软件故障定位关键技术研究综述

黄小红，赵逢禹

（上海理工大学 光电信息与计算机工程学院，上海 200093）

摘　要：软件故障定位旨在利用程序信息和测试信息定位出故障语句或给出出错语句的可疑范围，以辅助提高软件质量。随着软件结构的复杂化与软件规模的扩大化，程序运行状态呈指数级增长，已经无法单独依靠手动定位技术进行故障排查。因此，越来越多的辅助定位技术应运而生。为了探索这些技术的实用性，首先介绍了传统的故障定位技术以及近年来最新推广的故障定位技术，然后总结了常用的测试程序集，最后提出故障定位的关键问题并指出了未来值得关注的研究方向。

关键词：故障定位；程序调试；软件测试；执行跟踪；可疑代码

DOI：10.11907/rjdk.171181

中图分类号：TP301　　**文献标识码**：A　　**文章编号**：1672-7800(2017)007-0205-05

0 引言

软件应用的日益广泛已深入影响到人们的生活[1-3]，尤其在许多危及安全的行业，如医药、航空、核能行业等，而且这种趋势随着软件的复杂化而更加明显。与此同时，也衍生了令人堪忧的软件故障，故障的产生不仅导致经济损失，甚至可能造成生命威胁[4]。伴随着软件成本的增加，超过一半的修复成本被传递给消费者。因此，为了降低成本、提高软件质量，必须加强软件故障定位的研究。

软件调试中一个很重要的因素就是找到产生程序故障的根源，失败问题产生的根源可能离程序展示出的位置距离很远，例如程序崩溃或产生故障输出时无法直接定位具体故障源。因此，软件故障定位和软件故障修复应运而生。相对于软件修复，故障定位更为重要，定位是修复的必要条件。由于软件系统规模不断扩大，使传统的手工定位耗时耗力，而且受软件开发人员的思维定位，以及编码风格、约束规范限制，可能导致无法快速有效地找到故障位置。通过引入自动化故障定位技术，可使定位效率大大提高。目前，越来越多的研究者开始致力于软件故障定位的研究且取得了可观的成果，本文主要对2010年以来故障定位的相关成果进行研究，以期准确把握故障定位的发展方向。

像获取流程。根据深度成像特点，对原始深度数据帧进行中值滤波、闭运算和开运算等预处理分析，为行人运动目标的特征识别打下坚实基础。

参考文献：

[1] CHEN T H. An automatic bi-directional passing-people counting method based on color image processing[C]. IEEE，2003 International Carnahan Conference on Security Technology，2003：200-207.

[2] 吴国斌. KINECT 人机交互开发实践[M]. 北京：人民邮电出版社，2013.

[3] 黄露丹，严利民. 基于 Kinect 深度数据的人物检测[J]. 计算机技术与发展，2013(4)：119-121.

[4] SARBOLANDI H，LEFLOCH D，KOLB A. Kinect range sensing：structured-light versus time-of-flight kinect[J]. Computer Vision & Image Understanding，2015.

[5] L LI. Time-of-flight camera-an introduction[Z]. Texas Instruments-Technical White Paper.

[6] CORTI A，GIANCOLA S，MAINETTI G，et al. A metrological characterization of the Kinect V2 time-of-flight camera[J]. Robotics & Autonomous Systems，2015，75(8)：584-594.

[7] 杨召君. 基于视频人数统计与跟踪改进算法的研究与实现[D]. 南京：南京邮电大学，2013.

[8] 郭秀杰. 基于 Kinect 的人流量统计系统研究[D]. 重庆：重庆大学，2014.

[9] 吴晓阳. 基于 OpenCV 的运动目标检测与跟踪[D]. 杭州：浙江大学，2008.

