软件错误定位研究综述

曹鹤玲[1] 姜淑娟[1] 鞠小林[1,2]
(中国矿业大学计算机科学与技术学院 徐州 221116)[1] (南通大学计算机科学与技术学院 南通 226019)[2]

摘 要 错误定位是软件调试中的一个热点问题,旨在高效地检测出软件错误。首先根据研究方法的不同,将已有错误定位方法从轻量级和重量级两个角度进行分类并进行比较。轻量级错误定位技术不涉及程序依赖关系分析,在程序执行覆盖信息的基础上用统计学或数据挖掘等方法找出可疑错误代码的集合来定位错误;重量级错误定位技术涉及程序依赖关系分析,主要分析数据依赖、控制依赖关系或使用程序切片等来识别可疑代码。然后,总结了常用的评测数据集和评测标准。最后,对错误定位的未来研究趋势进行了展望。

关键词 错误定位,软件调试,软件缺陷,程序依赖关系
中图法分类号 TP311 **文献标识码** A

Survey of Software Fault Localization

CAO He-ling[1] JIANG Shu-juan[1] JU Xiao-lin[1,2]
(School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, China)[1]
(School of Computer Science and Technology, Nantong University, Nantong 226019, China)[2]

Abstract Fault localization is a hot research topic in software debugging, aiming at a rapid and efficient detection of software faults. First, the existing fault localization techniques were classified into two categories according to different research methods: light-weight and heavy-weight fault localization. And similar techniques were compared. The former does not involve the analysis of program dependencies, but it finds out the set of suspicious fault code using statistic or data mining methods based on the coverage information of program execution. The latter involves the analysis of program dependencies and mainly applies data-dependency or control-dependency or program slicing to identify suspicious code. Then, the commonly evaluation data sets and evaluation criteria were summarized. Finally, the trend of future study was discussed.

Keywords Fault localization, Software debugging, Software defect, Program dependency

1 引言

软件发展维护过程中50%～80%的成本用于软件测试和调试[1]。软件错误定位是调试过程中最昂贵、最耗时的活动之一。改进错误定位技术能够提高软件调试效率,降低调试成本。随着计算机技术的飞速发展及软件的普遍应用,软件错误定位已成为国内外研究的一个热点。

软件错误定位在近30年受到了国内外学者的广泛关注。最早可追溯到M. Weiser提出的程序切片[2]。程序切片可以缩小软件调试范围,协助程序员有效地定位程序错误[2]。随后研究者对该领域进行了深入研究,提出了许多行之有效的错误定位方法,如Jones等提出了基于怀疑度计算的Tarantula错误定位方法,即用不同颜色可视化表示错误代码可疑度大小[1,6];Wong等提出了基于改进的径向基函数神经网络错误定位方法[25];Zhang等提出了基于动态切片的错误定位方法[31]。研究人员从不同的角度提出了错误定位方法以协助调试者提高调试效率。

表1 软件错误定位论文数目统计

年份	IEEE	Elsevier	ACM	Springer	CNKI
2013	4	3	3	2	2
2012	36	9	2	1	9
2011	43	4	7	5	7
2010	38	6	6	3	5
2009	31	5	2	2	1
2008	16	2	5	3	1
2007	9	1	3	1	0
2006	14	0	2	4	0
2005	9	0	3	1	1
2004	5	0	0	1	1
2003	3	0	0	0	0
2002	2	0	1	1	0

为了进行全面分析与研究,我们在IEEE、Elsevier、ACM、Springer和CNKI等论文数据库中进行检索,并对检索

到稿日期:2013-04-21 返修日期:2013-06-24 本文受国家自然科学基金(61202006,60970032),江苏省青蓝工程,江苏省高校自然科学基金(12KJB520014),江苏省研究生创新工程(CXZZ12_0935),南通市应用研究计划(BK2011025,BK2012023)资助。
曹鹤玲(1980－),女,博士生,主要研究领域为软件分析与测试、数据挖掘,E-mail:caohl410@cumt.edu.cn;姜淑娟(1966－),女,博士,教授,博士生导师,CCF会员,主要研究领域为编译技术、软件工程等;鞠小林(1976－),男,博士生,讲师,主要研究领域为软件分析与测试。

出的论文进行查阅，最终找到与软件错误定位直接相关的论文316篇(截止2013年3月)。表1按年份和论文库进行分类汇总，从中可以看出，近些年软件错误定位逐渐成为研究热点。

对检索到的文献按中国计算机学会推荐期刊和会议(软件工程领域)分类统计，其中A类期刊和A类会议分别收录9篇和17篇，如发表在TSE上3篇、TOSEM上5篇和ICSE上15篇，以近4年文献居多。

