批评 Par 所属论文的算法和实验中的不足,进而引申到如何划定程序修复问题和评价算法的探讨中.在后续的讨论中,Monperrus 申明了基于测试集的程序修复研究的内容(即,修复算法核心为:在给定测试集的情况下获得高质量补丁,而低质量的测试集并非算法缺陷)和算法评价标准(即,算法比较应基于同样的数据集和同样的缺陷类型)等.尽管研究者对此文评价不一,但该文开启了基于测试集的程序修复研究自2008年以来持续6年研究后的第1次大规模学术争论.研究者们开始关注非算法部分,并深入探索问题的本质和修复实例的基础.

同是2014年,Martinez等人[47]和Barr等人[48]在互不知情的情况下,同时发表了两篇关于程序修复基础的文章.Martinez等人[47]发表了题为《代码修复的原料存在吗?》(do the fix ingredients already exist?)的文章,以实证调查(empirical inquiry)的形式探索了冗余假设(redundancy assumption)的存在性.所谓的"冗余假设"是指自动生成的补丁一定存在于程序的其他某处,而修复算法是将其他位置的代码重用(reuse)或组合(mix)而形成补丁的.早期算法,如 GenProg,AE,RSRepair 等,都基于该假设.文献[47]表明:冗余假设确实一定程度地存在着,但并非完全存在.几乎同一时间,Barr等人[48]发表了题为《整形外科手术猜想》(the plastic surgery hypothesis)的文章,探索程序修复的基础.所谓"整形外科手术猜想"与"冗余假设"类似,是指补丁中的代码能够从程序的其他位置(像植皮技术一样)移动到修复 bug 的位置.该文设计了复杂的实验,深入地讨论了猜想的存在性.

4.3 修复真实bug的能力

修复真实 bug 一直是程序修复研究的目标.以此为基础,程序修复进而才可能完成工业应用.然而,这一研究进程出现了波折.如第 4.1 节所述,Le Goues 等人[42]首次组织了 105 个真实的 C 程序 bug 的数据集,并应用 GenProg 算法评估修复能力.此后,多个研究者以此数据集作为实验平台,研究自动修复的相关问题.

直至2015年,Qi等人[10]发表了题为《貌似可信和正确的补丁的分析》(an analysis of patch plausibility and correctness)的文章,手工检查了早期算法在105个真实的C程序bug的数据集上的修复结果.结果发现:由于实验设置错误,在GenProg文章[42]报道的55个可修复的bug中,有37个为修复后的程序是无法通过全部测试用例的;而在AE文章[20]报道的54个可修复的bug中,有27个为修复后的程序是无法通过全部测试用例的.此外,基于人工验证:在全部105个bug中,GenProg的修复结果只有2个是语义正确的,而AE的结果只有3个是语义正确的;其他的修复均无法满足原有程序的需求说明.文献[20]的发表在领域内引起轩然大波,相关研究者不得不重新检验相关结果的正确性.在2014年的第1次学术争论之后,该文引发了领域内的第2次集体争论,进而导致研究者深度关注对于修复真实bug的可行性的研究.

同年,Smith等人[11]发表了题为《适得其反?》(is the cure worse than the disease?)的文章,研究修复算法的过拟合(overfitting).该文设计了可控实验,通过新开发者引入的bug和补丁,分析影响GenProg和RSRepair算法效果的因素,并指出修复算法对于测试用例的过拟合行为.同年,Le Goues等人[49]介绍了两个C语言的被修复程序集合 ManyBugs 和 IntroClass,共包含 1 141 个缺陷.基于该数据集,他们设计了量化实验,评估已有方法的修复效果.

仍是2015年,Durieux等人[26]选取了3种经典算法的Java版本,在Java语言的真实bug上进行修复实验.这3种算法是GenProg,Kali(Java版本[50])和Nopol[24,25];而数据集是包含222个bug的Defects4J[51].实验结果表明:在面向对象的程序中,Nopol能够修复35个bug,而GenProg和Kali分别修复27个和22个;而从语义正确的角度来看,Nopol可以正确地修复5个,而GenProg和Kali分别正确地修复了5个和1个.该结果表明:修复算法的当前研究仍处于初级阶段,与实际应用尚存一定距离.