（责任编辑：杜能钢）

基金项目：国家自然科学基金青年基金项目(61402288)

作者简介：黄小红(1989－)，女，河南平顶山人，上海理工大学光电信息与计算机工程学院硕士研究生，研究方向为软件工程、缺陷定位；赵逢禹(1963－)，男，山东德州人，上海理工大学光电信息与计算机工程学院教授、研究生导师，研究方向为软件工程、软件质量保证。

1 故障定位技术

1.1 故障定位常见技术

传统的故障定位技术在简单的小型程序中解决了软件可靠性低的难题，提高了软件质量，但是随着软件的规模化与复杂化发展，近年来不断出现新的故障定位技术，虽然没有任何一种技术可以适用于所有故障定位场景，但是各种技术都各具优势。

1.1.1 基于覆盖的故障定位技术

故障定位研究实质上是解决如何把存在故障的程序语句排名推至首位。基于覆盖的故障定位技术较为直观，覆盖分析法正是采用可疑度描述来降低定位复杂度，尤其适用于一些大规模的故障定位。Wong 等[6]作了如下假设：元素可疑度与被成功测试用例覆盖次数成反比，与被失败测试用例覆盖次数成正比。简而言之，如果该元素被失败测试用例覆盖的多，而很少被成功测试用例覆盖，该处引发故障的可能性则很大；Jones 等[5]基于同样的思想提出利用覆盖信息和测试信息定位错误，并开发了可视化工具 Tarantula；Xie 等[7]提出了集合理论的分析框架，之后又通过蜕变技术将软件测试分析技术集成到故障定位中[8]，反映了更加真实的场景。其不足之处是测试用例的数量、质量限制了覆盖的全面性，而且如果用例数量过少，覆盖不全面，或者用例质量差，大量测试用例重复覆盖相同代码，将产生大量冗余覆盖，从而导致故障定位的精准度降低[9]。

1.1.2 基于程序切片的故障定位技术

程序切片技术将程序抽象化，通过删除不相关的部分将程序用切片表示，切片保留原始程序的行为规范。程序切片技术包括静态切片技术和动态切片技术。静态切片技术是对源程序采取静态分析技术，利用语句之间的依赖关系将相关变量语句集合形成静态切片。静态切片技术最早由 Weiser 首次提出[10]，该技术主要将整个程序分为若干个切片，通过含有错误变量的程序切片来排除无关变量，从而缩减故障范围。静态切片的缺点是，片的大小可能会对给定的可执行语句带来影响，如变量值随着执行的改变导致切片发生变化，而且当程序规模较大时，算法时间开销会很大。因此，Zhang 等[11]提出了多种动态切片技术，包括后向动态切片（BWS）、前向动态切片（FWS）和双向动态切片（BIS）。BWS 通过执行语句输出而捕捉故障变量的值；FWS 通过计算成功用例的最小输入触发故障所在位置；BIS 通过确定失败执行用例的谓词执行情况，生成正确的输出，从而找到故障所在位置。

1.1.3 基于程序频谱的故障定位技术

基于频谱分析进行故障定位的方法简单而言是根据测试用例集合运行完成后的记录信息，使用风险评估方法对程序中的每一行语句计算故障可疑度。可利用的记录信息主要是程序执行过程中的剖面信息与测试用例是否通过的信息[31]。

表 1 给出了常用的符号说明作参考。

表 1 符号说明

P	程序	NU	测试用例未覆盖语句总数
N_{CF}	被失败的测试用例覆盖语句	N_S	执行成功的测试用例总数
N_{UF}	被失败的测试用例未覆盖语句	N_F	执行失败的测试用例总数
N_{CS}	被成功的测试用例覆盖语句	T	测试用例执行结果为成功
N_{US}	被成功的测试用例未覆盖语句	F	测试用例执行结果为失败
N_C	测试用例覆盖语句总数	T_i	第 i 个测试用例

早期研究[12-13]认为基于程序频谱的技术只使用执行失败的测试用例的频谱信息即能准确进行故障定位。后续研究结合了执行成功和失败两种频谱信息，并强调它们之间的对比关系。Pan 等通过各测试用例执行结果产生的频谱差异性对比，提出了交并集模型。并集关注源代码中由执行失败的测试用例产生的频谱，交集则排除了源代码所有执行成功的频谱；Reiss 等[30]提出最近邻模型技术，该技术的思想是找到与执行失败的测试距离最近的执行成功的测试，首先构建一个程序依赖图（PDG），然后检查图中相邻节点，直到所有节点被检查完毕。

