软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2013,24(8):1695−1712 [doi: 10.3724/SP.J.1001.2013.04420]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

回归测试中的测试用例优先排序技术述评*

陈 翔[1,2], 陈继红[1], 鞠小林[1], 顾 庆[2]

[1](南通大学 计算机科学与技术学院,江苏 南通 226019)

[2](计算机软件新技术国家重点实验室(南京大学),江苏 南京 210093)

通讯作者: 陈翔, E-mail: xchencs@ntu.edu.cn

摘 要: 测试用例优先排序(test case prioritization,简称 TCP)问题是回归测试研究中的一个热点.通过设定特定排序准则,对测试用例进行排序以优化其执行次序,旨在最大化排序目标,例如最大化测试用例集的早期缺陷检测速率.TCP 问题尤其适用于因测试预算不足以致不能执行完所有测试用例的测试场景.首先对 TCP 问题进行描述,并依次从源代码、需求和模型这 3 个角度出发对已有的 TCP 技术进行分类;然后对一类特殊的 TCP 问题(即测试资源感知的 TCP 问题)的已有研究成果进行总结;随后依次总结实证研究中常用的评测指标、评测数据集和缺陷类型对实证研究结论的影响;接着依次介绍 TCP 技术在一些特定测试领域中的应用,包括组合测试、事件驱动型应用测试、Web 服务测试和缺陷定位等;最后对下一步工作进行展望.

关键词: 回归测试;测试用例优先排序;贪心法;元启发式搜索;实证研究

中图法分类号: TP311 **文献标识码**: A

中文引用格式: 陈翔,陈继红,鞠小林,顾庆.回归测试中的测试用例优先排序技术述评.软件学报,2013,24(8):1695−1712. http://www.jos.org.cn/1000-9825/4420.htm

英文引用格式: Chen X, Chen JH, Ju XL, Gu Q. Survey of test case prioritization techniques for regression testing. Ruan Jian Xue Bao/Journal of Software, 2013,24(8):1695−1712 (in Chinese). http://www.jos.org.cn/1000-9825/4420.htm

Survey of Test Case Prioritization Techniques for Regression Testing

CHEN Xiang[1,2], CHEN Ji-Hong[1], JU Xiao-Lin[1], GU Qing[2]

[1](School of Computer Science and Technology, Nantong University, Nantong 226019, China)

[2](State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210093, China)

Corresponding author: CHEN Xiang, E-mail: xchencs@ntu.edu.cn

Abstract: Test case prioritization (TCP) issue is a hot research topic in regression testing research. This method tries to optimize the execution schedule based on a specific prioritization criterion. The purpose of the TCP techniques is to maximize a specific prioritization objective, such as the early fault detection rate of the original test suite. This technique is especially applied to some testing scenarios, for example testing resource is limited for executing all the test cases. This paper first describes the issue of TCP and classifies the existing TCP techniques into three categories: source code, requirement, and model. The paper secondly formulates a specific TCP issue (i.e., resource-aware TCP issue) and summarizes its research work. The paper finally summarizes commonly-used evaluation metrics and subjects in experimental studies, and empirical result affection of different fault injection types. The paper fourthly summarizes the application of TCP in some specific testing domains, such as combinatorial testing, event-driven applications testing, fault localization, and Web services testing and discusses some future work of the TCP issue.

* 基金项目: 国家自然科学基金(60873027, 61202006); 国家高技术研究发展计划(863)(2006AA01Z177); 国家重点基础研究发展计划(973)(2009CB320705); 江苏省高校自然科学研究项目(12KJB520014); 江苏省研究生培养创新工程(CXZZ120935); 南通市应用研究计划(BK2012023); 南京大学计算机软件新技术国家重点实验室开放课题(KFKT2012B29);

收稿时间: 2013-01-21; 修改时间: 2013-03-29; 定稿时间: 2012-04-22; jos 在线出版时间: 2013-05-23

CNKI 网络优先出版: 2013-05-23 15:18, http://www.cnki.net/kcms/detail/11.2560.TP.20130523.1518.002.html

© 中国科学院软件研究所 http://www.jos.org.cn

Key words: regression testing; test case prioritization; greedy algorithm; meta-heuristic search; empirical study

软件产品在开发和维护过程中,因移除软件内在缺陷、完善已有功能、重构已有代码或提高运行性能等,需要执行代码修改并触发软件演化.随着以统一过程和敏捷方法为代表的增量、迭代式开发过程的流行,软件演化频率也随之迅速提高并亟需经济、有效的测试方法来确保演化后软件产品的质量.回归测试作为一种有效的方法,可有效保证代码修改的正确性并避免代码修改对被测程序其他模块产生的副作用.但统计数据表明,回归测试一般占软件产品测试预算的80%以上,占软件维护预算的50%以上[1,2].为了大幅度削减这部分开销,国内外研究人员对自动高效的回归测试技术展开了深入研究,其中,测试用例维护策略的设定是一个核心问题.一种简单维护策略是重新执行已有的所有测试用例,但存在如下问题:

(1) 在一些测试场景中,若测试用例数量较多或单个测试用例执行开销较大,则项目实际预算不允许执行完所有测试用例.例如,Rothermel 等人在某一合作企业内,在测试一个包含约 20 000 行代码的软件产品时发现,运行所有测试用例所需的时间长达 7 周[3].

(2) 部分代码修改会影响到被测模块的原有外部接口或内在语义,并导致部分测试用例失效.

(3) 若代码修改生成新的测试需求,则需额外设计新的测试用例.

针对上述问题,学术界和工业界提出了一系列测试用例维护技术.典型技术包括失效测试用例的识别和修复、测试用例选择、测试用例优先排序(test case prioritization,简称 TCP)、测试用例集缩减和测试用例集扩充(test suite augmentation)等[4,5].