4.4 小 结

修复真实 bug 是自动程序修复方法通向真实应用的必经之路.自该领域研究初期开始,如何精确而高效地修复真实 bug 一直是研究核心之一.由于程序自身的复杂性,实证研究是发现和检验修复真实 bug 能力的主要方法.然而,正如本节所介绍,经历了两次领域内的集体学术争论之后,自动修复的实证研究在波折之中继续前进.由本节内容可以得到如下发现:

(1) 随着对自动修复的基础的探索,更多的细节被深入挖掘,实证研究可在一定程度上为程序修复提供事

实依据;

(2) 作为自动修复的核心内容之一,修复真实的 bug 仍未曾被研究者攻克.更进一步地,生成符合程序语义的补丁十分困难;

(3) 由于不同算法源自不同的研究者,建立公平的算法比较还存在一定困难.其中之一就是从真实项目提取的被修复 bug 数量有限,不足以满足现实 bug 的自然分布.

5 程序修复相关技术

5.1 非基于测试集的修复方法的近期工作简介

除了基于测试集的修复方法,研究者们也关注其他形式的修复方法,例如形式化修复[52]和动态程序状态修复[53]等.下面对近期的一些工作进行简要的介绍.

Dallmeier 等人[54]提出了 Pachika,一种检测程序对象非正常行为(object behavior anomaly detection)的修复方法.该方法识别两个对象间的不同,进而增加或者删除方法调用.Carzaniga 等人[55,56]开发了一种用于避免 Web 应用错误的自动方法.该方法旨在找到可运行的程序变种(program variant),作为自动的临时变通版本(automatic workaround).Pei 等人[57]设计了 AutoFix,一个基于契约的修复(contract based repair)系统.该方法需要程序契约中的规约信息作为输入,例如函数的前置和后置条件等,可用于 Effiel 语言的 bug 修复.Tan 和 Roychoudhury[58]的近期工作 Relifix 针对程序回归中的问题进行修复.

Sidiroglou-Douskos 等人[59]提出了 CodePhage,一种通过从被修复应用中定位缺陷,并从其他应用迁移代码(code transfer)来消除缺陷的算法.该算法依赖于作为输入的应用中能消除缺陷且成熟的代码,可修复指定缺陷,如整数溢出、缓冲区溢出及零除数等.Raychev 等人[60]提出了 JSNice,一种面向 JavaScript 语言的自动程序美化工具.该工具通过从海量代码库学习代码特性,为程序预测变量名或生成一定的注释.该工具的特点是自动化,且生成的代码及注释具有一定程度的语义信息.另外一组相关工作是测试用例修复(test case repair)[61−64],旨在自动修复由于代码更新造成的测试用例无法正常运行的问题[65].

5.2 程序修复的相关技术的近期工作简介

定位 bug 和获得更好的测试集,是自动程序修复算法中两个相关步骤.下面简单介绍这两方面的近期相关工作.

故障定位旨在通过收集测试用例运行信息,推断潜在的 bug 所处的位置.自动修复算法中借助已有的故障定位技术,根据疑似故障的风险给出语句的排列,进而逐个尝试修复.基于程序谱的故障定位(spectrum based fault localization)方法是其中一类常见的方法.故障定位的成果浩如烟海,这里只列出部分近期工作.

Zhang 等人[66]设计了只依赖于失败测试用例的故障定位方法,有效地减少了测试用例的运行成本.Xie 等人[3]从理论上分析了基于程序谱的故障定位方法中风险函数(risk formula)之间的关系;Lucia 等人[67]从实证研究的角度给出了风险函数间的关联.Masri 和 Assi[68]针对故障定位中的巧合正确性(coincidental correctness)问题,分析了缓解影响定位准确率因素的措施.Xuan 和 Monperrus[69]通过机器学习技术,组合已有的风险函数,进一步提升故障定位的准确率.另外的一些近期工作包括 Jiang 等人[70]、Debroy 和 Wong[71]、Xu 等人[72]、Xuan 等人[73]的工作等.