1.2 最新推广的故障定位技术

1.2.1 基于数据挖掘的故障定位技术

基于数据挖掘的故障定位技术试图从大量程序运行信息数据中提取相关信息，产生一个可使用的数据模型，从而达到定位效果。Cellier 等[15]提出了数据挖掘的关联规则和概念分析以协助故障定位，该技术通过语句覆盖率之间的关系和相应的执行失败的测试用例之间的关联关系，测量每个规则出现的频率，通过已给阈值筛选出满足关联规则的语句执行数量，生成规则点阵，最后生成一个可疑度排名来定位故障；Zhang 等[16]提出一种基于数据挖掘的故障定位技术，该技术基于马尔可夫逻辑，结合一阶逻辑与加权马尔可夫随机域进行感知器算法学习；王鹏等[14]提出基于图挖掘的技术，通过收集程序的执行轨迹信息构造出程序调用图，提取频繁边辅助定位故障。

1.2.2 基于模型检测的故障定位技术

基于模型检测的故障定位技术通过推导期望的程序行为模型，检测失效执行对期望行为的违背情况，从而捕捉故障行为，主要需解决的关键问题是模型表达能力问题，以及模型对定位效果是否有显著影响力，是否快速有效。Baah 等[17]提出通过程序依赖图模型研究程序的内部行为，促进了可能与故障有关语句的概率分析和推理。基于检测的模型主要依赖模型检查器来定位故障[18]。如果一个模型不满足相应的程序规范（这意味着模型包含至少一个故障），模型检查器可提供反例显示违反规范详情，反例并不直接指定哪些部分的模型与故障相关，但是它可以被视为一个可确定因果关系的失败测试用例。宗芳芳等[20]提出二次定位策略技术，首先从程序中抽象出函数调用图，用基于模型的方法对可能存在故障的函数进行可疑排序，最后利用 DStar 定位函数中故障的代码行；Wotawa 等[19]利用约束求解的技术将程序自动编译成一组约束，实现模型检测定位，优点是减少了可疑语句的搜索域

范围，缺点是不同程序往往具有不同的异常行为。因此，并没有一个完整的模型可以满足所有可能的故障模式以及故障类型之间的关联。

1.2.3　其它故障定位技术

除了上面讨论的技术之外，近年来还出现了很多其它的软件故障定位技术。Souza 等[21]提出使用覆盖率数据集成技术来定位故障，通过创建可疑语句排名列表的技术进行定位。以信息检索技术为切入点的研究也取得了很多新的研究成果[22-23,25]。这些研究使用基于源代码的搜索得到故障相关报告的降序等级排列，可以帮助开发人员检查被标注的文件是否包含故障。算法调试也是一个创新性的故障定位技术，Silva[25]提出将复杂计算分解为一系列子计算帮助故障定位。公式故障定位技术[26-29]则依赖于对编码失败的程序公式执行轨迹跟踪，该技术使用特定工具或算法公式帮助程序员捕捉导致失败的相关语句。

2　主流程序集

表 2 总结了当前在故障定位研究中常用的程序集，分别从各程序集的名称、代码行数、简述以及语言类型作简要展示，程序集多来源于 SIR[32] 和 SourceForge[33]。

表 2 展示了故障定位中常用的程序集，设计各个方面的程序测试，其中的 Siemens 程序集是最常用的程序集，该程序集由 7 个程序组成，程序结构多样化，通用性较强，但是每个子程序仅有不到 600 行代码（不包括空白行），而且是针对 C 语言程序定位的，因此后续出现了 grep、flex 等数万行代码的程序集，而且新增了 Java 和 C++ 语言，极大地方便了各种语言下的定位应用。