本文对软件错误定位已有研究成果进行了系统的总结。本文第2节对错误定位问题进行描述；第3节分别从轻量级(不分析程序依赖关系)和重量级(分析程序依赖关系)两个角度对现有的错误定位方法进行比较和总结；第4节介绍了实证研究中常用的评测数据集和评测标准；第5节对错误定位未来研究趋势进行展望；最后对软件错误定位进行总结。

与文献[3,4]不同，本文首次从轻量级和重量级两个角度，对错误定位的已有研究成果进行归类、比较与分析。同时补充了近两年国内外研究者在相关期刊和会议发表的最新研究成果。此外，对基于统计和基于数据挖掘的错误定位方法进行了深入的总结。

2 软件错误定位问题描述

软件错误定位旨在找到隐含在程序源代码中的错误指令、过程或数据定义。错误定位的粒度可以是程序语句、基本块、分支、函数或类[3]。错误定位一般采用基于静态分析和基于动态测试的错误定位[4]。基于动态测试的错误定位需要测试用例驱动源程序，利用程序的覆盖信息(或执行轨迹)和运行结果(成功或失败)反向推理程序错误位置。本文重点关注基于测试的动态错误定位方法。

问题描述：(1)有 n 条语句的程序 $P=\{s_1,s_2,s_3,\cdots,s_n\}$。(2)测试用例集 $T=\{t_1,t_2,t_3,\cdots,t_n\}$，每个测试用例 $t_i=\{\langle I_i,O_i\rangle \mid I_i$ 为输入，O_i 为期望输出$\}$，A_i 为实际输出，若 $O_i=A_i$，则 t_i 为执行成功的测试用例；若 $O_i\neq A_i$，则 t_i 为执行失败的测试用例[21]。(3)程序的执行轨迹或覆盖信息。按测试谕言将执行轨迹或覆盖信息分为成功和失败两类。执行轨迹是程序执行过程指令流信息的记录，完整记录了程序执行过程中所执行指令的内容与顺序；覆盖信息只关注某条语句或基本块是否被覆盖到，而不关注覆盖多少次或先后次序[5-8,10,16]。

3 软件错误定位分类

本文根据研究方法的不同，将软件错误定位技术分为轻量级和重量级错误定位两大类[26]。轻量级错误定位一般不涉及程序依赖关系分析，只收集测试的覆盖信息(或执行轨迹)，然后在此基础上采用统计方法或数据挖掘方法进行信息处理。重量级错误定位涉及程序依赖关系分析，主要分析数据依赖和控制依赖关系。程序切片是另一类使用数据依赖和控制依赖关系的错误定位方法，此类方法的时间和空间开销比较大。

3.1 轻量级错误定位

为了实现软件错误定位自动化，人们提出了许多基于统计的错误定位技术，如Tarantula[6]和Ochiai[8]是基于语句的错误定位，SOBER[9]是基于谓词的错误定位。基于统计的错误定位一般使用某一统计指标或模型来处理程序的覆盖信息(或执行轨迹)以定位错误。数据挖掘错误定位是备受关注的另一类轻量级的错误定位技术，主要挖掘小概率事件，擅长处理海量数据。

3.1.1 基于统计的错误定位

基于统计的错误定位按统计分析方法[23]分为参数统计和非参数统计错误定位两类，基于参数统计分析方法涉及描述性统计(如DStar[16]基于Kulczynski系数)、变量数字特征(如FOnly[14]使用方差)、分布函数(如SOBER[9]使用分布密度函数)等方面；非参数统计分析方法包括分布类型检验(Crosstab[10]使用卡方统计)、分布位置检验、秩变换检验等。目前，基于统计的错误定位研究以参数统计方法为主，非参数统计研究较少。

3.1.1.1 参数统计错误定位

Renieris和Reiss[5]于2003年提出了最近邻查询错误定位方法。该方法假定存在一个失效运行和多个成功运行，使用距离度量(Hamming距离和Ulam's距离)方法，从成功运行中统计出与失效运行最“相似”的程序谱，比较这两种运行，去除被失效运行和成功运行都执行的语句，生成可疑语句报告。最近邻查询错误定位比基于集合的错误定位更有效，若错误语句不在可疑报告中，则程序员需检查更多语句定位错误。

Tarantula[6]、Jaccard[7]、Ochiai[8]、DStar[16]等是统计程序实体怀疑度的错误定位方法。此类方法一般统计如下9个指标：成功测试总数 N_p；失败测试总数 N_f；测试用例总数 $N=N_p+N_f$；覆盖语句 s 的成功测试 $N_p(s)$；覆盖语句 s 的失败测试 $N_f(s)$；覆盖语句 s 的测试总数 $N_c(s)=N_p(s)+N_f(s)$；未覆盖语句 s 成功执行次数 $N_{up}(s)$；未覆盖语句 s 的失败执行次数 $N_{uf}(s)$；未覆盖语句 s 的执行次数 $N_u(s)=N_{up}(s)+N_{uf}(s)$。