TCP 问题是目前回归测试研究中的一个热点,针对该问题的研究工作最早可以追溯到 Wong 等人在 1997 年发表的论文[6].TCP 技术通过优化测试用例的执行次序来提高回归测试效率.具体来说,在特定排序准则指导下,将测试用例按重要程度从高到低进行排序并依次执行.实践表明,TCP 技术可以协助测试人员提高缺陷检测速率并使开发人员尽早展开调试工作.目前,TCP 技术中采用的排序准则一般建立在对源代码、需求或模型的分析基础上;同时,TCP 技术不仅成功应用到传统软件的测试中,而且也逐渐成功应用到特定领域的测试中,包括图形用户界面测试、Web 应用测试、Web 服务测试、组合测试和缺陷定位等.为了对该研究问题进行系统分析、总结和比较,我们通过论文数据库的检索,最终选择出与该主题直接相关的论文共 77 篇(截止到 2013 年 1 月).从论文发表的时间来看,该问题一直是回归测试中的一个研究热点;从论文发表的会议和期刊来看,绝大部分论文均发表在软件工程领域的权威期刊和会议上.

本文第 1 节对 TCP 问题进行描述.第 2 节依次从源代码、需求和模型这 3 个角度对已有的 TCP 技术进行分类.第 3 节对资源感知的 TCP 问题的已有研究成果进行总结.第 4 节依次分析实证研究中经常采用的评测数据集、评测指标和缺陷植入类型对实证研究结论的影响.第 5 节依次总结 TCP 技术在不同特定测试领域中的应用.最后对 TCP 问题的未来可能研究工作进行了展望.

1 测试用例优先排序问题描述

对 TCP 问题的研究最早可以追溯到 Wong 等人在 1997 年发表的研究工作[6].他们结合测试用例历史覆盖信息和代码修改信息,针对特定程序版本提出一种混合方法,并开发出 ATAC 工具.该方法首先借助测试用例集缩减技术识别出冗余测试用例,对余下的非冗余测试用例,根据其覆盖能力进行排序.但未对冗余测试用例给出明确的排序方法.

随后,Elbaum 和 Rothermel 等人在 2000 年对 TCP 问题给出如下一般性描述[7]:

- 给定:测试用例集 T,T 的全排列集 PT,排序目标函数 f,其定义域为 PT,值域为实数.
- 问题:寻找 $T'\in PT$,使得$(\forall T'')(T''\in PT)(T''\neq T')[f(T')\geqslant f(T'')]$.

根据上述描述可知,集合 PT 包含了 T 中所含测试用例的所有可能执行次序,函数 f 的输入是一个特定执行次序,输出值与排序目标相关,假设取值越大,该执行次序的排序效果越好.一种典型排序目标是最大化测试用例集的早期缺陷检测速率,实践中还可以考虑其他排序目标,例如代码覆盖率、高风险缺陷的检测率、代码修

© 中国科学院软件研究所 http://www.jos.org.cn

改相关缺陷的检测率和系统可靠性等[8].

随后,研究人员又将 TCP 问题进一步细分为两类:非特定版本(general)TCP 问题和特定版本(specific)TCP 问题.前者在测试用例排序时不需要执行代码修改分析,并希望排序结果可以在随后的一系列程序版本上均有效;而后者则需执行代码修改分析,并希望排序结果在特定程序版本上有效.

2 测试用例优先排序技术分类

本文依次从代码、模型和需求这 3 个角度对已有 TCP 技术进行分类.

2.1 基于代码的TCP技术

2.1.1 贪婪法

贪婪法是目前解决 TCP 问题的一类常用方法.在搜集完测试用例的历史覆盖信息后,依据每个测试用例的程序实体(例如语句、分支或函数)覆盖能力,为其设置权重,并应用贪婪法来指导测试用例的排序,其假设是提高程序实体的早期覆盖率有助于提高缺陷的早期检测速率.根据是否引入反馈机制,可以将贪婪法细分为两种策略:Total 策略和 Additional 策略[3,8].其中,Total 策略直接计算出各个测试用例的程序实体覆盖率,并从高到低进行排序.假设测试用例有 m 个,需要覆盖的程序实体有 n 个,则该策略的时间复杂度为 $O(mn)$.而 Additional 策略则引入反馈机制.该策略从整体角度出发,认为当一部分程序实体被已执行测试用例覆盖后,对剩余测试用例进行排序时则并不需要考虑对上述程序实体的覆盖.具体来说,即每次执行完一个测试用例时,需要对余下测试用例的覆盖信息进行实时更新.当所有程序实体均覆盖后,将这些程序实体重新设置为未覆盖并对剩余测试用例迭代应用上述过程.该策略的时间复杂度为 $O(m^2n)$.

Rothermel 等人在研究 TCP 问题时,首先关注的是测试用例的语句和分支覆盖情况[3,8].这类程序实体可归为细粒度程序实体.他们提出了一系列排序策略,其中,stmt-total 和 branch-total 属于 Total 策略,stmt-addtl 和 branch-addtl 属于 Additional 策略.他们在实证研究中将这些策略与保持原有次序不变策略、随机排序策略和最优排序策略进行了比较,并验证了 TCP 技术的有效性.随后,Elbaum 等人进一步分析了程序实体的粒度大小对排序效果的影响[7,9].他们关注的是测试用例的函数覆盖情况.这类程序实体可归为粗粒度程序实体.实证研究结果验证了他们的推测,即基于粗粒度程序实体的排序方法虽然程序分析的开销较小,但也会降低排序效果.Jones 和 Harrold 考虑的程序实体是 MC/DC(modified condition/decision coverage)[10].MC/DC 覆盖准则在测试充分性上要强于分支和条件覆盖准则,但他们在实证研究中仅考察了排序所需时间,而未考察其缺陷检测速率.上述研究工作均采用基于 C 编程语言的评测程序进行有效性评估,Do 等人则尝试将 TCP 技术应用于 Java 编程语言实现的评测程序和基于 JUnit 框架的测试用例.他们提出了一系列基于语句块和方法覆盖的 Total 策略和 Additional 策略[11,12].实证研究结果表明,TCP 技术在面向对象程序的回归测试中同样有效.上述 TCP 技术均需通过插桩被测程序和执行测试用例来搜集程序实体的覆盖信息.梅宏等人对 JUnit 测试用例和被测程序进行静态分析来构建出静态调用图,随后,通过估计每个测试用例的代码覆盖能力来完成测试用例的排序[13,14].