表 2　主流程序集

	名称	代码行数	简述	语言		名称	代码行数	简述	语言
1	Siemens：tcas	173	报警与防撞系统	C	21	Ant	75 333	Java 程序构造器	Java
2	Siemensschedule	412	优先级调度	C	22	XML－sec	21 613	XML 库文件加密	C
3	Siemens：print_tokens	565	词法分析程序	C	23	Unix：Look	170	查找单词或按字典列表排序	C
4	Siemens：replace	563	模式识别	C	24	Unix：Comm	167	选择或消除两个排序文件的相同行	C
5	Siemens：print_tokens2	510	词法分析程序	C	25	tar	25 854	创建文件档案的工具	C
6	Siemens：schedule2	307	优先级调度	C	26	DC	2 700	逆波兰台式计算器	Java
7	Siemens：tot_info	406	信息度量	C	27	Unix：Crypt	134	通过用户提供的密码加密、解密文件	C
8	grep	12 653	命令行实用程序搜索纯文本数据集	C	28	Unix：Sort	913	文件排序及合并	C
9	space	9 126	ADL 解析器	C	29	gcc	222 196	GNU C 编译器	C
10	gzip	6 573	数据压缩	C	30	apache	85 661	Web 应用程序托管	C
11	sed	12 062	GNU 批编辑器	C	31	schoolmate	4 263	基于 PHP / MySQL 的解决方案	PHP
12	flex	13 892	词法分析生成器	C	32	faqforge	734	创建和管理文档的工具	PHP
13	NanoXML	7 646	XML 解析器	Java	33	webchess	2 226	在线棋类游戏	JS、PHP
14	Unix：Cal	202	日历打印	C	34	jtopas	5 400	文本解析器	Java
15	Unix：Col	308	反向过滤器线	C	35	timeclock	13 879	基于 Web 的时钟系统	C
16	Unix：Tr	137	字符转换	C	36	phpsysinfo	7 745	显示系统信息，如运行时间、CPU、内存等	C
17	Unix：Spline	338	基于给定数据插入光滑曲线	C	37	TCC	1 900	C 语言编译器	C
18	Unix：Uniq	143	报告或删除相邻重复行	C	38	Xerces	52 528	XML 解析器	C++
19	Unix：Chckeq	102	报告分隔符缺失或成对关系不平衡	C	39	Mozilla firefox	3.4M	Web 浏览器	C、C++
20	make	20 014	管理构建代码可执行的其它产品	C	40	tidy	31 132	编辑 Web 内容的文本编辑器	C++

3　未来研究方向展望

现有的故障定位技术已取得了不少研究成果，但是还有很多结果有待完善，如多故障定位技术[35-36]、测试用例套件质量[37-38]以及巧合正确性带来的影响[39]。

3.1　多故障定位

对于故障数量及其相关性有几个常见假设，这些假设

往往基于某种技术效果显而易见，但是换一种定位技术或者换一种实验程序集则不奏效，常见假设如下：①假设程序只有一个故障。实际应用中程序通常包含多个故障，因此需质疑基于单个故障场景的定位效果，并尽量研究多故障并存下的故障定位效果；②假设即使程序中有多个故障，这些故障也都独立执行。这一假设也在近年的研究中被质疑，因为同一个程序的不同故障很有可能相互作用，相长作用的结果往往会触发新故障，相消作用的结果则往往掩盖掉已有的故障；③假设调试一轮即能找到一个故障。通常程序经过一次调试后被程序员修复了一个故障，然而再次运行修改之后的程序不能保证不会对其它故障造成影响，因此被改动之后的程序往往不能当作测试用例集中的一员再次运行，而是要重新启动其它测试用例。

3.2 测试用例套件

常见的定位技术多用于测试用例的执行状态，如执行成功状态或者执行失败状态来讨论对应计数所需寻找的信息，关注点几乎都在测试用例的执行结果上。因此，一个良好的测试用例套件对程序的有效定位具有重要意义。关于测试用例套件重要性的表现如下：

（1）测试用例套件会影响故障定位技术的有效性。实验证明，现阶段采用的同时关注多条测试用例取得的定位效果往往要高于仅关注导致程序出现故障的一条失败测试用例和多条成功测试用例的效果，因为在多条测试用例的对比下，可以提供更加详细的故障线索。因此，如何生成一个自动化的测试用例套件仍然需要进一步探讨。

（2）不能实现高覆盖的测试用例套件对目标程序的故障定位结果可能造成不利影响。为了减少这类影响，可用禁忌搜索技术自动生成一个测试用例套件，以达到最大的分支覆盖或者对原始测试用例集约简后，确保原有故障定位的准确性并保持较高约简比。

（3）对给定的被测程序使用全套测试用例未必有效。如果合理减少测试用例数量，选择测试用例子集或者给其分配一定的优先级，可能定位效率更高。分配测试用例优先级的技术是给予导致执行失败的测试用例优先执行，因为这样可以提供更多信息，减少搜索域。