测试用例增强(test case augmentation)是近期涌现的一系列工作.尽管测试用例生成技术的研究历史悠久,但受限于测试路径组合爆炸及面向对象程序的复杂状态等问题,测试用例生成技术尚不能满足所有应用的需求.测试用例增强旨在基于给定的测试用例,获得更好的测试效果,如生成可以弥补当前测试用例不足的新测试用例或更好地使用当前测试用例.

Xu 等人[74,75]提出了早期的直接测试用例增强(directed test suite augmentation)技术,该方法在当前测试用例集的基础上生成新的测试用例,进而提高测试用例的覆盖率等指标.Yoo 和 Harman[76]设计了测试重生成(test data regeneration)算法.该方法对于一个测试集,基于遗传算法生成全新的测试用例,用以弥补已有测试用例的

不足.Xuan 和 Monperrus[77]提出了测试用例提纯(test case purification)方法.该方法能够分离执行失败的测试用例,并切分为多个子测试用例,优化测试用例的执行,进而增强其识别故障的能力.

6 机遇与挑战

本文回顾了基于测试集的自动程序修复的研究进展.自动程序修复技术分析测试用例,搜索潜在的程序补丁,以通过全部测试用例集.在充分调研相关文献之后,从自动程序修复方法和实证研究基础两个方面详细介绍了该领域的近期成果.在自动修复方法方面,分 3 个类别依年份介绍了算法的发展历程;在实证研究方面,详述了实证研究的成果和潜在问题.基于测试集的自动程序修复的研究仅有 8 年,领域历史较短但成果丰富.

前文详述了领域的研究进展,下面简要分析自动程序修复面临的机遇和挑战:

(1) 修复算法的指引:在搜索程序补丁的过程中,自动程序修复算法往往并没有得到足够的指引.例如,早期的 GenProg 等算法实际上是借助于启发式规则寻找可能存在的补丁;后续的 SemFix 等算法虽然能够从测试用例提取出约束以缩减搜索空间,但仍遗留大量的潜在的不正确的补丁.因此,如何利用被修复程序代码结构和已知的测试用例指引自动算法搜索补丁,是该领域仍存在的重要问题;

(2) 多点 bug 修复:目前的自动修复算法均以单点 bug(single-point bug)为修复对象,即,程序中的 bug 只包含于一条语之中.尽管单点 bug 的修复研究尚不成熟,修复多点 bug(multi-point bug,即 bug 存在于多条语句之中)仍具有重要的科研价值.多点 bug 的修复并非简单地组合单点 bug 的修复算法:仅仅更新一条语句不能通过全部测试用例,因此,修复算法无法知晓当前操作的正确性.已有的多故障定位技术为多点 bug 修复提供了一定的思路和技术支持;

(3) 补丁定位:为了修复 bug,自动算法首先通过故障定位技术将潜在的 bug 位置按照疑似包含 bug 的可能性排序.尽管故障定位的研究已经发展了 15 年,但精确定位 bug 仍然面临着一系列难题.而自动程序修复的研究既要面对故障定位领域原有的困难,又为故障定位方法的改进提供了机遇;

(4) 测试集的利用:测试集是自动修复中的重要输入.如何基于当前的测试集获得更多的测试用例以及如何更好地利用当前测试用例,是利用测试集改进程序修复的途径之一.程序修复将为测试用例生成的相关技术提供应用场景和发展机遇;而测试用例生成也将为改进程序修复方法提供支撑.

References:

[1] Pressman RS. Software Engineering: A Practitioner's Approach. 7th ed., New York: McGraw-Hill, 2010. 437−443.