但 Rothermel 等人也承认,上述两类策略与最优解之间仍存在一定的差距.李征等人在 Additional 策略基础上提出另一种贪婪法,即 2-Optimal 策略[15].该策略的不同之处在于,每一次需选择出两个测试用例,其时间复杂度为 $O(m^3n)$.在执行 Total 策略或 Additional 策略中,有时会出现多个候选测试用例拥有相同的覆盖能力.传统方法是从中随机选择一个,而姜博等人则从另一个角度出发,采用自适应随机测试(adaptive random testing)技术进行选择并提出了多种排序方法[16].

也有研究人员借助软件度量技术来进一步提高 TCP 技术的有效性.Rothermel 等人根据测试用例的 FEP (fault exposing potential)值来指导测试用例的排序[3,8].FEP 值可以用于估计测试用例的缺陷检测能力,其计算基于 PIE 模型[17]和变异测试分析.PIE 模型认为可检测程序内在缺陷的测试用例,需满足 3 个条件:(1) 测试用例执行了包含缺陷的语句;(2) 造成程序内部状态出错;(3) 通过传播错误的内部状态影响到程序的输出.变异测试是评估测试用例集充分性的一种有效手段.对被测程序执行符合语法约束的简单代码修改(即变异算子)可生成大量变异体,当测试用例在变异体和原有程序上的执行行为不一致时,则称该测试用例可以检测到该变异体.以

© 中国科学院软件研究所 http://www.jos.org.cn

语句覆盖为例,给定被测程序 P 和测试用例 t,首先计算语句 $s(s\in P)$的 ms 值,计算公式为

$$ms(s,t)=killed(s,t)/mutants(s).$$

其中,$mutants(s)$返回一组变异算子应用到语句 s 上得到的变异体数,$killed(s,t)$返回测试用例 t 在上述生成的变异体中可以检测出的数量.假设测试用例 t 覆盖的语句集为 S,则该测试用例的 FEP 值为 $\sum_{s\in S} ms(s,t)$.

基于 FEP 值,Rothermel 等人分别应用 Total 策略和 Additional 策略完成测试用例的排序.Elbaum 等人则从函数覆盖角度出发,对测试用例的 FEP 值进行计算[7,9].除此之外,Elbaum 等人还将 FEP 值与其他缺陷倾向值(fault proneness)(例如 fault index 值和 Diff 值)结合使用,提出一系列排序方法.以 Diff 值、FEP 值和 Total 策略为例,首先根据 Diff 值对测试用例进行排序,然后对拥有相同 Diff 值的测试用例,按照 FEP 值进行排序[7,9].

Jeffrey 和 Gupta 借助程序切片来解决 TCP 问题[18,19].他们基于测试用例输出的相关切片(relevant slice)提出一种新的 TCP 技术,应用相关切片可以识别出所有可能影响程序输出结果的语句或分支,并在排序时仅考虑对这些语句或分支的覆盖情况.

Elbaum 等人认为,在测试用例排序时仅考虑对程序实体的覆盖并不充分,他们在排序时进一步考虑了测试用例的执行开销和缺陷的危害程度[20].在实证研究中,他们考虑了不同的测试用例执行开销和缺陷危害程度分布,其中,测试用例执行开销的分布包括均匀分布、随机分布、正态分布、取自 Mozilla 开源项目中的分布和取自 QTP 应用中的分布.而缺陷危害程序的分布包括均匀分布和取自 Mozilla 开源项目的相关分布.赵建军等人通过分析被测程序内部结构,可以估计出每个模块的缺陷倾向性和重要性,并用于指导测试用例的排序[21].

虽然研究人员提出了大量 TCP 技术,但根据测试场景特征从中选出合适的 TCP 技术同样值得关注.Elbaum 等人通过实证研究考察了不同测试场景(主要考察因素包括被测程序特征、测试用例特征和代码修改类型等)对 TCP 技术有效性的影响.实证结果为测试人员在不同测试场景下选择出合适的 TCP 技术提供了重要依据[22].Arafeen 等人认为,在软件持续演化过程中,每次演化时的代码修改类型对 TCP 技术的选择存在影响.他们将该问题建模为多准则决策(multiple criteria decision making)问题,并采用层次分析法(analytical hierarchy process, 简称 AHP)进行选择.结果表明,他们提出的方法可以为每次回归测试活动选择出最为经济有效的 TCP 技术[23].

2.1.2 机器学习法

李征等人通过将 TCP 问题多项式时间规约为背包问题,证实该问题是一个 NP 难问题[15].所以,除了上述贪婪法,也有研究人员尝试采用典型的机器学习方法来提高 TCP 技术的执行效果.

李征等人将元启发式搜索(meta-heuristic search)技术应用到 TCP 问题上[15].元启发式搜索技术提供了一个高层框架,基于特定的启发式技术,可以采用合理的计算开销去求解组合优化问题.李征等人考虑了两种元启发式搜索技术:爬山法和遗传算法.其中,爬山法属于局部搜索算法,容易陷入局部最优;而遗传算法属于全局搜索算法,受达尔文的“适者生存和优胜劣汰”思想的启发来指导搜索最优解.结果表明,Additional 策略、2-Optimal 策略和遗传算法都要优于其他方法.陈振宇等人同样采用爬山法和遗传算法[24].他们采用模拟实验方式将这两种策略与 Total 策略、Additonal 策略以及 2-Optimal 策略进行了比较,结果表明,绝大部分情况下,Additonal 策略和 2-Optimal 策略的有效性要优于遗传算法、爬山法和 Total 策略.

Mirarab 和 Tahvildari 从概率观点出发,将贝叶斯网络应用到 TCP 问题上[25].他们在构建贝叶斯网络模型时需要如下信息:程序实体、每个程序实体的缺陷倾向和每个测试用例检测到缺陷的概率.随后,他们通过引入反馈机制对上述方法进行了扩充[26],即假设若选择出的测试用例覆盖了一部分程序实体,则将降低覆盖同类程序实体的测试用例的选择概率.该方法与 Rothermel 等人提出的 Additional 策略类似.