3.3 巧合正确性

巧合正确性即在各种程序状态下，本来执行结果失败的测试用例由于特殊原因导致执行结果仍然是正确的。这种现象发生的原因很多，例如给定一个存在故障的语句，对语句变量分配不正确的值，测试执行过程中，该值可能会影响程序的输出导致失败。然而在测试不断执行的过程中，该变量的值可能因为被覆盖而未影响到程序输出，或者根本没有触发到故障所在的位置，因此在实际操作中要考虑巧合正确性的存在，以减少定位误差。

4 结语

随着软件结构复杂化与软件规模的扩大化，软件故障定位需要更多的时间投资和资源投资，定位程序故障不再是简单的机械过程，而是朝着自动化、智能化的方向发展。在以往的实践中，对软件的调试工作往往由具有丰富经验的程序员手动调试，如果判断失误，则采用回退技术返回起点再次调试，这种机械枯燥的操作使程序员急切需要一种能快速有效定位的技术。当务之急是将定位技术融入到实际应用中，结合新兴技术的发展趋势以及程序员的经验和应用能力，根据特定场景应用相应的定位技术。本文梳理了当前的最新定位研究成果，分析了常见的以及最新流行的定位技术和当前的主流程序集，最后针对当前出现的主要问题，提出了若干值得研究的方向。

参考文献：

[1] SCHOOOP G，LAMINA J. On the effectiveness of the tarantula fault localization technique for different fault classes[J]. IEEE International Symposium on High-assurance Systems Engineering，2011，69(5)：317-324.

[2] 王勇. 软件过程资产库的研究与实现[J]. 计算机应用与软件，2016(7)：106-108.

[3] ZHOU X，WANG H，ZHAO J. A fault-localization approach based on the coincidental correctness probability[C]. IEEE International Conference on Software Quality，Reliability and Security，2015.

[4] WRIGHT C S，ZIA T A. A quantitative analysis into the economics of correcting software bugs[C]. Computational Intelligence in Security for Information Systems，International Conference，Held at Iwann 2011，Torremolinos-Málaga，Spain，2011：198-205.

[5] 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，IEEE Computer Society，2002：467-477.

[6] WONG W E，DEBROY V，GAO R，et al. The DStar method for effective software fault localization[J]. IEEE Transactions on Reliability，2014，63(1)：290-308.

[7] XIE X，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 & Methodology，2013，22(4)：402-418.

[8] KAVITHA P. An integrated method for program proving，testing and debugging[J]. Middle East Journal of Scientific Research，2014，20(12)：2310-2315.

[9] JIANG B，CHAN W K，TSEe T H. On practical adequate test suites for integrated test case prioritization and fault localization[C]. International Conference on Quality Software. IEEE Computer Society，2011：21-30.

[10] WEISER M D. Program slices：formal，psychological，and practical investigations of an automatic program abstraction method[M]. University of Michigan，1979.

[11] ZHANG X，NEELAM G，RAJIV G. Locating faulty code by multiple points slicing[J]. Software Practice & Experience，2007，37(9)：935-961.

[12] KOREL B. PELAS-program error-locating assistant system[J]. IEEE Transactions on Software Engineering，1988，14(9)：1253-1260.

[13] KOREL B，LASKI J. STAD-a system for testing and debugging：user perspective[C]. The Workshop on Software Testing，1988：13-20.

[14] 杨书新，徐丽萍，王鹏. 基于图挖掘和决策树的软件故障定位研究[J]. 计算机工程与应用，2015，51(20)：67-71.

[15] DUCASSé M，FERRé S，RIDOUX O，et al. Multiple fault localization with data mining[J]. Seke，2011.

[16] ZHANG S，ZHANG C. Software bug localization with markov logic[C]. Companion Proceedings of the 36th International Conference on Software Engineering. ACM，2014：424-427.

[17] BAAH G K，PODURSKI A，HARROLD M J. The probabilistic program dependence graph and its application to fault diagnosis[J]. IEEE Transactions on Software Engineering，2009，36(4)：528-545.

[18] GRIESMAYER A，STABER S，BLOEM R. Fault localization using a model checker[J]. Software Testing Verification & Reliability，2010，20(2)：149-173.