[2] Xie X, Chen TY, Kuo FC, Xu B. A theoretical analysis of the risk evaluation formulas for spectrum-based fault localization. ACM Trans. on Software Engineering and Methodology, 2013,22(4):31:1−31:40. [doi: 10.1145/2522920.2522924]

[3] Wen WZ, Li BX, Sun XB, Liu CC. Technique of software fault localization based on hierarchical slicing spectrum. Ruan Jian Xue Bao/Journal of Software, 2013,24(5):977−992 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4342.htm [doi: 10.3724/SP.J.1001.2013.04342]

[4] Xuan J, Jiang H, Ren Z, Zou W. Developer prioritization in bug repositories. In: Proc. of the 34th Int'l Conf. on Software Engineering (ICSE). 2012. 25−35. [doi: 10.1109/ICSE.2012.6227209]

[5] Le Goues C, Nguyen T, Forrest S, Weimer W. GenProg: A generic method for automatic software repair. IEEE Trans. on Software Engineering, 2012,38:54−72. [doi: 10.1109/TSE.2011.104]

[6] Kim D, Nam J, Song J, Kim S. Automatic patch generation learned from human-written patches. In: Proc. of the ACM/IEEE Int'l Conf. on Software Engineering (ICSE). 2013. 802−811. [doi: 10.1109/ICSE.2013.6606626]

[7] Harman M, Mansouri A, Zhang Y. Search based software engineering: Trends, techniques and applications. ACM Computing Surveys, 2012,45(1):11:1−11:61. [doi: 10.1145/2379776.2379787]

[8] Weimer W, Nguyen T, Le Goues C, Forrest S. Automatically finding patches using genetic programming. In: Proc. of the Int'l Conf. on Software Engineering (ICSE). 2009. 364−367. [doi: 10.1109/ICSE.2009.5070536]

[9] Monperrus M. A critical review of automatic patch generation learned from human-written patches: Essay on the problem statement and the evaluation of automatic software repair. In: Proc. of the 36th Int'l Conf. on Software Engineering (ICSE). 2014. 234−242. [doi: 10.1145/2568225.2568324]

[10] Qi Z, Long F, Achour S, Rinard M. An analysis of patch plausibility and correctness for generate-and-validate patch generation systems. In: Proc. of the Int'l Symp. on Software Testing and Analysis (ISSTA). 2015. 24−36. [doi: 10.1145/2771783.2771791]

[11] Smith EK, Barr E, Le Goues C, Brun Y. Is the cure worse than the disease? Overfitting in automated program repair. In: Proc. of the European Software Engineering Conf. and ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering (ESEC/FSE). 2015. 532−543. [doi: 10.1145/2786805.2786825]

[12] Qi Y, Mao X, Lei Y. Making automatic repair for large-scale programs more efficient using weak recompilation. In: Proc. of the IEEE Int'l Conf. on Software Maintenance (ICSM). 2012. 254−263. [doi: 10.1109/ICSM.2012.6405280]

[13] Qi Y, Mao X, Wen Y, Dai Z, Gu B. More efficient automatic repair of large-scale programs using weak recompilation. Science China Information Sciences, 2012,55(12):2785−2799. [doi: 10.1007/s11432-012-4741-1]

[14] Qi Y, Mao X, Lei Y. Efficient automated program repair through fault-recorded testing prioritization. In: Proc. of the IEEE Int'l Conf. on Software Maintenance (ICSM). 2013. 180−189. [doi: 10.1109/ICSM.2013.29]

[15] Qi Y, Mao X, Lei Y, Wang C. Using automated program repair for evaluating the effectiveness of fault localization techniques. In: Proc. of the Int'l Symp. on Software Testing and Analysis (ISSTA). 2013. 191−201. [doi: 10.1145/2483760.2483785]

[16] Qi Y, Mao X, Lei Y, Dai Z, Wang C. The strength of random search on automated program repair. In: Proc. of the 36th Int'l Conf. on Software Engineering (ICSE). 2014. 254−265. [doi: 10.1145/2568225.2568254]

[17] Zhong H, Su Z. An empirical study on real bug fixes. In: Proc. of the 37th Int'l Conf. on Software Engineering (ICSE). 2015. 913−923. [doi: 10.1109/ICSE.2015.101]

[18] Harman M, Jones B. Search based software engineering. Journal of Information and Software Technology, 2011,43(14):833−839.