Leon 和 Podguski 引入一类基于执行轨迹分布的测试用例筛选和排序方法[27].首先设置相异性函数来评估两个测试用例执行轨迹间的相异程度,然后借助相异性函数完成测试用例的聚类分析.通过聚类分析可获得如下辅助信息:(1) 同一聚类内的测试用例可能是冗余测试用例;(2) 孤立点包含的测试用例有可能会触发失效;(3) 空间内低密度区域包含的测试用例可能覆盖了罕见行为模式.第 1 点可用于指导冗余测试用例的筛选,而满足后两点的测试用例则更有可能检测到被测软件的内在缺陷.所以,给这类测试用例设置更高的优先级有助于提高缺陷检测速率.随后,他们进一步提出 Cluster Filtering 和 Failure-pursuit 采样策略来指导测试用例的筛选和

© 中国科学院软件研究所 http://www.jos.org.cn

排序.实证研究结果表明,基于分布的方法在缺陷检测能力上要优于基于覆盖的方法,但这两类方法在不同类型缺陷上的检测能力并不相同,且可以形成互补.所以,他们综合这两类方法提出了一种有效的测试用例混合排序方法.Carlson 等人同样将具有相似性质的测试用例划分到同一聚类里,并推断同一聚类内的测试用例具有相似的缺陷检测能力.但与 Leon 和 Podguski 的研究工作不同,他们在完成聚类分析后,在随后的测试用例排序时还综合考虑了代码覆盖信息、代码复杂度以及测试用例的历史缺陷检测信息等[28].

与 Elbaum 等人的研究[20]相似,Huang 等人认为,在实际软件测试过程中,测试用例的执行开销和缺陷危害程度并不一致.他们通过搜集测试用例的历史执行信息来估计测试用例的执行开销和缺陷危害程度,并提出了一种基于遗传算法的 TCP 技术[29].

2.1.3 融合专家知识法

研究人员认为,在已有排序方法的基础上,若考虑融合专家知识,则有助于进一步提高测试用例的排序效果.

Tonella 等人提出一种基于实例的排序(case-based ranking,简称 CBR)方法[30].该方法借鉴了机器学习中的提升(boosting)策略.提升策略提供了一个通用框架,可以将多个简单学习器合并为一个高效学习器.CBR 方法可以同时考虑多种不同类型的排序准则,并可以进一步融合专家知识.其中,通过指定部分测试用例间的两两优先级关系来构造专家知识库.CBR 方法采用迭代方式进行学习,且每次迭代完成后,均会返回一个临时排序结果,其排序效果随迭代次数的增加而逐渐提高.Yoo 等人则将专家知识融合到基于聚类分析的 TCP 技术中[31].该方法可以有效减少专家需要指定的测试用例间的两两优先级关系.具体来说,在完成聚类分析后,簇间的排序依赖专家知识,而簇内测试用例的排序则基于代码覆盖率.在完成这两层排序后,他们提出一种交织聚类排序(interleaved clusters prioritisation)技术,即,首先从优先级最高的簇类内选择一个优先级最高的测试用例,然后根据簇间优先级选择下一个簇类,并从中选择一个优先级最高的测试用例,依此类推,直到所有测试用例都被选择出来.在上述方法中,他们采用层次分析法完成专家知识的构建,层次分析法最早由运筹研究界提出并已经成功应用于需求工程研究中.实证结果表明,他们所提出的方法的有效性要高于传统的基于覆盖的 TCP 技术.同时,该方法具有一定的鲁棒性,即,在存在专家误判的情况下,该方法仍然可以取得很高的缺陷检测速率.他们认为,上述现象的出现是因为这类方法的有效性受聚类分析结果的影响较大.该结论与 Leon 和 Podgurski 的研究工作[27]保持一致.而且,这类方法有时候可以让具有缺陷检测能力的测试用例优先执行,即使这类测试用例可能因语句覆盖率较低而导致优先级较低.

2.1.4 其他方法

除了上述方法,也有研究人员从其他角度对 TCP 问题进行研究.Srivastava 和 Thiagarajan 开发出了 Echelon 工具[32].该工具可以对修改前后程序的二进制代码进行比较并识别出受代码修改影响的基本语句块,随后根据测试用例对这些基本语句块的覆盖能力进行排序.Sherriff 等人借助矩阵分析中的奇异值分解来获取软件制品中的关联聚类(association cluster),并提出一种 TCP 方法[33].该 TCP 方法依赖关联聚类、测试用例和文件间的关联关系以及修改向量.具体来说,在修改矩阵上应用奇异值分解可以生成关联聚类,即,若两个文件在缺陷修复时经常被同时修改,则将这两个文件聚类到同一关联聚类内.每个文件与影响或覆盖该文件的测试用例建立关联关系.修改向量被用于描述代码修改信息,其中,相关分量反映了对应文件是否被修改过.使用关联聚类和修改向量可以为每个文件设置优先级,以衡量新的代码修改与每个测试用例的匹配程度.该方法的一个新颖之处在于,可以使用任意软件制品来指导测试用例的排序.Ramanathan 等人提出了 PHALANX 框架.他们搜集测试用例的历史覆盖信息,并将其建模为测试用例相异图(test-case dissimilarity graph),其中,节点代表测试用例,边上的权重代表测试用例间的相似程度,基于该模型提出了多种 TCP 技术[34].

2.1.5 基于源代码的经典 TCP 研究工作比较

本节对基于源代码的经典 TCP 研究工作进行系统的比较,比较因素包括方法类型、排序准则、排序策略以及实现评测程序的编程语言,最终结果见表 1.

