北京大学

硕士研究生学位论文

题目：测试文档

姓　　名：某某

学　　号：0123456789

院　　系：某某学院

专　　业：某某专业

研究方向：某某方向

导　　师：某某教授

某年某月

版权声明

摘要

关键词：其一，其二

Test Document

Test (Some Major)

Directed by Prof. Somebody

ABSTRACT

Test of the English abstract.

KEYWORDS: First, Second

目录

序言

第一章　研究背景

1.1　缺陷定位技术

随着软件的发展，生活中越来越多的方面都与软件有着紧密的关系。小到人们的日常出行、购物、餐饮等，大到航空航天、医药等领域，软件在人们的生活中扮演着重要的角色。随着软件的应用领域的扩大，软件的复杂性上升，提升了软件缺陷的可能性。软件缺陷可能会导致巨大的损失。一个著名的被广泛引用的例子，是在海湾战争时，一颗导弹由于导航软件的精度缺陷而偏离了目标，导致28人死亡和100人受伤{TODO:cite}。美国国家标准与技术研究院(NIST)2002年发表的一篇报告({TODO:cite})显示，软件缺陷每年会导致约595亿美元的经济损失。{TODO:每年这么多国内公司漏洞事件} 发现并修复软件缺陷，保障软件的高质量成为一项重要的任务。

在发现软件缺陷之后，开发人员为了解决这个缺陷往往需要三步[23]。第一步，缺陷定位，需要找到程序中和这个缺陷有关的语句。第二步，理解缺陷，明白为什么会发生缺陷。第三步，修复缺陷，修改代码以让缺陷消失。这三个步骤合起来就是调试的过程。缺陷定位作为调试的第一步，其完成速度和准确性对后面的步骤有着很大的影响。在传统的开发环境当中，人们可以手动调试来定位缺陷，比如插入断点、打印日志信息等等。在1989年Collofello等人就指出尝试去减少软件中的错误会花费50%到80%的开发和维护的精力[7]。随着软件的复杂性的上升，手动地定位软件缺陷将会耗费更多开发者的时间和精力。为了提高定位缺陷的速度，研究人员对自动化的缺陷定位展开了研究，并取得了巨大的进展{TODO:cite}。然而在2011年，Partin和Osro的一篇调查[23]通过研究缺陷定位技术在实际应用场景下的效果，发现以往的评价指标并不能准确的反映缺陷定位技术在实际应用中的效果。以往的缺陷定位技术是基于一系列关于开发人员会如何调试的假设，而这些假设在实际场景的某些情况下会失效。自动化缺陷定位技术还有很大的发展空间。

1.1.1　自动缺陷定位技术概述

Weiser在1981年提出的程序切片[26, 27]是自动调试（特别是缺陷定位）最早的技术之——。给定一个程序P和一个在P的语句s中使用的变量v，程序切片会找到P中所有可能会影响s中v的值的语句。如果s中v的值是错误的，那么导致这个错误的错误语句一定在这个切片当中。也就是说，不在这个切片当中语句可以在调试过程中被忽略。尽管程序切片已经减少了可能出错的语句的数量，但是切片中的语句的数量仍然比较

大。为了解决这个问题，Korel和Laski在1988年提出了动态程序切片[17]。动态程序切片计算某一个特定执行的切片。后来又有很多的动态程序切片的变种被提出[9, 11, 31, 32]，用于解决调试问题，并且产生了大量研究工作[3, 4, 15, 16, 20, 21, 28]。

为了解决程序切片调试方法的短板，一种通过观察错误程序的执行特征和正确程序的执行特征的调试技术被提出。这些技术通过收集程序执行信息，观察不同的某种特征，来定位缺陷。比如使用路径概要[25]，反例[5, 10]，语句覆盖[14]和谓词值[18, 19]等等。

本文根据北京大学熊英飞研究员对缺陷定位的分类[29]，将缺陷定位分为以下几类。

- 基于切片的缺陷定位
- 基于频谱的缺陷定位
- 基于状态覆盖的缺陷定位
- 基于变异的缺陷定位
- 基于构造正确执行状态的缺陷定位
- 基于算法式调试的缺陷定位
- 基于差异化调试的缺陷定位

本文的研究内容主要根据基于频谱的缺陷定位和基于状态覆盖的缺陷定位。

1.1.2 基于频谱的缺陷定位

基于频谱的缺陷定位是使用最广泛的自动化缺陷定位方法[29]。程序频谱(Program Spectrum)最早由Reps等人于1997年提出[25]，用于解决千年虫问题。Harrold等人在2002年[12]提出使用测试覆盖信息作为频谱信息的调试方法。 Renieris等人在2003年提出使用成功的测试用例和失败的测试用例进行缺陷定位[24]，奠定了此后基于频谱的缺陷定位的基础。

