UNIVERSITY OF SCIENCE AND TECHNOLOGY BEIJING

北京科技大学

拟攻读博士学位科研计划

姓　　名: 代贺鹏

报考专业: 计算机科学与技术

导　　师: 孙昌爱

1 研究方向与目标

围绕基于频谱的故障技术没有充分利用成功的测试用例执行信息这一问题展开研究，提出从测试用例生成到测试再到故障定位的一整套技术并利用机器学习技术辅助推断故障的位置。

2 研究背景与意义

计算机软件在日常的生活中占据越来越重要的地位并且对保密、安全系统也极其重要。与此同时软件的规模与复杂程度也急剧增加。由此软件中的故障可能越来越多，这些故障可能产生严重的事故或者带来巨大的经济损失 [19] [20] [32]。因此对软件可靠性问题的研究日益得到了人们的广泛重视 [28]。软件中的故障通常由程序故障引起，程序故障是指程序在运行过程中出现的一种不正常的内部状态 [30]。故障定位是确定程序中可疑故障的确切性质和位置，为故障修复做准备。程序故障定位的有效性受到程序规模、程序设计语言和开发人员能力等因素的影响 [21]，导致程序故障定位成为调试过程中一项耗时费力的工作 [28]。为了降低开发维护成本，帮助软件测试人员找到软件故障，研究自动化故障定位技术具有十分重要的现实意义。

1987 年 Collofello 和 Cousins 在 [7]中提出可以利用频谱进行故障定位。从此之后基于频谱的故障定位技术受到了广泛的关注，是目前研究最多的故障定位技术 [28]。许多基于频谱的故障定位技术是从概率模型或者统计模型中受到启发，它的主要思想是：一个程序频谱详细地记录了程序执行过程中的某种信息，例如条件分支 [12]，当执行失败时类似的信息可以帮助调试人员识别可疑的语句。

早期的基于频谱的故障定位技术 [1] [15] [16] [25]仅利用失效的测试用例的执行信息导致基于频谱的故障定位技术的效率可能不高 [3] [13]。后来将成功的测试用例以及失效的测试用例的执行信息一起考虑取得了较好的结果 [23]。直觉上，如果一条语句的执行模式和所有失效的测试用例的执行模式越接近，那么这条语句越可能是故障语句，因此该语句的怀疑度就越高。反之，这条语句越不可能是故障语句，怀疑度越低。基于系数的度量方法可以用来量化这种“远近”，量化的结果可以用来解释某一条语句可能存在故障的怀疑度。Tarantula [14]技术是运用这一直觉的典型技术并且取得了较好的效果。

目前大多数基于频谱的故障定位技术没有定量失败执行信息与成功执行信息分别对找到故障语句的贡献度。Xie 在 [33]中将程序中所有代码分成可疑和不可疑两组代码。可疑的代码至少被一个失败的测试用例执行到，程序中剩下的代码为不可疑的代码。并且只对可疑的代码组进行风险评估，不可疑的代码组分配一个最低的风险值。然而成功的测试用例也可能执行了含有故障的语句。因此对成功的测试用例组进行风险评估可疑进一步提高定位故障的效率。

另一方面机器学习技术能够在研究人员很少的干预下通过数据产生模型。因此可以将成功和失败的测试用例的执行信息作为输入数据利用机器学习的技术辅助学习或者推断故障的位置。

3 国内外研究现状

传统的故障定位技术有：日志 [11]、断言 [24]、断点 [9]、剖面 [4]。随着软件复杂度以及规模不断扩大，传统的故障定位技术的定位效率不断下降。利用事件/原因(程序中存在缺陷)和现象/效果(执行失败)这一哲学上的因果关系 [17]，许多新的故障定位技术被提出。在 [22]中提到了许多的因果关系模型，例如：概率模型、统计模型、因果关系计算模型，其中概率模型是最广泛被使用的理论模型用来评价代码的可疑度。

程序切片技术旨在利用某种规约将原始程序精简成较小的程序。自从 1979 Weiser第一个提出静态切片 [27] 之后已经有几百篇类似的文章发表 [34] [26] [5]。程序切片技术的一个最重要的应用是降低故障的搜索域。Lyle 和 Weiser 通过构造程序块 [18]进一步降低故障的搜索域。然而静态切片一个明显的缺点是：某一个变量的静态切片包含了可能影响该变量值的所有可执行语句，由于静态切片不能确定执行时的信息，因此该切片很有可能包含一些执行不到的语句。动态切片 [2] 技术可以很好的弥补这一缺陷。在 [31] 中 Wotawa 将动态切片技术与基于模型的诊断技术相结合提出了更高效的故障定位技术。Zhang 将后向动态切片、前向动态切片以及交叉动态切片技术集中提出了多点动态切片技术 [35]。