本文选取几种典型的方法进行介绍，其怀疑度计算公式如表2所列。Jones等[6]提出了Tarantula错误定位方法。应用此公式计算每个语句的怀疑度，然后对每个语句按照怀疑度排序。一个语句怀疑度值越大，说明该语句错误的概率值越大，否则该语句出错的概率越小。Chen等[7]提出了Jaccard错误定位方法，使用了聚类分析中的系数。Abreu等[8]提出了Ochiai错误定位方法，引入分子生物学领域的相似系数(similarity coefficients)Ochiai来定义语句的怀疑度，然后对每个语句按怀疑度大小进行排序，并通过实验表明Ochiai的定位效果优于Tarantula方法。Artzi等人[36]将Tarantula、

Jaccard、Ochiai 3 种错误定位技术应用到 Web 错误定位中(PHP 语言)。实验结果表明此 3 种方法在 Web 错误定位中效果良好。Wong 等[16]基于 Kulczynski 系数,提出 DStar 错误定位方法,该方法可定位单错误和多错误程序。用 16 种错误定位方法对 21 个程序进行了比较,结果表明该方法优于其他同类方法(如 Tarantula 和 Ochiai)。

表 2　几种典型的怀疑度计算公式(名称,年代,作者,发表会议/期刊)

名称	年代	作者	发表会议/期刊	公式
Tarantula	2002	Jones 等	International Conference on Software Engineering	$\frac{Nf(s)/Nf}{Np(s)/Np+Nf(s)/Nf}$
Jaccard	2002	Chen 等	International Conference on Dependable Systems and Networks	$\frac{Nf(s)}{Nf(s)+Nuf(s)+Np(s)}$
Ochiai	2007	Abreu 等	Testing: Academic and Industrial Conference Practice and Research Techniques	$\frac{Nf(s)}{\sqrt{Nf(s)\times(Nf(s)+Np(s))}}$
DStar	2012	Wong 等	International Conference on Software Security and Reliability	$\frac{Nf^{*}(s)}{Nuf(s)+Np(s)}(*=2,3,\cdots)$

Naish[24]总结了 33 种基于怀疑度尺度的错误定位方法,并提出了两种最优的怀疑度计算公式如下:

$$O_1(s)=\begin{cases}-1, & N_f(s)<N_f\\ N_p-N_p(s), & N_f(s)=N_f\end{cases} \quad \text{(Naish1)}$$

$$O_2(s)=N_f(s)-\frac{N_p(s)}{N_p(s)+N_{up}(s)+1} \quad \text{(Naish2)}$$

Xie 等[41]进一步对文献[24]中 30 个怀疑度公式的错误定位性能进行理论分析与证明,证明了两组公式为最优。其中一组为上述 Naish[24]提出的两个最优公式;另一组包括如下 3 种:

$$O_3(s)=N_f(s) \quad \text{(Wong1)}$$

$$O_4(s)=\frac{N_f(s)}{N_f(s)+N_{uf}(s)+N_p(s)+N_{up}(s)} \quad \text{(Russel\&Rao)}$$

$$O_5(s)=\begin{cases}0, & N_{uf}(s)>0\\ 1, & \text{otherwise}\end{cases} \quad \text{(Binary)}$$

Liu 等[9]基于参数统计中的分布函数,提出了 SOBER 错误定位模型。该模型对正确和错误测试进行了统计分析,通过比较程序谓词在不同测试用例下的分布情况来寻找最有可能出错的谓词。基本思路为:对于一个谓词 X,若它的潜在分布密度函数 $f(X|f_f)$不同于所有正确执行中的分布密度函数 $f(X|f_p)$,则它与软件错误相关。Liblit 等[12]提出了 Liblit05 错误定位模型。该统计模型收集谓词的覆盖信息,对成功测试和失效测试中谓词为真的概率进行统计分析,进而检测谓词发生错误的可能性,并且首先找到一个最有可能出错的谓词,修复此错误后继续寻找下一个错误,其从而定位多错误。文献[9,12]从谓词的角度使用统计方法来定位错误,其不同于 Tarantula、Jaccard 和 Ochiai 等从语句的角度定位错误的方法。

Hong 等[13]提出了一种基于简化运行程序的统计错误定位方法。该方法首先通过聚类减少冗余执行路径,然后计算简化后的失败和成功执行路径之间差异的统计特征,对其进行排序,从而定位错误。Zhang 等[14]提出了只使用失效运行轨迹的(FOnly)错误定位方法,FOnly 只关注失败运行轨迹,从统计趋势(方差)上估计失败运行从而定位错误,而一般的错误定位方法考虑成功和失效两类运行轨迹。不管是减少冗余执行路径,还是只关注失效测试,与使用统计方法的 Tarantula、Jaccard 和 Ochiai 相比,文献[13,14]的方法可以减少信息处理量。