[19] Jiang H, Xuan J, Ren Z. Approximate backbone based multilevel algorithm for next release problem. In: Proc. of the 12th Annual Conf. on Genetic and Evolutionary Computation (GECCO). 2010. 1333−1340. [doi: 10.1145/1830483.1830730]

[20] Weimer W, Fry ZP, Forrest S. Leveraging program equivalence for adaptive program repair: Models and first results. In: Proc. of the IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE). 2013. 356−366. [doi: 10.1109/ASE.2013.6693094]

[21] Debroy V, Wong WE. Using mutation to automatically suggest fixes for faulty programs. In: Proc. of the 3rd Int'l Conf. on Software Testing, Verification and Validation (ICST). 2010. 65−74. [doi: 10.1109/ICST.2010.66]

[22] Nguyen HDT, Qi D, Roychoudhury A, Chandra S. SemFix: Program repair via semantic analysis. In: Proc. of the Int'l Conf. on Software Engineering (ICSE). 2013. 772−781. [doi: 10.1109/ICSE.2013.6606623]

[23] Mechtaev S, Yi J, Roychoudhury A. DirectFix: Looking for simple program repairs. In: Proc. of the 37th Int'l Conf. on Software Engineering (ICSE). 2015. 448−458. [doi: 10.1109/ICSE.2015.63]

[24] DeMarco F, Xuan J, Le Berre D, Monperrus M. Automatic repair of buggy if conditions and missing preconditions with SMT. In: Proc. of the 6th Int'l Workshop on Constraints in Software Testing, Verification, and Analysis (CSTVA). 2014. 30−39. [doi: 10.1145/2593735.2593740]

[25] Xuan JF, Martinez M, DeMarco F, Clément M, Lamelas-Marcote S, Durieux T, Le Berre D, Monperrus M. Nopol: Automatic repair of conditional statement bugs in Java programs. Technical Report, INRIA, 2015. 1−22.

[26] Martinez T, Durieux T, Xuan JF, Monperrus M. Automatic repair of real bugs in Java: A large-scale experiment on the Defects4J dataset. Technical Report, arXiv:1505.07002, ArXiv, 2015. 1−11.

[27] Arcuri A, Yao X. A novel co-evolutionary approach to automatic software bug fixing. In: Proc. of the IEEE Congress on Evolutionary Computation (CEC). 2008. 162−168. [doi: 10.1109/CEC.2008.4630793]

[28] Forrest S, Weimer W, Nguyen T, Le Goues C. A genetic programming approach to automated software repair. In: Proc. of the Genetic and Evolutionary Computing Conf. (GECCO). 2009. 947−954. [doi: 10.1145/1569901.1570031]

[29] Weimer W, Forrest S, Le Goues C, Nguyen T. Automatic program repair with evolutionary computation. Communications of the ACM, 2010,53(5):109−116. [doi: 10.1145/1735223.1735249]

[30] Fast E, Le Goues C, Forrest S, Weimer W. Designing better fitness functions for automated program repair. In: Proc. of the Genetic and Evolutionary Computing Conf. (GECCO). 2010. 965−972. [doi: 10.1145/1830483.1830654]

[31] Le Goues C, Weimer W, Forrest S. Representations and operators for improving evolutionary software repair. In: Proc. of the Genetic and Evolutionary Computation Conf. (GECCO). 2012. 959−966. [doi: 10.1145/2330163.2330296]

[32] Schulte E, Forrest S, Weimer W. Automated program repair through the evolution of assembly code. In: Proc. of the ACM/IEEE Int'l Conf. on Automated Software Engineering (ASE). 2010. 313−316. [doi: 10.1145/1858996.1859059]

[33] Martinez M, Monperrus M. Mining software repair models for reasoning on the search space of automated program fixing. Empirical Software Engineering, 2015,20(1):176−205. [doi: 10.1007/s10664-013-9282-8]

[34] Long F, Rinard M. Prophet: Automatic patch generation via learning from successful patches. Technical Report, MIT-CSAIL-TR-2015-027, CSAIL MIT, 2015. 1−11. http://hdl.handle.net/1721.1/97735

[35] Jia Y, Harman M. An analysis and survey of the development of mutation testing. IEEE Trans. on Software Engineering, 2011, 37(5):649−678. [doi: 10.1109/TSE.2010.62]