© 中国科学院软件研究所 http://www.jos.org.cn

Table 1 Comparison for typical TCP research work based on source code

表 1 基于源代码的经典 TCP 研究工作比较

References	Type	Prioritization criterion	Prioritization strategy	Programming language
Rothermel, *et al*.[3,8]	Greedy	Statement/decision coverage; FEP based on statement	Total/additional	C
Elbaum, *et al*.[7,9]	Greedy	Statement/function coverage; FEP based on statement/function; FI-FEP based on function; Diff-FEP based on function	Total/additional	C
Jones, *et al*.[10]	Greedy	MC/DC coverage	Additional	C
Do, *et al*.[11,12]	Greedy	Block/method coverage; Diff based on method	Total/additional	Java
Mei, *et al*.[13,14]	Greedy	Method/statement coverage; Static call graphs of JUnit test cases and the program under test	Total/additional	Java
Li, *et al*.[15]	Greedy	Block/decision/ statement coverage	Total/additional/2-Optimal	C
	Meta-Heuristic search		Hill climbing/genetic algorithm	
Jiang, *et al*.[16]	Greedy	Statement/decision/ function coverage	Total/additional	C
			Adaptive random testing	
Jeffrey, *et al*.[18,19]	Greedy	Statement/decision coverage	Relevant slices	C
Li, *et al*.[24]	Greedy	Requirement coverage	Total/additional/2-optimal	Simulation
	Meta-Heuristic search		Hill climbing/genetic algorithm	
Mirarab, *et al*.[25,26]	Machine learning	Class/method coverage	Bayesian network	Java
Carlson, *et al*.[28]	Machine learning	Code coverage, code complexity metric, and fault history information	Cluster analysis	C++
Tonella, *et al*.[30]	Machine learning	Statement coverage, loop complexity, and expert knowledge	Case based ranking	C
Yoo, *et al*.[31]	Machine learning	Statement coverage and expert knowledge	Interleaved cluster analysis and analytical hierarchy process	C

2.2 基于模型的TCP技术

目前,实践人员和研究人员逐渐倾向于借助模型来辅助软件系统的设计、开发和维护.Korel 等人重点关注了一类基于状态的软件系统,典型系统包括计算机通信系统和工业控制系统等.他们将这类系统用扩展有限状态自动机(EFSM)进行描述,EFSM 模型由状态(state)和迁移(transition)构成.首先,通过对比修改前后模型可以识别出模型差异.模型差异通过基本模型修改操作(包括迁移添加、迁移移除等)序列来表示;随后,在修改后的模型上执行所有测试用例,以搜集测试用例对模型差异的覆盖信息;最后,基于上述覆盖信息提出两种排序方法:选择性排序法和基于模型依赖关系的排序法.前一种方法将测试用例分为两类:高优先级测试用例 TS_H 和低优先级测试用例 TS_L,将覆盖了模型差异的测试用例放置到集合 TS_H,否则放置到集合 TS_L.在执行时,先执行 TS_H 中的测试用例,后执行 TS_L 中的测试用例.对同一集合内的测试用例,通过随机排序方式决定先后执行次序.后一种方法对 TS_H 中的测试用例,通过分析模型差异与模型其他模块间的控制和数据依赖关系来确定测试用例的执行次序[35].在上述工作的基础上,他们进一步提出一些启发式设定来辅助测试用例的排序.这些启发式包括为覆盖更多迁移修改操作的测试用例设置更高的优先级,为覆盖迁移修改操作频次更高的测试用例设置更高的优先级等[36,37].在面向对象软件系统的系统测试阶段,Kundu 等人采用 UML 中的序列图(sequence diagrams),基于场景覆盖准则来指导测试用例的设计.随后,基于序列图中的消息权重和边权重,提出了 3 种不同的测试用例排序准则:消息权重总和、加权平均路径长度和代码权重[38].

2.3 基于需求的TCP技术

也有研究人员从需求角度出发来指导测试用例的排序.Srikanth 等人在系统测试阶段提出了 PORT 排序方法[39].该方法在对测试用例排序时考虑如下的影响因素:需求变更的可能性、客户定义的需求优先级、需求的实现复杂度和需求的缺陷倾向性.该方法的不足在于,一些影响因素的取值需要人工估计,造成取值具有主观性并会影响随后的排序效果.屈波等人基于测试用例的设计信息,提出了一组基于需求的测试用例优先级动态调整算法.该方法不仅可以充分利用测试用例的设计信息(论文中主要考虑的是测试用例对测试需求的预期覆盖

© 中国科学院软件研究所 http://www.jos.org.cn

情况),而且还可以通过搜集测试用例的执行信息对测试用例优先级进行动态调整[40,41].已有研究假设测试需求优先级和测试用例执行开销保持一致,但章晓芳等人认为,该假设在实际软件测试中很难成立.因此,她们分别提出了基于 Total 策略和 Additional 策略,考虑测试需求优先级和测试用例执行开销的 TCP 技术[42].与 Srikanth 等人的研究工作[39]相似,Krishnamoorthi 和 Sahaaya 基于软件需求规约,同样提出了一种适用于系统测试阶段的 TCP 技术.他们在对测试用例进行排序时考虑了更多的影响因素包括客户定义的需求优先级、需求变动信息、需求实现复杂度、需求完整性、需求可追踪性和缺陷影响程度等[43].Yu 和 Lau 则在规约测试时提出并实现了一种基于缺陷的测试用例排序技术.该技术在排序时考虑了不同缺陷类型间的层次结构;同时,他们提出了 PATE 评测指标进行有效性评估,并将该方法用于检测与逻辑表达式相关的程序缺陷[44].

3 测试资源感知的测试用例排序技术

与测试用例集缩减和测试用例选择等技术不同,理想情况下,TCP 技术假设测试人员可以按照测试用例执行次序依次执行完所有测试用例.但在实际测试过程中,受资源约束,不能保证执行完所有测试用例.例如,一些项目在开发过程中需要白天编程实现的功能模块在晚上完成所有链接、编译和单元测试等工作.本节将这类问题统称为测试资源感知(cost-aware)的测试用例排序问题.