Yan 等[33]提出了一种基于反馈(Feedback)的错误定位方法。在软件调试阶段,软件调试和错误定位技术之间的有效互动可以极大改善错误定位效率。但多数方法忽视了这种交互,仅利用测试信息进行错误定位,导致信息不充分。该方法首先通过反馈模拟这种交互作用,利用测试数据生成技术和错误定位获取的结果进行交互,自动产生反馈,然后迭代该过程以改善错误定位效率,直到满足终止条件。Qin 等[17]提出了一种基于覆盖信息的面向对象程序的错误定位评估框架。此框架采用程序谱记录每个测试用例的执行信息,然后结合执行语句的程序谱信息和类的相关信息来提高错误定位的准确度和效率。

惠战伟和黄松[15]提出了基于程序特征谱覆盖统计的整数溢出错误定位方法,旨在解决整数溢出的错误问题。获取分支覆盖特征谱信息和定义使用对特征谱覆盖信息,计算分支可疑度和定义使用对的怀疑度,计算错误传播率,从而求出语句怀疑度之间的关系,定位错误。

3.1.1.2　非参数统计错误定位

Wong 等[10]基于非参数统计中的卡方检验,提出了交叉表(Crosstab)统计的错误定位方法。该方法分别统计成功执行与失败执行情况下可执行语句 s 分别被覆盖的次数并构建该语句的交叉表,进而进行 s 卡方统计检验。实验结果表明该方法仅需检测较少可疑语句就可以定位出错误语句,比 Tarantula 技术更有效;运行时间与 Tarantula 差别不大。

Wong 等[11]改进了基于 Crosstab 统计的错误定位方法,即 CBT(Crosstab-Based Technique)错误定位方法。该方法在 2 组小数据集和 4 个大数据集上进行了对比实验。在小数据集(Siemens suite)上的实验表明该方法比 SOBER[9]和 Liblit05[12]更有效;在大数据集(如 Ant)上的实验表明,该方法在时间消耗、测试组件敏感性等方面具有明显优势。

张震宇等[41]从单元测试的特性出发,采用马尔科夫模型对错误类型进行预测,找到目标错误和程序类型的相关性,从而选择合适的错误定位技术,提出了基于马尔科夫模型的错误定位方法。

3.1.2 基于数据挖掘的错误定位

Brun 和 Ernst[18]使用支持向量机和决策树模型对错误不变式分类，提出了基于支持向量机的错误定位方法。Giuseppe 等[19]通过构造函数调用树，在函数调用树基础上挖掘与失败测试相关的频繁子树，提出了基于频繁模式挖掘的错误定位方法。Denmat 和 Ducasse[20]挖掘执行轨迹，提出了基于关联规则的错误定位方法，用支持度和置信度筛选与错误相关语句。Wong 等[21]通过构造长度为 N 的实际执行序列，利用关联规则挖掘 N 长度实际执行子序列与测试用例失败的关联关系，进而提出了 N-gram 和关联规则相结合的错误定位技术。Cellier 和 Ducasse[22]使用形式化概念和关联规则，判别出语句和失败测试的关联，从而定位程序错误。Wong 等[25]使用改进的径向基(RBF)神经网络来揭示语句覆盖信息与相应的成功/失败测试之间的关联，从而提出了基于神经网络的软件错误定位方法。此类错误定位方法都使用数据挖掘相关技术来定位错误，是错误定位与数据挖掘方法的交叉研究，有着广阔的研究前景。

3.2 重量级错误定位

涉及数据依赖或控制依赖分析的错误定位方法，与不分析依赖关系的方法相比，需要耗费高昂时间和空间代价，是重量级错误定位方法。程序依赖关系主要用来增加执行实例间的权值，从而定位程序错误，此类方法为基于依赖关系的错误定位。程序切片是另一类基于程序依赖关系的错误定位，与仅使用依赖关系增加执行实例间权值的方法不同，程序切片从执行轨迹中选择与错误输出语句相关的所有语句或使用程序插桩构建程序依赖图，从而缩小错误定位可疑语句范围，此类方法为基于程序切片的错误定位。

3.2.1 基于依赖关系的错误定位

Baah 等[27]提出了基于概率程序依赖图的错误定位模型。在程序依赖关系图上增加节点和边建立概率依赖图，用来反映程序内部行为，对错误相关联的程序行为不确定性进行概率分析和推理，从而使用概率依赖图诊断错误。Zhao 等[28]提出了基于控制流分析的有意识执行(Execution- Aware)错误定位方法。其使用边的怀疑度和加权的覆盖信息统计方法分析错误定位过程。Christ 等[29]提出了一种控制流敏感的错误定位方法。该方法使用控制流信息来识别错误执行轨迹中不相关条件分支，判断错误定位相关的条件分支真值，生成更有意义的错误解释来定位错误。衷璐洁等[40]提出一种基于域敏感、流敏感和上下文敏感的传播引擎，在传播引擎的基础上检测 C 语言中指针引用错误。以上错误定位都是考虑依赖关系来增强程序实体的相互关系从而定位错误。