考虑一种极端的情况。比如当某一个语句s被执行的之后，测试用例就会失败。而成功的测试用例都不会执行语句s。那么语句s很有可能就是导致缺陷的语句。找出所有这样的语句s就可以大幅减少需要排查错误的语句。但是，在实际的代码中这种极端的情况很少出现。对于一个出错的语句s，它很可能既被失败的测试用例执行，也被成功的测试用例执行。因为一个语句在其不同的上下文作用下会产生不同的效果。简单地计算成功的测试用例覆盖的语句和失败的测试用例覆盖的语句的差集是无法准确找出错误语句的。利用成功的测试用例覆盖的语句的交集和并集，与失败的测试用例覆盖的语句取差集，是最早的一种基于频谱的缺陷定位方法[24]。这种方法也隐含着基于频谱的缺陷定位的假设：被失败的测试用例执行的语句，更有可能有错误。而被成

a_{ef}	一个语句被失败的测试用例覆盖的次数
a_{nf}	一个语句未被失败的测试用例覆盖的次数
a_{ep}	一个语句被成功的测试用例覆盖的次数
a_{np}	一个语句未被成功的测试用例覆盖的次数
a_f	失败的测试用例的个数
a_p	成功的测试用例执行的次数

表 1.1　基于频谱的错误定位的数学符号及其意义

功的测试用例执行的语句，更有可能是正确的。

为方便此后的表述，引入一些数学符号，见表1.1。表中的统计量就是程序频谱。

Jones等人提出的Tarantula[14]，直观地展示给开发者展示了每个语句在成功的测试用例和失败的测试用例下的参与情况。每条语句的参与情况，使用公式

$$\text{Tarantula}(s) = \frac{\frac{a_{ep}}{a_p}}{\frac{a_{ep}}{a_p} + \frac{a_{ef}}{a_f}}$$

计算。这个公式计算的值也被称为怀疑度。怀疑度更高的语句会在怀疑列表更靠前的位置。相比于交集并集差集的方法，Tarantula在Siemens数据集上可以将错误的语句放在怀疑列表更前面的位置[13]。

Tarantula之后，又有很多计算怀疑度的公式被提出。 Ochiai由Abreu等人提出[1]。

$$\text{Ochiai}(s) = \frac{a_{ef}}{\sqrt{a_f \times (a_{ef} + a_{ep})}}$$

Ochiai由[22]提出用于计算基因的相似度。 Abreu等人将其引入用于计算怀疑度，并与Jaccard[6]，Tarantula，AMPLE[8]比较，发现Ochiai计算的怀疑度使得定位效果更好[1, 2]。

谢晓园等人在理论上证明了不存在单一最佳公式[30]。

1.1.3　基于状态覆盖的缺陷定位

结论

参考文献

[1] R Abreu, P Zoeteweij and A. J. C Van Gemund. "*An Evaluation of Similarity Coefficients for Software Fault Localization*". In: *Pacific Rim International Symposium on Dependable Computing*, **2006**: 39–46.

[2] R Abreu, P Zoeteweij and A. J. C Van Gemund. "*On the Accuracy of Spectrum-based Fault Localization*". In: *Testing: Academic and Industrial Conference Practice and Research Techniques - Mutation*, **2007**: 89–98.

[3] Hiralal Agrawal, Richard A. Demillo and Eugene H. Spafford. *Debugging with dynamic slicing and backtracking*, **1993**: 589–616.

[4] Elton Alves, Milos Gligoric, Vilas Jagannath *et al.* "*Fault-localization using dynamic slicing and change impact analysis*". In: *Ieee/acm International Conference on Automated Software Engineering*, **2011**: 520–523.

[5] Thomas Ball, Mayur Naik and Sriram K Rajamani. "*From symptom to cause: localizing errors in counterexample traces*". *Acm Sigplan Notices*, **2003**, *38*(1): 97–105.

[6] Mike Y. Chen, Emre Kiciman, Eugene Fratkin *et al.* "*Pinpoint: Problem Determination in Large, Dynamic Internet Services*". In: *Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on*, **2002**: 595–604.

[7] James S. Collofello and Scott N. Woodfield. *Evaluating the effectiveness of reliability-assurance techniques*. Elsevier Science Inc., **1989**: 191–195.

[8] Valentin Dallmeier, Christian Lindig and Andreas Zeller. *Lightweight Defect Localization for Java.* Springer Berlin Heidelberg, **2005**: 528–550.

[9] Richard A. Demillo, Hsin Pan and Eugene H. Spafford. "*Critical slicing for software fault localization*". In: **1996**: 121–134.

[10] Alex Groce, Daniel Kroening and Flavio Lerda. "*Understanding Counterexamples with explain*". *In Computer-Aided Verification*, **2004**, *3114*: 453–456.

[11] Gyim, Tibor Thy, Besz *et al.* "*An efficient relevant slicing method for debugging*". *Acm Sigsoft Software Engineering Notes*, **1999**, *24*(6): 303–321.

[12] Mary Jean Harrold, Gregg Rothermel, Kent Sayre *et al.* "*An empirical investigation of the relationship between spectra differences and regression faults*". *Software Testing Verification & Reliability*, **2000**, *10*(3): 171–194.