Kim 和 Porter 首先对该问题展开研究[45].他们将回归测试视为一组具有偏序关系的测试行为序列,其中每一个阶段的测试行为受到上一个阶段测试行为的影响,同时需要考虑测试资源和时间的约束.他们提出两种程序演化模型,其中,模型 1 从基程序出发,每次植入单一缺陷并形成新的程序版本,直到所有缺陷均被植入;模型 2 从一个包含所有缺陷的程序出发,每次移除其中一个缺陷并形成新的程序版本,直到所有缺陷均移除.他们提出的 TCP 方法包含 4 个步骤:(1) 应用测试用例选择技术,选择出部分测试用例并构成集合 T';(2) 为 T'中的每个测试用例设置选择概率;(3) 根据选择概率,从 T'中选出一个测试用例并执行;(4) 重复步骤(3),直至测试资源和时间用完.该方法的核心是选择概率值的设定.受统计质量控制和统计预测理论的启发,他们将测试用例 tc 在第 t 个测试阶段的选择概率定义为 P_t,其计算公式为

$$\begin{cases} P_0 = h_1 \\ P_k = \alpha \times h_k + (1-\alpha) \times P_{k-1} \end{cases} \tag{1}$$

其中,h_k 是测试用例 tc 在第 k 个阶段的观察值,其取值可以基于测试用例的在该阶段的执行情况、缺陷检测情况或程序实体的覆盖信息;$\alpha(0\leqslant\alpha\leqslant1)$是平滑常量,用于平滑不同测试阶段观察值的权重.以缺陷检测情况为例,若测试用例 tc 在第 k 个阶段未检测到缺陷,则 h_k 的取值设置为 0;否则,h_k 的取值设置为 1.

Walcott 等人认为,测试用例的执行时间是回归测试中的一个重要因素,并提出一种时间感知(time-aware)的测试用例优先级排序技术.这类技术不需要对整个测试用例集进行排序,而是从中选择一个子集进行排序,并确保其执行时间满足一个给定时间的约束[46,47].他们将该问题形式化描述如下:

给定:测试用例集 T,T 的所有子集构成的全排列集 PT,函数 f 和 $time$,两者定义域为 PT,值域为实数,时间约束为 $t_{\max}$.

问题:寻找 $T'\in PT$ 且 $time(T')\leqslant t_{\max}$,使得$(\forall T'')(T''\in PT)(T''\neq T')[time(T'')\leqslant t_{\max}][f(T')\geqslant f(T'')]$.

Walcott 和 Alspaugh 等人将这类问题规约为 0/1 背包问题,从而证实该问题是 NP-Complete 问题[46,47]. Walcott 等人首先采用遗传算法对该问题求解,并将该方法与保持原有次序不变、逆向排序、随机排序和缺陷感知的排序策略进行了比较[46].Alspaugh 等人应用传统背包求解器对该问题进行求解,并比较了不同求解器的求解效果[47].张路等人将该问题求解分为两个阶段:首先,应用整数线性规划从已有测试用例集 T 中选择出子集 T';然后,应用 Total 策略或 Additional 策略对 T'中的测试用例进行排序.他们将该类方法与基于遗传算法的方法和传统 TCP 方法进行比较发现:他们所提出的方法在测试时间较短时具有一定的优势;当测试时间较长时,传统 TCP 方法也具有一定的优势[48].陈振宇等人采用西门子套件和 space 程序作为他们的实证研究对象[49],发现不同方法在实证研究中的差异性并不显著.

上述方法均假设在测试用例排序时已经预先知道了时间约束,而 Do 等人则深入探讨了时间约束对 TCP 技

© 中国科学院软件研究所 http://www.jos.org.cn

术的成本和收益的影响[50,51].在实证研究中,他们考虑了 6 种不同的 TCP 技术:保持原有次序不变、随机排序、基于语句块覆盖的 Total 策略和 Additional 策略、未引入反馈机制和引入反馈机制的贝叶斯网络法;采用了 4 种时间约束,分别是 25%,50%,75%和 100%;并通过成本收益模型对 TCP 技术的成本和收益进行度量.结果表明:在给定特定 TCP 技术时,不同时间约束对 TCP 技术的成本和收益存在影响;在给定特定时间约束时,时间约束对不同 TCP 技术的影响并不一致,即,在某一时间约束下最为有效的 TCP 技术,在其他时间约束下并不一定是最有效的.

4 实证研究

通过分析已有论文可以看出,大部分研究工作均采用实证研究方式对 TCP 技术的有效性进行评估.

该节首先总结常用的评测指标,随后对常用的评测程序进行总结,最后探讨缺陷植入类型对实证研究结论的影响.

4.1 评测指标

软件测试的目的是设计出高质量的测试用例来尽可能多地检测出被测软件内部存在的缺陷.在设计 TCP 技术评测指标时,需要体现出测试用例的缺陷检测速率.Rothermel 等人首先提出了 APFD(average percentage of fault detection)评测指标.当给定测试用例的执行次序时,该评测指标可以给出测试用例执行过程中检测到缺陷的平均累计比例.其取值范围介于 0~100%之间,取值越高,则缺陷检测速度越快.Elbaum 等人随后给出了 APFD 的计算公式[7],假设有测试用例集 T,其中包含 n 个测试用例,m 个缺陷,给定一个测试用例执行次序,其中,TF_i 表示首个可检测到第 i 个缺陷的测试用例在该执行次序中所处次序,则 APFD 的计算公式为

$$APFD = 1 - \frac{TF_1 + TF_2 + \ldots + TF_m}{nm} + \frac{1}{2n} \tag{2}$$

考虑如表 2 所示的实例,该实例有 5 个测试用例和 10 个缺陷,假设测试用例的缺陷检测信息已知,例如测试用例 B 可以检测出编号为 1,5,6 和 7 的缺陷.在该实例中,测试用例排序的目的是最大化缺陷的早期检测能力.