3.2.2 基于程序切片的错误定位

李必信等[30]提出了基于面向对象程序的层次切片错误定位方法。该方法通过删除通过测试的包、类和方法来缩小软件错误搜索范围；通过变量密集程度计算优先级，根据模块内使用变量数目的规模，逐步求精；通过动态分析，选取一组测试用例计算产生错误的程序执行轨迹与静态分析过程中产生模块的交集及交集的后向切片，直至定位到错误。

Zhang 等[31]提出了一种基于动态切片的错误定位方法。在明确程序输出的前提下定义切片准则，从而计算数据切片、全切片和相关切片，进而使用程序切片有效分离出错误相关语句来定位错误。他们还提出一种基于多点切片的错误定位方法[32]，比较了 3 种单点切片的不同：后向切片是在有错误输出的前提下，后向遍历动态依赖图；前向切片是在找到最小诱导输入集的前提下，前向遍历动态依赖图，双向切片是前向和后向遍历动态依赖图。为了解决单点切片错误定位候选语句集过大的问题，Zhang 等[32]提出了多点切片错误定位方法。多点动态切片是指从错误的值、导致错误的输入和关键谓词开始计算程序切片，而不只从单一点开始计算程序切片。首先检查关键谓词，若程序崩溃，则检查前向，强制执行其他分支语句，以确定其他分支是否错误；如果错误输出，则检查后向所遍历语句和变量，从而得到一个较小的错误候选集，缩小错误定位范围。

Wen[42]提出了基于程序切片谱的错误定位方法。因只计算程序元素怀疑度的错误定位方法能力有限，该文结合了程序切片和统计指标。程序切片可以提取程序元素间依赖关系和精炼程序执行轨迹；统计指标计算出程序切片谱中语句一个怀疑度等级，从而定位错误。

我们提出了一种识别空指针异常的错误定位方法[34]。该方法首先在实时堆栈信息指导下计算程序切片，然后在切片后的程序上进行空指针和别名分析，从而解决空指针异常的错误定位问题。

4 实证研究

已有的错误定位研究大多数采用实验验证的方法对错误定位技术的有效性进行评测。本节总结了常用的评测数据集和经典的评测指标。

4.1 评测数据集

研究人员通常使用一些评测数据集来检验软件错误定位技术的有效性。评测数据集涉及面向过程语言和面向对象语言，如 C、C++、C# 和 Java 等，错误类型为植入错误、真实错误和变异错误。

西门子套件(Siemens Suite)是最常用的评测数据集，由西门子研究院为研究数据流、控制流对错误的探测能力而创建[3]。文献[6-8,16,25]通过 Siemens Suite 验证错误定位的有效性。Unix suites 是工业中包含真实错误的程序，共包括 10 个不同程序。错误定位文献中常用数据集的名称、描述、代码行数、错误版本数与测试用例数如表 3 所列。数据集大多从 SIR(The Software Infrastructure Repository)[37] 或 SouceForge[38]下载。

表 3　数据集简介

Program		Language	Description	Lines of Code	Faulty versions	Test cases	Fault types	Source
Siemens suite	print_tokens	C	lexical analyzer	565	7	4130	seeded	[14,16,42]
	print_tokens2	C	lexical analyzer	510	10	4115		
	schedule	C	priority scheduler	412	9	2650		
	schedule2	C	priority scheduler	307	10	2710		
	replace	C	pattern replacement	563	32	5542		
	tcas	C	altitude separation	173	41	1608		
	tot_info	C	information measure	406	23	1052		
Unix suite	cal	C	Print a calendar for a specified year or month	202	20	1162	real	[16,42]
	checkeq	C	Report missing or unbalanced delimiters and .EQ/.EN pairs	102	20	166		
	col	C	Filter reverse paper motions from nroff output	308	30	156		
	comm	C	Select or reject lines common to two sorted files	167	12	186		
	crypt	C	Encrypt and decrypt a file using a user supplied password	134	14	156		
	look	C	Find words in the system dictionary or lines in a sorted list	170	14	193		
	sort	C	Sort and merge files	913	21	997		
	spline	C	Interpolate smooth curves	338	13	700		
	tr	C	Translate characters	137	11	870		
	uniq	C	Report or remove adjacent duplicate lines	143	17	431		
gzip		C	Reduce the size of named files using the Lempel-Ziv coding	6573	28	211	seeded	[14,16,42]
grep		C	Search for a pattern in a file	12653	19	470	seeded	[14,16,42]
make		C	Manage building of executables and other products from code	20014	31	793	seeded	[16,42]
space		C	Provide a user interface to configure an array of antennas	9564	38	13585	real	[42]
sed		C	A stream text editor	14427	22	370	real, seeded	[14]
flex		C	A fast lexical analyzer generator	10459	5	567	seeded	[14,32]
NanoXML v1		Java	XML parser	3497	7	214	seeded	[26]
NanoXML v2		Java		4009	7	214	seeded	[26]
NanoXML v3		Java		4608	10	216	seeded	[26]
NanoXML v5		Java		4782	8	216	seeded	[26]
JTopas v1		Java	Text data interpreter	3341	4	126	seeded	[43]
JTopas v2		Java		4172	6	145	seeded	[43]
JTopas v3		Java		5400	4	205	seeded	[43]
XML-security v1		Java	XML encryption	21613	7	92	seeded	[26]
XML-security v2		Java		22318	7	94	seeded	[26]
XML-security v3		Java		19895	2	84	seeded	[26]
JABA		Java	Program analyzer	37966	677	11	real	[26]
JABA		Java	Program analyzer	37966	677	11	real	[26]
Ant		Java	A Java library and a tool for automating software build processes	75333	23	871	seeded	[16]