程序频谱详细的记录了程序的某种信息，例如分支覆盖以及执行的语句信息，该技术可以用来追踪程序的行为举止。Collofello 和 Cousins 在 [8]建议可以用频谱技术辅助故障定位。早期的基于频谱的故障定位技术 [1] [15] [16] [25]仅利用失效的测试用例的执行信息导致基于频谱的故障定位技术的效率可能不高 [3] [13]。后来将成功的测试用例以及失效的测试用例的执行信息一起考虑取得了较好的结果 [23]。Tarantula [14]是基于频谱的一个知名的故障定位技术。它利用覆盖以及测试用的执行结果：通过或者没有通过计算每一条语句的怀疑度。在 [10] 中Debroy 等人更进一步修正了 Tarantula 技术：将多个失败测试用例执行相同的语句分为一组，然后按照测试用例数目对分区进行降序排名，再利用 Tarantula 技术对每一组中的语句进行可疑度计算。由此可以看出成功的测试用例的执行信息没有得到充分的利用，但是成功的测试用例的执行语句很有可能包含故障语句。充分提取成功测试用例的执行信息可以进一步提高基于频谱的故障定位技术。

机器学习技术能够在研究人员很少的干预下通过数据产生模型。因此可以将成功和失败的测试用例的执行信息作为输入数据利用机器学习的技术辅助学习或者推断故障的位置。Wong 等人基于反向传播的神经网络技术提出了一种

新的故障定位技术 [29]。反向传播的神经网络技术易于实现，并且可以处理复杂的非线性函数。将测试用例的执行信息以及对应的测试结果作为反向传播的神经网络的训练集，其输出可以作为某一个居于包含故障的可能性。Briand 等人在 [6] 中利用 C4.5 决策树算法 将执行失败的测试用例分组。决策树中的每一条路径都表示一种故障模式。利用 Tarantula 技术对每一个组中的测试用例涉及到的语句进行可疑度计算。

4 研究内容与思路

目前基于频谱的故障定位技术没有充分利用成功的测试用例的执行信息，但是成功的测试用例很有可能包含故障语句。因此进一步从成功的测试用例一中提取相关信息可以进一步提高基于频谱的古战定位技术的效率。另一方面机器学习以及数据挖掘技术可以在较少的人类干预下分析失败以及成功的测试用例的执行信息以及对应结果之间的关系，从而可以辅助故障定位。并且目前这方面的研究还很少。因此本人希望从测试用例生成到软件测试再到故障定位做一整套技术。其中故障定位方面用机器学习的方法充分利用成功的测试用例执行信息以及失败的测试用例执行信息，通过设置两者恰当的比例，得到更可靠的怀疑语句序列。

参考文献

[1] Hiralal Agrawal, Richard A. Demillo, and Eugene H. Spafford. An execution backtracking approach to program debugging. pages 283–299, 1991.

[2] Hiralal Agrawal, Richard A DeMillo, and Eugene H Spafford. Debugging with dynamic slicing and backtracking. *Software: Practice and Experience*, 23(6):589–616, 1993.

[3] Hiralal Agrawal, Joseph R Horgan, Saul London, and W Eric Wong. Fault localization using execution slices and dataflow tests. In *Software Reliability Engineering, 1995. Proceedings., Sixth International Symposium on*, pages 143–151. IEEE, 1995.

[4] Thomas Ball and James R Larus. Optimally profiling and tracing programs. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 16(4):1319–1360, 1994.

[5] David Binkley and Mark Harman. A survey of empirical results on program slicing. *Advances in Computers*, 62:105–178, 2004.

[6] Lionel C Briand, Yvan Labiche, and Xuetao Liu. Using machine learning to support debugging with tarantula. In *Software Reliability, 2007. ISSRE'07. The 18th IEEE International Symposium on*, pages 137–146. IEEE, 2007.

[7] James S. Collofello and Larry Cousins. Towards automatic software fault location through decision-to-decision path analysis. In *afips*, page 539, 1987.

[8] Larry Dean Cousin. Towards automatic software fault location through decision-to-decision path analysis. Master's thesis, Arizona State University, 1986.

[9] Deborah S Coutant, Sue Meloy, and Michelle Ruscetta. Doc: A practical approach to source-level debugging of globally optimized code. *ACM SIGPLAN Notices*, 23(7):125–134, 1988.

[10] Vidroha Debroy, W Eric Wong, Xiaofeng Xu, and Byoungju Choi. A grouping-based strategy to improve the effectiveness of fault localization techniques. In *Quality Software (QSIC), 2010 10th International Conference on*, pages 13–22. IEEE, 2010.