Table 2 An instance from Ref.[7]

表 2 来自文献[7]中的一个实例

	1	2	3	4	5	6	7	8	9	10
A	×				×					
B	×				×	×	×			
C	×	×	×	×	×	×	×			
D					×					
E								×	×	×

若测试用例的执行次序是 *A-B-C-D-E*,则 APFD 的取值是 50%;若测试用例的执行次序是 *E-D-C-B-A*,则 APFD 的取值是 64%.所以,在该实例中,执行次序 2 的效果要优于执行次序 1.图 1 表示不同执行次序下,测试用例执行比例与检测出的缺陷比例之间的关系.以图 1(a)为例,测试用例 *A* 执行结束后(即执行了 20%的测试用例),可以检测出 20%的缺陷,对应图中点(0.2,20).将所有点连成折线,则 APFD 取值对应的是折线下方阴影占整个面积的百分比.

但测试人员无法事先预知到测试用例的缺陷检测信息,李征等人随之提出评测指标 APBC,APDC 和 APSC[15].这些指标可以衡量测试用例的执行次序对原有程序不同程序实体(分别是语句块、分支和语句)的覆盖速率.这些指标的计算公式与 APFD 相似.以 APBC 为例,其计算公式为

$$APBC = 1 - \frac{TB_1 + TB_2 + \ldots + TB_m}{nm} + \frac{1}{2n} \tag{3}$$

其中,TB_i 表示首个可覆盖到第 i 个语句块的测试用例在该执行次序中所处的次序,其他变量含义与公式(2)保持一致.

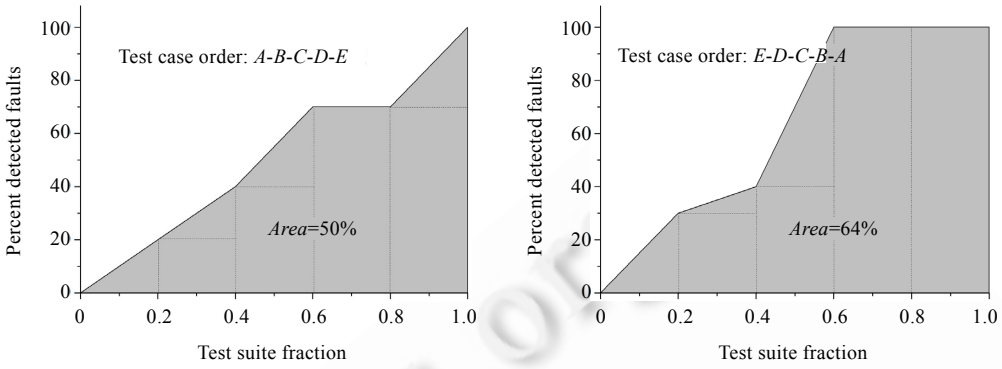
除此之外,Elbaum 等人还提出一种新的评测指标 $APFD_C$[20],该指标综合考虑了测试用例的执行开销和缺陷

© 中国科学院软件研究所 http://www.jos.org.cn

危害程度,其计算公式为

$$APFD_C = \frac{\sum_{i=1}^{m}\left(f_i \times \left(\sum_{j=TF_i}^{n} t_j - \frac{1}{2} \times t_{TF_i}\right)\right)}{\sum_{i=1}^{n} t_i \times \sum_{i=1}^{m} f_i} \tag{4}$$

其中,t_i 代表第 i 个测试用例的执行开销,f_i 代表第 i 个缺陷的危害程度,其他变量含义与公式(2)保持一致.当执行开销和缺陷危害程度保持一致时,通过化简公式(4)可得到公式(2).



(a) 测试用例次序 *A-B-C-D-E* 对应的 APFD 值 (b) 测试用例次序 *E-D-C-B-A* 对应的 APFD 值

Fig.1 APFD value for different orders

图 1 不同次序对应的 APFD 值

APFD 及其他相关评测指标的理论取值上限是 100%,即至少存在 1 个测试用例可以检测出所有可能缺陷,并且该测试用例被最先执行.但在实际情况中,很少存在满足上述条件的测试用例,所以这些评测指标主要用于比较不同排序策略之间的优劣性.

4.2 评测程序

本节将实证研究中采用的来自实验室或开源软件界的评测程序进行了搜集和汇总,最终结果见表 3.表 3 依次罗列了程序名称、编程语言、程序规模、程序功能描述、首次使用时间和累计使用次数,其中,将代码行数超过 10 000 行的程序设定为大规模程序,将代码行数介于 1 000 行~10 000 行之间的程序设定为中规模程序,而将代码行数小于 1 000 行的程序设定为小规模程序.例如通过表 3,可以获知 Ant 程序由 Java 编程语言实现,属于大规模程序,该程序是 Java 语言的项目构建工具.研究人员在 2004 年首次使用该程序作为评测程序,截至目前为止,累积使用次数为 11 次.表 3 中罗列的大部分评测程序都可以从内布拉斯加大学林肯分校(University of Nebraska-Lincoln)的 SIR 库中去下载[52].在表 3 中,我们将评测程序按照累计使用次数从高到低进行排序.从排序结果可以看出:最常用的面向结构程序是以 C 语言编程实现的西门子套件(Siemens suite)和 space 程序,最常用的面向对象程序是以 Java 语言编程实现的 Ant,XML-security,Jmeter 和 Jtops.接下来对这些重要的评测程序的特征依次加以介绍.

西门子套件包括 7 个小规模 C 语言程序,其中,tcas 是防止航空器空中相撞系统,schedule 和 schedule2 是优先级调度器,tot_info 针对指定的输入数据生成统计信息,print_tokens 和 print_tokens2 是词法分析器,replace 程序完成模式匹配和替换.这些程序早期被西门子研究院用于研究控制流和数据流覆盖准则的缺陷检测能力[53].西门子套件中的每个程序均配有大量测试用例,测试用例的设计过程包括两个阶段:首先,基于黑盒测试中的等价类划分技术;然后采用白盒测试技术,确保每个可执行的语句、分支和定义使用对至少被 30 个测试用例覆盖到.研究人员也为每个基准程序创建了大量缺陷版本.这些缺陷版本一般通过修改一行代码生成,少量通过修改 2~5 行代码生成.注入的缺陷来源于实际的编程经历.在生成的大量缺陷版本中,被保留下的缺陷版本至少被 3

© 中国科学院软件研究所 http://www.jos.org.cn

个、最多被 350 个测试用例覆盖到.