4.2　评测标准

错误定位的评测标准是用来评价软件错误定位技术定位精度和准确率的指标。

(1)分数值法(EXAM Score)。早期研究中,Renieris 和 Reiss[4]采用分数值法来评测错误定位效果指标。分数值法指用户定位到错误语句时未检测的代码数占总代码数的百分比,分数越高表明错误定位效果越好。Harrold 等在文献[6]中也采用此评测标准。Liu 等[35]将 T-score 作为检测标准,其为第一层深度优先遍历所检测的节点数占程序依赖图总节点的百分比。Wong 等[16]改进了分数值法评测软件错误定位效果的方法,该方法亦即直到检测到错误语句时需要检测的语句占总语句的百分比。

(2)累积检查语句(Cumulative Number of Statements Examined)。此指标是指若程序有 m 个错误,分布在 n 个错误版本中,累积检查语句数为检测出 n 个错误版本中所有 m 个错误需要总共检查的语句数;检测的总语句数越少,表明错误定位技术的效果越好,文献[16,25]采用了此指标来评测错误定位技术的效果。

5　研究展望

本文检索了权威论文数据库 IEEE、Elsevier、ACM、Springer 和 CNKI 中错误定位相关论文,并进行了较为全面的对比和总结,得出结论:(1)不需要分析程序实体之间的依赖关系,只在收集程序执行结果成功/失败的覆盖信息基础上,使用统计学方法对程序实体(语句、分支、函数和类等)的怀疑度进行计算排序的错误定位技术的相关研究成果最多,而使用数据挖掘或机器学习方法处理覆盖信息找出可疑错误代码的相关研究较少。(2)需要分析程序实体之间依赖关系的错误定位方法(如基于程序切片错误定位方法)的研究也有一些。

但上述方法有一个共同的问题,即只考虑了程序的动态信息。程序动态信息的获取需要设计、运行大量测试用例,并

记录海量动态执行信息且软件测试充分性难以保证，动态分析方法存在漏报可能性。而静态分析方法不需要运行程序且能获取错误定位的一些有用信息，因此，静动结合的错误定位方法是未来错误定位的一个研究趋势。

上述两类方法各有优点，将两者优点有机结合是值得研究的错误定位方法，比如使用程序切片缩小错误定位范围，再使用统计方法计算出切片后语句的优先级次序以协助定位错误是值得研究的方向之一。

随着软件规模增大和复杂程度的提高，程序执行信息海量增长，这使得现有错误定位方法面临严峻挑战。而数据挖掘具备处理大数据能力，机器学习具备自适应学习能力。因此基于数据挖掘、机器学习等的错误定位技术值得深入研究。

结束语 软件错误定位是软件调试中的一个热点问题。本文通过调研经典论文数据库，对该问题的研究背景、已有研究成果、评测数据集和评测指标进行了系统的梳理和总结。近些年该研究领域非常活跃。本文根据研究方法的不同将现有的错误定位技术分为轻量级和重量级两大类，阐述了具有代表性的错误定位技术。最后对软件错误定位的未来研究趋势进行预测，提出了若干值得研究的方向。

参 考 文 献

[1] 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. New York, USA: ACM, 2005: 273-282

[2] Weiser M. Program slicing[J]. IEEE Transactions on Software Engineering, 1984, 10(5): 498-509

[3] 虞凯，林梦香. 自动化软件错误定位技术研究进展[J]. 计算机学报，2011，34(8)：1411-1422

[4] 鞠小林，姜淑娟，张艳梅，等. 软件故障定位技术进展[J]. 计算机科学与探索，2012，6(6)：481-494

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

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

[7] Chen M Y, Kiciman E, Fratkin E, et al. Pinpoint: problem determi- nation in large, dynamic internet services[C]// Proceedings of the 2002 International Conference on Dependable Systems and Networks. USA: IEEE Computer Society, 2002: 595-604