[11] Jermaine Charles Edwards. Method, system, and program for logging statements to monitor execution of a program, March 25 2003. US Patent 6,539,501.

[12] Mary Jean Harrold, Gregg Rothermel, Kent Sayre, Rui Wu, and Liu Yi. An empirical investigation of the relationship between spectra differences and regression faults. *Software Testing Verification & Reliability*, 10(3):171–194, 2015.

[13] James A Jones and Mary Jean Harrold. Empirical evaluation of the tarantula automatic fault-localization technique. In *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*, pages 273–282. ACM, 2005.

[14] James A Jones, Mary Jean Harrold, and John T Stasko. Visualization for fault localization. In *in Proceedings of ICSE 2001 Workshop on Software Visualization*, 2001.

[15] Bogdan Korel. Pelas-program error-locating assistant system. *IEEE Transactions on Software Engineering*, 14(9):1253–1260, 1988.

[16] Bogdan Korel and Janusz Laski. Stad-a system for testing and debugging: User perspective. In *Software Testing, Verification, and Analysis, 1988., Proceedings of the Second Workshop on*, pages 13–20. IEEE, 1988.

[17] David Lewis. Causation. *The journal of philosophy*, 70(17):556–567, 1974.

[18] James R Lyle. Automatic program bug location by program slicing. In *The Second International Conference on Computers and Applications*, pages 877–883, 1987.

[19] J. C. Munson and T. M. Khoshgoftaar. The detection of fault-prone programs. *IEEE Transactions on Software Engineering*, 18(5):423–433, 1992.

[20] Ganesh J. Pai and Joanne Bechta Dugan. Empirical analysis of software fault content and fault proneness using bayesian methods. *IEEE Transactions on Software Engineering*, 33(10):675–686, 2007.

[21] MICHAEL P. PAPAZOGLOU, PAOLO TRAVERSO, SCHAHRAM DUSTDAR, and FRANK LEYMANN. Service-oriented computing: A research roadmap. *International Journal of Cooperative Information Systems*, 17(02):223–255, 2008.

[22] Judea Pearl. Causality: models, reasoning and inference. *Econometric Theory*, 19(675-685):46, 2003.

[23] Manos Renieres and Steven P Reiss. Fault localization with nearest neighbor queries. In *Automated Software Engineering, 2003. Proceedings. 18th IEEE International Conference on*, pages 30–39. IEEE, 2003.

[24] David S. Rosenblum. A practical approach to programming with assertions. *IEEE Transactions on software engineering*, 21(1):19–31, 1995.

[25] A-B Taha, Stephen M Thebaut, and S-S Liu. An approach to software fault localization and revalidation based on incremental data flow analysis. In *Computer Software and Applications Conference, 1989. COMPSAC 89., Proceedings of the 13th Annual International*, pages 527–534. IEEE, 1989.

[26] Frank Tip. *A survey of program slicing techniques*. Centrum voor Wiskunde en Informatica, 1994.

[27] Mark David Weiser. Program slices: formal, psychological, and practical investigations of an automatic program abstraction method. 1979.

[28] W. Eric Wong, Ruizhi Gao, Yihao Li, Abreu Rui, and Franz Wotawa. A survey on software fault localization. *IEEE Transactions on Software Engineering*, 42(8):707–740, 2016.

[29] W Eric Wong and Yu Qi. Bp neural network-based effective fault localization. *International Journal of Software Engineering and Knowledge Engineering*, 19(04):573–597, 2009.

[30] Michaelf. Worboys, Hilarym. Hearnshaw, and Davidj. Maguire. Object-oriented data modelling for spatial databases. *International Journal of Geographical Information Systems*, 4(4):369–383, 1990.

[31] Franz Wotawa. Fault localization based on dynamic slicing and hitting-set computation. In *Quality Software (QSIC), 2010 10th International Conference on*, pages 161–170. IEEE, 2010.

[32] Craig S. Wright and Tanveer A. Zia. *A Quantitative Analysis into the Economics of Correcting Software Bugs*. Springer Berlin Heidelberg, 2011.

[33] Xiaoyuan Xie, Tsong Yueh Chen, and Baowen Xu. Isolating suspiciousness from spectrum-based fault localization techniques. In *Quality Software (QSIC), 2010 10th International Conference on*, pages 385–392. IEEE, 2010.

[34] Baowen Xu, Ju Qian, Xiaofang Zhang, Zhongqiang Wu, and Lin Chen. A brief survey of program slicing. *ACM SIGSOFT Software Engineering Notes*, 30(2):1–36, 2005.

[35] Xiangyu Zhang, Neelam Gupta, and Rajiv Gupta. Locating faulty code by multiple points slicing. *Software: Practice and Experience*, 37(9):935–961, 2007.