[36] Long F, Rinard M. Staged program repair with condition synthesis. In: Proc. of the 10th Joint Meeting of the European Software Engineering Conf. and the ACM SIGSOFT Symp. on the Foundations of Software Engineering (ESEC/FSE). 2015. 166−178. [doi: 10.1145/2786805.2786811]

[37] Jha S, Gulwani S, Seshia SA, Tiwari A. Oracle guided component-based program synthesis. In: Proc. of the ACM/IEEE Int'l Conf. on Software Engineering (ICSE). 2010. 215−224. [doi: 10.1145/1806799.1806833]

[38] Jones JA, Harrold MJ. Empirical evaluation of the tarantula automatic fault-localization technique. In: Proc. of the 20th IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE). 2005. 273−282. [doi: 10.1145/1101908.1101949]

[39] Zhang X, Gupta N, Gupta R. Locating faults through automated predicate switching. In: Proc. of the 28th Int'l Conf. on Software Engineering. 2006. 272−281. [doi: 10.1145/1134285.1134324]

[40] Abreu R, Zoeteweij P, Gemund AGV. On the accuracy of spectrum-based fault localization. In: Proc. of the Testing: Academic and Industrial Conf. on Practice and Research Techniques (Mutation). 2007. 89−98. [doi: 10.1109/TAIC.PART.2007.13]

[41] Ke Y, Stolee KT, Le Goues C, Brun Y. Repairing programs with semantic code search. In: Proc. of the IEEE/ACM Conf. on Automated Software Engineering (ASE). 2015. 295−306. [doi: 10.1109/ASE.2015.60]

[42] Le Goues C, Dewey-Vogt M, Forrest S, Weimer W. A systematic study of automated program repair: Fixing 55 out of 105 bugs for $8 each. In: Proc. of the ACM/IEEE Int'l Conf. on Software Engineering (ICSE). 2012. 3−13. [doi: 10.1109/ICSE.2012.6227211]

[43] Fry ZP, Landau B, Weimer W. A human study of patch maintainability. In: Proc. of the Int'l Symp. on Software Testing and Analysis (ISSTA). 2012. 177−187. [doi: 10.1145/2338965.2336775]

[44] Le Goues C, Forrest S, Weimer W. Current challenges in automatic software repair. Software Quality Journal, 2013,21(3):421−443. [doi: 10.1007/s11219-013-9208-0]

[45] Tao Y, Kim J, Kim S, Xu C. Automatically generated patches as debugging aids: A human study. In: Proc. of the 22nd ACM SIGSOFT Int'l Symp. on the Foundations of Software Engineering (FSE). 2014. 64−74. [doi: 10.1145/2635868.2635873]

[46] Tao Y, Han D, Kim S. Writing acceptable patches: An empirical study of open source project patches. In: Proc. of the 30th Int'l Conf. on Software Maintenance and Evolution (ICSME). 2014. 271−280. [doi: 10.1109/ICSME.2014.49]

[47] Martinez M, Weimer W, Monperrus M. Do the fix ingredients already exist? An empirical inquiry into the redundancy assumptions of program repair approaches. In: Proc. of the 36th Int'l Conf. on Software Engineering (ICSE). 2014. 492−495. [doi: 10.1145/2591062.2591114]

[48] Barr ET, Brun Y, Devanbu PT, Harman M, Sarro F. The plastic surgery hypothesis. In: Proc. of the 22nd ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering (FSE). 2014. 306−317. [doi: 10.1145/2635868.2635898]

[49] Le Goues C, Holtschulte N, Smith EK, Brun Y, Devanbu P, Forrest S, Weimer W. The ManyBugs and IntroClass benchmarks for automated repair of C programs. IEEE Trans. on Software Engineering, 2015. [doi: 10.1109/TSE.2015.2454513]

[50] Martinez M, Monperrus M. Astor: Evolutionary automatic software repair for Java. Technical Report, arXiv:1410.6651, ArXiv, 2015. 1−6.