[8] 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

[9] Liu Chao, Yan Xi-feng, Fei Long, et al. SOBER: statistical model-based bug localization [J]. ACM SIGSOFT Software Engineering Notes, 2005, 30(5): 286-295

[10] Wong W E, Wei Ting-ting, Qi Yu, 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, Washington, DC, USA: IEEE Computer Society, 2008: 42-51

[11] Wong W E, Debroy V, Xu Dian-xiang. Towards better fault localization: a crosstab-based statistical approach [J]. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 2012, 42(3): 378-396

[12] Liblit B, Naik M, Zheng AX, et al. Scalable statistical bug isolation [J]. ACM SIGPLAN Notices, 2005, 40(6): 15-26

[13] Hong Li-na, Chen Rong. Statistical fault localization with reduced program runs[J]. Artificial Intelligence Applications and Innovations IFIP Advances in Information and Communication Technology, 2010, 339: 319-327

[14] Zhang Z, Chan W K, Tse T H. Fault localization based only on failed runs [J]. Computer, 2012, 45(6): 64-71

[15] 惠战伟，黄松，嵇孟雨. 基于程序特征谱整数溢出错误定位技术研究[J]. 计算机学报，2012，35(10)：2204-2214

[16] Wong W E, Debroy V, Li Y h, et al. Software fault localization using DStar (D*)[C]// 2012 IEEE Sixth International Conference on Software Security and Reliability. 2012: 21-30

[17] Qing X, Yu P, Wang L. An evaluation framework of coverage-based fault localization for object-oriented programs[J]. Trustworthy Computing and Services Communications in Computer and Information Science, 2013, 320: 591-597

[18] Brun Y, Ernst M D. Finding latent code errors via machine learning over program executions[C]// Proceedings of the 26th International Conference on Software Engineering. 2004: 480-490

[19] Giuseppe D F, Stefan L, Evghenia S. Discriminative pattern mining in software fault detection[C]// Proceedings of the Third International Workshop on Software Quality Assurance. 2006: 62-69

[20] Denmat T, Ducasse M, Ridoux O. Data mining and cross-checking of execution traces: a re-interpretation of Jones, Harrold and Stasko test information[C]// Proceedings of the 20th International Conference on Automated Software Engineering. Long Beach, CA, 2005: 396-399

[21] Nessa S, Abedin M, Wong W E, et al. Software fault localization using n-gram analysis [C]// Wireless Algorithms, Systems, and Applications Lecture Notes in Computer Science. 2008: 548-559

[22] Cellier P, Ducasse S, Ferre S, et al. Formal concept analysis enhances fault localization in software[C]// Proceedings of the 4th International Conference on Formal Concept Analysis. Montréal, Canada, 2008: 273-288

[23] 易丹辉，董寒青. 非参数统计：方法与应用[M]. 中国统计出版社，2009：6-7

[24] Naish L, Lee H, Ramamohanarao K. A model for spectra-based software diagnosis [J]. ACM Transactions on Software Engineering and Methodology, 2011, 20(3): 57-89

[25] Wong W E, Debroy V, Golden R, et al. Effective software fault localization using an RBF neural network[J]. IEEE Transactions on Reliability, 2012, 61(1): 149-169

[26] Santelices R, Jones J A, Yu Yan-bing, et al. Lightweight fault-localization using multiple coverage types[C]// Proceedings of the 31st International Conference on Software Engineering. Washington, DC, USA: IEEE Computer Society, 2009: 56-66

[27] Baah G K, Podgurski A, Harrold M J. The probabilistic program dependence graph and its application to fault diagnosis[J]. IEEE Transactions on Software Engineering, 2010, 36(4): 528-545