[13] James A. Jones and Mary Jean Harrold. "*Empirical evaluation of the tarantula automatic fault-localization technique*". In: *Ieee/acm International Conference on Automated Software Engineering*, **2005**: 273–282.

[14] Jones, A James, Harrold *et al.* "*Visualization of test information to assist fault localization*". *Biochemical Engineering Journal*, **2002**, *24*(2): 115–123.

[15] Xiaolin Ju, Shujuan Jiang, Xiang Chen *et al.* "*HSFal: Effective fault localization using hybrid spectrum of full slices and execution slices*". *Journal of Systems & Software*, **2014**, *90*(1): 3–17.

[16] Z. A. Al-Khanjari, M. R. Woodward, Haider Ali Ramadhan *et al.* "*The Efficiency of Critical Slicing in Fault Localization*". *Software Quality Journal*, **2005**, *13*(2): 129–153.

[17] Bogdan Korel and Janusz Laski. "*Dynamic program slicing ☆*". *Information Processing Letters*, **1988**, *29*(3): 155–163.

[18] Ben Liblit, Mayur Naik, Alice X. Zheng *et al.* "*Scalable statistical bug isolation*". In: **2005**: 15–26.

[19] Chao Liu, Xifeng Yan, Long Fei *et al.* "*SOBER: statistical model-based bug localization*". In: *European Software Engineering Conference Held Jointly with ACM Sigsoft International Symposium on Foundations of Software Engineering*, **2005**: 286–295.

[20] Chao Liu, Xiangyu Zhang, Jiawei Han *et al.* "*Indexing Noncrashing Failures: A Dynamic Program Slicing-Based Approach*". In: *IEEE International Conference on Software Maintenance*, **2007**: 455–464.

[21] Xiaoguang Mao, Yan Lei, Ziying Dai *et al.* "*Slice-based statistical fault localization ☆*". *Journal of Systems & Software*, **2014**, *89*(1): 51–62.

[22] Meyer, Andréia Da Silvagarcia, Antonio Augusto Francosouza *et al.* "*Comparison of similarity coefficients used for cluster analysis with dominant markers in maize (Zea mays L)*". *Genetics & Molecular Biology*, **2004**, *27*(1): 83–91.

[23] Chris Parnin and Alessandro Orso. "*Are automated debugging techniques actually helping programmers?*" In: *International Symposium on Software Testing and Analysis*, **2011**: 199–209.

[24] M Renieres and S. P Reiss. "*Fault localization with nearest neighbor queries*". In: *IEEE International Conference on Automated Software Engineering, 2003. Proceedings*, **2003**: 30–39.

[25] Thomas Reps, Thomas Ball, Manuvir Das *et al.* "*The use of program profiling for software maintenance with applications to the year 2000 problem*". *Acm Sigsoft Software Engineering Notes*, **1997**, *22*(6): 432–449.

[26] Mark Weiser. "*Program Slicing*". *IEEE Transactions on Software Engineering*, **1984**, *SE-10*(4): 352–357.

[27] Mark Weiser. "*Program slicing*". In: *International Conference on Software Engineering*, **1981**: 439–449.

[28] Franz Wotawa. "*Fault Localization Based on Dynamic Slicing and Hitting-Set Computation.*" In: *International Conference on Quality Software*, **2010**: 161–170.

[29] Yingfei Xiong. *Fault Localization*, **2018**. `http://sei.pku.edu.cn/~xiongyf04/SA/2017/18_fault_localization.pdf`, retrieved on 2018-04-07.

[30] Shin Yoo, Xiaoyuan Xie, Fei-Ching Kuo *et al.* "*No pot of gold at the end of program spectrum rainbow: Greatest risk evaluation formula does not exist*". *RN*, **2014**, *14*(14): 14.

[31] Xiangyu Zhang, Neelam Gupta and Rajiv Gupta. "*Pruning dynamic slices with confidence*". *Acm Sigplan Notices*, **2006**, *41*(6): 169–180.

[32] Xiangyu Zhang, R Gupta and Youtao Zhang. “*Precise dynamic slicing algorithms*”. In: *International Conference on Software Engineering, 2003. Proceedings*, **2003**: 319–329.

附录 A　附件

致谢

北京大学学位论文原创性声明和使用授权说明

原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品或成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本声明的法律结果由本人承担。

论文作者签名：　　日期：　　年　　月　　日

学位论文使用授权说明

（必须装订在提交学校图书馆的印刷本）

本人完全了解北京大学关于收集、保存、使用学位论文的规定，即：

- 按照学校要求提交学位论文的印刷本和电子版本；
- 学校有权保存学位论文的印刷本和电子版，并提供目录检索与阅览服务，在校园网上提供服务；
- 学校可以采用影印、缩印、数字化或其它复制手段保存论文；
- 因某种特殊原因需要延迟发布学位论文电子版，授权学校在□一年/□两年/□三年以后在校园网上全文发布。

（保密论文在解密后遵守此规定）

论文作者签名：　　导师签名：　　日期：　　年　　月　　日