[51] Just R, Jalali D, Ernst MD. Defects4J: A database of existing faults to enable controlled testing studies for Java programs. In: Proc. of the Int'l Symp. on Software Testing and Analysis (ISSTA). 2014. 437−440. [doi: 10.1145/2610384.2628055]

[52] Jobstmann B, Griesmayer A, Bloem R. Program repair as a game. In: Proc. of the Computer Aided Verification (CAV). 2005. 226−238. [doi: 10.1007/11513988_23]

[53] Perkins JH, Kim S, Larsen S, Amarasinghe S, Bachrach J, Carbin M, Pacheco C, Sherwood F, Sidiroglou-Douskos S, Sullivan G, Wong WF, Zibin Y, Ernst MD, Rinard M. Automatically patching errors in deployed software. In: Proc. of the ACM Symp. on Operating Systems Principles (SOSP). 2009. 87−102. [doi: 10.1145/1629575.1629585]

[54] Dallmeier V, Zeller A, Meyer B. Generating fixes from object behavior anomalies. In: Proc. of the IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE). 2009. 550−554. [doi: 10.1109/ASE.2009.15]

[55] Carzaniga A, Gorla A, Mattavelli A, Perino N, Pezzè M. Automatic recovery from runtime failures. In: Proc. of the ACM/IEEE Int'l Conf. on Software Engineering (ICSE). 2013. 782−791. [doi: 10.1109/ICSE.2013.6606624]

[56] Carzaniga A, Gorla A, Mattavelli A, Perino N, Pezzè M. Automatic workarounds for Web applications. In: Proc. of the ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering (FSE). 2010. 237−246. [doi: 10.1145/1882291.1882327]

[57] Pei Y, Furia CA, Nordio M, Wei Y, Meyer B, Zeller A. Automated fixing of programs with contracts. IEEE Trans. on Software Engineering, 2014,40(5):427−449. [doi: 10.1109/TSE.2014.2312918]

[58] Tan SH, Roychoudhury A. Relifix: Automated repair of software regressions. In: Proc. of the Int'l Conf. on Software Engineering (ICSE). 2015. 471−482. [doi: 10.1109/ICSE.2015.65]

[59] Sidiroglou-Douskos S, Lahitnen E, Long F, Rinard M. Automatic error elimination by horizontal code transfer across multiple applications. In: Proc. of the 36th ACM SIGPLAN Conf. on Programming Language Design and Implementation (PLDI). 2015. 43−54. [doi: 10.1145/2737924.2737988]

[60] Raychev V, Vechev M, Krause A. Predicting program properties from "big code". In: Proc. of the 42nd Annual ACM SIGPLAN/SIGACT Symp. on Principles of Programming Languages (POPL). 2015. 111−124. [doi: 10.1145/2676726.2677009]

[61] Daniel B, Jagannath V, Dig D, Marinov D. Reassert: Suggesting repairs for broken unit tests. In: Proc. of the Int'l Conf. on Automated Software Engineering (ASE). 2009. 433−444. [doi: 10.1109/ASE.2009.17]

[62] Mirzaaghaei M, Pastore F, Pezzè M. Supporting test suite evolution through test case adaptation. In: Proc. of the 5th Int'l Conf. on Software Testing, Verification and Validation (ICST). 2012. 231−240. [doi: 10.1109/ICST.2012.103]

[63] Mirzaaghaei M, Pastore F, Pezzè M. Automatic test case evolution. Software Testing, Verification and Reliability, 2014,24(5):386−411. [doi: 10.1002/stvr.1527]

[64] Gao Z, Chen Z, Zou Y, Memon A. Sitar: GUI test script repair. IEEE Trans. on Software Engineering, 2016,42(2):170−186. [doi: 10.1109/TSE.2015.2454510]

[65] Zhang ZY, Chen ZY, Xu BW, Yang R. Research progress on test case evolution. Ruan Jian Xue Bao/Journal of Software, 2013, 24(4):663−674 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4379.htm [doi: 10.3724/SP.J.1001.2013.04379]

[66] Zhang Z, Chan WK, Tse TH. Fault localization based only on failed runs. IEEE Computer, 2012,45(6):64−71. [doi: 10.1109/MC.2012.185]