（下转第 14 页）

ted,2000:120-126
[3] Gardner H. Changing Minds: the Art and Science of changing our own and Other People's Minds[M]. New York: Penguin Group (USA) Incorporated,2006:89-96
[4] APA-CAP. The Digital Phoenix:How Computers Are Changing Philosophy[M]. New York: Springer Publishing Company, 1998:78-82
[5] 郝宁湘. 计算:一个新的哲学范畴[J]. 哲学动态,2000,18(11):32-36
[6] 李建会. 走向计算主义[J]. 自然辩证法通讯,2003,22(3):31-36
[7] Karp R. Understanding Science Through The Computational Lens[J]. Journal of Computer Science And Technology,2011,26(4):68-75
[8] 黄崇福. 信息扩散原理与计算思维机器在地震工程中的应用[M]. 北京:北京师范大学,1992:168-178
[9] 董荣胜,古天龙. 计算机科学与技术方法论[M]. 北京:人民邮电出版社,2002:98-110
[10] Wing J M. Computational Thinking[J]. Communications of the ACM,2006,22(9):45-49
[11] Cuny J, Snyder L, Wing J M. Demystifying CT For Non-Computer Scientists[J]. Work In Progress,2010,8(3):30-36
[12] ISTE. Operational Definition of Computational Thinking for k-12 Education[OL]. http://www. iste. /Libra ries/PDFs/Operational_Definition_of_Computational_Think ing. sflb. ashx,2012-04-15
[13] 王飞跃. 从计算思维到计算文化[J]. 中国计算机学会通讯,2007,17(11):10-15
[14] 爱因斯坦. 爱因斯坦文集[M]. 北京:科学出版社,1989:245
[15] 李廉. 计算思维——概念与挑战[J]. 中国大学教学,2012,7(1):7-12
[16] 谭浩强. 研究计算思维,坚持面向应用[J]. 计算机教育,2012,6(21):45-50
[17] 西安交通大学计算机教学实验中心. 国家级精品课《大学计算机基础》问卷调查统计数据[OL]. http://computer. xjtu. edu. cn/dcwj_l. htm,2012-5-17
[18] IEEE/ACM. Computing Curricula 2004[OL]. http://www. acmtyc. org,2005-6-21
[19] Wing J M. Computational Thinking and Thinking about Computing[J]. Philosophical Transactions of the Royal Society A, 2008,10(4),37-45
[20] 朱亚宗. 论计算思维——计算思维的科学定位、基本原理及创新路径[J]. 计算机科学,2009,12(4):53-56
[21] 吴文虎,王建德. 世界大学生程序设计竞赛高级教程—程序设计中常用的计算思维方式[M]. 北京:中国铁道出版社,2009,180-184
[22] 何明昕. 关注点分离在计算思维和软件工程中的方法论意义[J]. 计算机科学,2009,12(4):60-63
[23] 赵岭忠,钱俊彦,蔡国永. 算法设计策略与计算思维[J]. 企业科技与发展,2010,7(8):35-38
[24] 包云岗. 世纪图灵纪念[J]. 中国计算机学会通讯,2012,22(11):42-47

(上接第 6 页)

[28] Zhao Lei, Wang Li-na, Xiong Zuo-ting, et al. Execution-aware fault localization based on the control flow analysis[J]. Information Computing and Applications Lecture Notes in Computer Science,2010,6377:158-165
[29] Christ J, Ermis E, Schäf M, et al. Flow-sensitive fault localization[J]. Verification, Model Checking, and Abstract Interpretation Lecture Notes in Computer Science,2013,7737:189-208
[30] 许高阳,李必信,孙小兵,等. 一种基于层次切片的软件错误定位方法[J]. 东南大学学报:自然科学版,2010,40(4):692-698
[31] Zhang X, He H, Gupta N, et al. Experimental evaluation of using dynamic slices for fault location[C]// Proceedings of the 6th interna- tional symposium on Automated analysis-driven debugging. Monterey, California, USA,2005:33-42
[32] Zhang X, Gupta N, Guptaand R. Locating faulty code by multiple points slicing [J]. Software: Practice and Experience, 2007, 37:935-961
[33] Lei Yan, Mao Xiao-guang, Dai Zi-ying, et al. Effective fault localization approach using feedback[J]. IEICE Transactions on Information and Systems,2012,95(9):2247-2257
[34] Jiang Shu-juan, Li Wei, Li Hai-yang, et al. Fault localization for null pointer exception based on stack trace and program slicing [C]//Proceedings of the 12th International Conference on Quality Software. 2012:9-12
[35] 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
[36] Artzi S, Dolby J, Tip F, et al. Fault localization for dynamic web applications[C]//Proceedings of the 19th international symposium on Software testing and analysis. Trento, Italy, ACM,2012, 38:314-335
[37] http://sir. unl. edu/portal/index. html
[38] http://sourceforge. net
[39] 张云乾,郑征,季晓慧,等. 基于马尔可夫模型的软件错误定位方法[J]. 计算机学报,2013,36(2):445-456
[40] 衷璐洁,霍玮,李丰,等. 基于传播引擎的指针引用错误检测[J]. 计算机学报,2013,36(2):432-444
[41] Xie X Y, Chen T Y, Kuo F C, et al. A theoretical analysis of the risk evaluation formulas for spectrum-based fault localization [J]. ACM Transactions on Software Engineering and Methodology,2013
[42] Wong W E, Debroy V, Choi B. A family of code coverage-based heuri- stics for effective fault localization[J]. Journal of Systems and Software,2010,83(2):188-208
[43] 顾庆,唐宝,陈道蓄. 一种面向测试需求部分覆盖的测试用例集约简技术[J]. 计算机学报,2011,34(5):879-888
[44] Wen wan-zhi. Software fault localization based on program slicing Spectrum[C]// Proceedings of the 2012 International Conference on Software Engineering. 2012:1511-1514