]//Proceedings of 11st Asia-Pacific Software Engineering Conference. Busan: IEEE Computer Society, 2004:366-373.

[5] LIU C, FEI L, YAN X, et al. Statistical debugging: a hypothesis testing-based approach[J]. IEEE Transactions on Software Engineering, 2006,32(10):831-848.

[6] HAO D, ZHANG L, XIE T, et al. Interactive fault local-

ization using test information[J]. Journal of Computer Science and Technology, 2009, 24(5):962-974.

[7] WONG W E, WEI T, QI Y, et al. A crosstab-based statistical method for effective fault localization[C]//Proceedings of the 1st International Conference on Software Testing, Verification and Validation. Lillehammer: IEEE Computer Society, 2008:42-51.

[8] XIE X, WONG W E, CHEN T Y, et al. Spectrum-based fault localization without test oracles[C]//Proceedings of the 11st International Conference on Quality Software (QSIC). Xi'an: IEEE Computer Society, 2011:1-10.

[9] ZHANG X, GUPTA N, GUPTA R. Locating faults through automated predicate switching[C]//Proceedings of the 28th International Conference on Software Engineering. ShangHan: IEEE Computer Society, 2006:272-281.

[10] TUCEK J, LU S, HUANG C, et al. Triage-diagnosing production run failures at the user's site[C]//Proceedings of the 21st ACM Symposium on Operating Systems Principles. Stevenson: Association for Computing Machinery, 2007:131-134.

[11] CHEN M, KICIMAN E, FRATKIN E, et al. Pinpoint: Problem determination in large, dynamic internet services[C]//Proceedings of the 32nd IEEE/IFIP International Conference on Dependable Systems and Networks. Los Alamitos: IEEE Computer Society, 2002:595-604.

[12] JONES J, HARROLD M, STASKO J. Visualization of test information to assist fault localization [C]//Proceedings of the 24th International Conference on Software Engineering. Orlando: IEEE Computer Society, 2002: 467-477.

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

[14] WONG W E, DEBROY V. A survey of software fault localization [R]. Dallas, UTD: Department of Computer Science, 2009.

[15] MACQUEEN J B. Some methods for classification and analysis of multivariate observations[C]//Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. Berkeley: University of California Press, 1967:281-297.

[16] HARTIGAN J A, WONG M A. A *k*-means clustering algorithm[J]. Applied Statistics, 1979, 28(1):100-108.

[17] REPS T, BALL T, DAS M, et al. The use of program profiling for software maintenance with applications to the year 2000 problem[C]//Proceedings of the 6th European Software Engineering Conference Held Jointly with the 5th ACM SIGFOFT International Symposium on Foundations of Software Engineering. Zurich: Springer, 1997:432-449.

[18] HARROLD M, ROTHERMEL G, SAYRE K, et al. An empirical investigation of the relationship between spectra differences and regression faults[J]. Software Testing Verification and Reliability, 2000, 10(3):171-194.

[19] GAMMA E, BECK K, Junit[CP/OL]. Eugene, USA: University of Oregon, 2008.

(编辑:陈燕)

(上接第18页)

[18] LI Dengfeng. Some measures of dissimilarity in intuitionistic fuzzy stru ctures[J]. Journal of Computer and System Sciences, 2004, 68:115-122.

[19] HANEL M, BUISHAND T A. On the value of hourly precipitation extremes in regional climate model simulations[J]. Journal of Hydrology, 2010, 393(3-4):265-273.

[20] IMMONEN A, PALVIAINEN M. Trustworthiness evaluation and testing of open source components[C]// Proceedings of the Seventh International Conference on Quality Software. Los Alamitos: IEEE Computer Society Press, 2007:316-321.

(编辑:陈燕)