[19] NICA M，NICA S，WOTAWA F. On the use of mutations and testing for debugging[J]. Software Practice & Experience，2013(43)：1121-1142.

[20] 宗芳芳，黄鸿云，丁佐华. 基于二次定位策略的软件故障定位[J]. 软件学报，2016(8)：1993-2007.

[21] DE SOUZA H A，LORDELLO CHAIM M. Adding context to fault localization with integration coverage[C]. Ieee/acm，International Conference on Automated Software Engineering，2013：628-633.

[22] RAO S，KAK A. Retrieval from software libraries for bug localization：a comparative study of generic and composite text models[C]. International Working Conference on Mining Software Repositories，MSR. 2011：43-52.

[23] SAHA R K，LEASE M，KHURSHID S，et al. Improving bug localization using structured information retrieval[C]. Ieee/acm，International Conference on Automated Software Engineering. 2013：345-355.

[24] ZHOU J，ZHANG H，LO D. Where should the bugs be fixed? more accurate information retrieval-based bug localization based on bug reports[C]. International Conference on Software Engineering，2012：14-24.

[25] SILVA J. A survey on algorithmic debugging strategies[J]. Advances in Engineering Software，2011，42(11)：976-991.

[26] CHRIST J，ERMIS E，SCHÄF M，et al. Flow-sensitive fault localization[M]. Verification，Model Checking，and Abstract Interpretation. Springer Berlin Heidelberg，2013：189-208.

[27] ERMIS E，SCHÄF M，WIES T. Error invariants[M]. FM 2012：Formal Methods. Springer Berlin Heidelberg，2012：187-201.

[28] 施小燕. 基于条件概率公式的故障定位问题研究[D]. 南京：南京大学，2013.

[29] JOSE M，MAJUMDAR R. Cause clue clauses：error localization using maximum satisfiability[J]. Acm Sigplan Notices，2011，46(6)：437-446.

[30] RENIERES M，REISS S P. Fault localization with nearest neighbor queries[C]. IEEE International Conference on Automated Software Engineering，2003：30-39.

[31] 黄燕勤. 有关基于频谱和聚类方法定位故障的实证研究[D]. 南京：南京大学，2015.

[32] Software-artifact infrastructure repository[EB/OL] http://sir.unl.edu/portal/index.html.

[33] Sourceforge[EB/OL]http://sourceforge.net.

[34] LUCIA，THUNG F，LO D，et al. Are faults localizable[C]. IEEE Working Conference on Mining Software Repositories，2012：74-77.

[35] DIGIUSEPPE N，JONES J A. On the influence of multiple faults on coverage-based fault localization[C]. International Symposium on Software Testing and Analysis. ACM，2011：556-559.

[36] ARTZI S，DOLBY J，TIP F，et al. Practical fault localization for dynamic web applications[C]. ACM/IEEE，International Conference on Software Engineering. IEEE，2010：265-274.

[37] JIANG B，CHAN W K，TSE T H. On practical adequate test suites for integrated test case prioritization and fault localization[C]. International Conference on Quality Software. IEEE Computer Society，2011：21-30.

[38] ZHANG Z，CHAN W K，TSE T H. Fault localization based only on failed runs[J]. Computer，2012，45(6)：64-71.

[39] MASRI W，ASSI R A. Cleansing test suites from coincidental correctness to enhance fault-localization[C]. International Conference on Software Testing，Verification and Validation，ICST 2010，Paris，France，2010：165-174.

（责任编辑：黄　健）

Research Summary for the Key Technology of Software Fault Localization

Abstract：Software fault location is designed by using the information of programs andtests to improve the quality of software. By giving a suspicious range of the wrong statements can easily locate the source of detect. With the complication of software structure and the expansion of the software scale，the number of running states is growing exponentially. Only rely on manual localization technology for troubleshooting will not work，so more and more auxiliary positioning technologies are proposed now. In order to explore the practicability of these technologies，this paper firstly introduces the traditional and the latest promotion of fault localization technologies in recent years；and then summarizes the common test procedure；finally，key scientific problems which need further researched are summarized.

Key Words：Fault Localization；Program Debugging；Software Testing；Execution Trace；Suspicious Code