[67] Lucia L, Lo D, Jiang L, Thung F, Budi A. Extended comprehensive study of association measures for fault localization. Journal of Software: Evolution and Process, 2014,26(2):172−219. [doi: 10.1002/smr.1616]

[68] Masri W, Assi RA. Prevalence of coincidental correctness and mitigation of its impact on fault localization. ACM Trans. on Software Engineering Methodology, 2014,23(1):8:1−8:28. [doi: 10.1145/2559932]

[69] Xuan JF, Monperrus M. Learning to combine multiple ranking metrics for fault localization. In: Proc. of the 30th Int'l Conf. on Software Maintenance and Evolution (ICSME). 2014. 191−200. [doi: 10.1109/ICSME.2014.41]

[70] Jiang B, Zhai K, Chan WK, Tse TH, Zhang Z. On the adoption of MC/DC and control-flow adequacy for a tight integration of program testing and statistical fault localization. Journal of Information and Software Technology, 2013,55:897−917. [doi: 10.1016/j.infsof.2012.10.001]

[71] Debroy V, Wong WE. Combining mutation and fault localization for automated program debugging. Journal of Systems and Software, 2014,90:45−60. [doi: 10.1016/j.jss.2013.10.042]

[72] Xu J, Zhang Z, Chan WK, Tse TH, Li S. A general noise-reduction framework for fault localization of Java programs. Journal of Information and Software Technology, 2013,55:880−896. [doi: 10.1016/j.infsof.2012.08.006]

[73] Xuan JF, Xie X, Monperrus M. Crash reproduction via test case mutation: Let existing test cases help. In: Proc. of the 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE). 2015. 910−913. [doi: 10.1145/2786805.2803206]

[74] Xu Z, Kim Y, Kim M, Rothermel G, Cohen MB. Directed test suite augmentation: Techniques and tradeoffs. In: Proc. of the 18th ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering (FSE). 2010. 257−266. [doi: 10.1145/1882291.1882330]

[75] Xu Z, Cohen MB, Rothermel G. Factors affecting the use of genetic algorithms in test suite augmentation. In: Proc. of the 12th Annual Conf. on Genetic and Evolutionary Computation (GECCO). 2010. 1365−1372. [doi: 10.1145/1830483.1830734]

[76] Yoo S, Harman M. Test data regeneration: Generating new test data from existing test data. Journal of Software Testing, Verification and Reliability, 2012,22(3):171−201. [doi: 10.1002/stvr.435]

[77] Xuan JF, Monperrus M. Test case purification for improving fault localization. In: Proc. of the 22nd ACM SIGSOFT Int'l Symp. on the Foundations of Software Engineering (FSE). 2014. 52−63. [doi: 10.1145/2635868.2635906]

附中文参考文献:

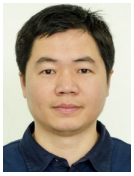
[3] 文万志,李必信,孙小兵,刘翠翠.一种基于层次切片谱的软件错误定位技术.软件学报,2013,24(5):977−992. http://www.jos.org.cn/1000-9825/4342.htm [doi: 10.3724/SP.J.1001.2013.04342]

[65] 张智轶,陈振宇,徐宝文,杨瑞.测试用例演化研究进展.软件学报,2013,24(4):663−674. http://www.jos.org.cn/1000-9825/4379.htm [doi: 10.3724/SP.J.1001.2013.04379]

玄跻峰(1984－),男,黑龙江哈尔滨人,博士,研究员,CCF 会员,主要研究领域为软件分析与测试,软件数据分析,基于搜索的软件工程.

谢晓园(1983－),女,博士,教授,CCF 会员,主要研究领域为软件测试调试,程序分析,基于搜索的软件工程.

任志磊(1984－),男,博士,讲师,CCF 会员,主要研究领域为演化计算,基于搜索的软件工程.



江贺(1980－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为基于搜索的软件工程,软件仓库挖掘.

王子元(1982－),男,博士,副教授,CCF 会员,主要研究领域为软件测试.