Table 3 Evaluation programs used in previous empirical studies

表 3 已有实证研究中使用的评测程序

Name	Programming language	Size	Description	First used	Cumulative number
Siemens suite	C	Small	Small programs developed by Siemens	1999	13
Ant	Java	Large	A Java-based build tool	2004	11
Space	C	Medium	An interpreter for an array definition language	1997	10
XML-security	Java	Large	Implements security standards for XML	2004	10
Jmeter	Java	Large	A Java desktop application used to load-test functional behavior and measure performance	2004	9
Jtopas	Java	Medium	Facilitates users to tokenize and parse arbitrary text data	2004	8
Flex	C	Medium	A lexical analyzer generator	2002	5
Sed	C	Medium	Stream editor	2004	5
Galileo	Java	Large	A Java bytecode analyzer	2006	5
Nanoxml	Java	Medium	A small XML parser for Java	2006	5
Grep	C	Large	Searches input files for a pattern	2002	4
Gzip	C	Medium	Data compression/decompression	2004	4
Jdepend	Java	Large	Measures the design quality of Java software	2006	3
Bash	C	Large	Unix shell	2004	2
Gradebook	Java	Large	Provides functions to perform typical grade book tasks	2006	2
QTB	C	Large	An embedded real-time subsystem that performs initialization tasks on a level-5 RAID storage system	2002	1
GCC	C	Large	A compiler for C programming language	2003	1
Jikes	Java	Large	A compiler for Java programming language	2003	1
Javac	Java	Large	A compiler for Java programming language	2003	1
Emp-server	C	Large	Server module of internet game	2004	1
Make	C	Large	Build manager for C programming language	2004	1
Xearth	C	Large	View earth from space	2004	1
Vim	C	Large	Text editor	2009	1

Space 程序是为欧洲航天局开发的、针对数组定义语言(ADL)的一个解释器.该程序包含 35 个版本,每个仅包含单一缺陷,其中,30 个缺陷版本是在开发过程中发现的.测试用例集的构建过程也包括两个阶段:第 1 阶段随机生成 10 000 个测试用例;第 2 个阶段通过添加测试用例,使得程序对应控制流图中每条可执行边至少被其中 30 个测试用例覆盖到.最终生成的测试用例集包括 13 585 个测试用例.

Ant 是一个基于 Java 的程序构建工具,类似于 Unix 系统中的 make 工具.Jmeter 是一个 Java 桌面应用程序,用于执行性能测试.XML-security 实现了 XML 中的安全协议.JTopas 是一个用于解析文本数据的 Java 类库.这些程序的规模介于 1.8KLOC~80KLOC 之间.同时,在 SIR 库中,这些程序均配备了大量基于 Junit 测试框架的测试用例.根据粒度大小,测试用例可以分为两类:基于方法级别的测试用例和基于类级别的测试用例.表 4 对上述 4 个程序特征进行总结,并依次罗列了程序的版本数、程序包含的类的个数、基于类级别的测试用例数、基于方法级别的测试用例数和可用的缺陷版本总数[11].

Table 4 Characteristics of Java evaluation programs[11]

表 4 Java 评测程序特征[11]

Name	Versions	Size	Test classes	Test methods	Faults
Ant	9	627	150	877	21
Jmeter	4	143	14	83	6
XML-Security	6	389	28	78	9
JTopas	4	50	11	128	5

通过对实证研究中的评测程序使用趋势进行分析我们发现:

(1) 在 2001 年以前,研究人员主要采用西门子套件中的小程序和 space 程序作为评测程序,但这类程序因代码规模较小和植入缺陷类型较少,实证研究中所得结论不具有一般性,并逐渐受到学术界和工业界的质疑.

(2) 为了提高研究工作的说服力,从 2001 年开始,研究人员逐渐开始使用大规模程序作为评测程序.这些

© 中国科学院软件研究所 http://www.jos.org.cn

大规模程序大部分来自实验室或开源软件界,也有一部分是来自企业中的实际软件产品.例如,Carlson 等人使用了微软公司的内部产品 Dynamics AX 作为评测程序[28].

(3) 从实现评测程序的编程语言来看,早期以 C 编程语言为主,代表程序包括西门子套件和 space 程序,从 2004 年开始,逐渐以 Java 编程语言为主,代表程序包括 Ant,XML-security,Jmeter 和 Jtopas.其主要原因在于,这类程序的代码规模较大,且有很多实用的基于 Java 的代码分析工具来辅助实验设计.

4.3 缺陷植入类型的影响

已有研究在评估 TCP 技术有效性时一般采用手工植入的缺陷进行评估,假设这类缺陷与软件开发过程中产生的实际缺陷性质相似.但一些研究工作表明,变异测试分析中产生的变异缺陷与实际缺陷更为相似,而且手工植入的缺陷对实证结论的有效性存在影响[54].所以,Do 和 Rothermel 等人采用变异缺陷对 TCP 技术的有效性进行了评估.结果表明,TCP 技术在变异缺陷检测上同样有效,同时,测试用例集特征和缺陷特征对 TCP 技术的有效性存在一定的影响[55].

5 特定应用领域的 TCP 技术研究

目前,TCP 技术已经成功地从传统软件测试领域应用到不同特定应用领域的软件测试工作中,包括组合测试、事件驱动型应用测试、Web 服务测试和缺陷定位等.

5.1 组合测试

组合测试(combinatorial testing)是一种重要的测试用例设计技术,当完成对软件制品的建模和分析后,测试人员可以获得由因素和因素取值构成的实例,组合测试通过覆盖少数因素间的取值组合可以构造出小规模组合测试用例集.

Bryce 和 Colbourn 通过对组合设置权重,基于 one-test-at-a-time 贪心策略提出一种新的贪心法 DDA (deterministic density algorithm),最终在生成的排序后组合测试用例集中,可以使重要的组合被排在前面的测试用例优先覆盖到[56].随后,他们对上述方法进行扩充,可以有效支持组合测试中的约束和种子[57].王子元等人对组合测试用例集进行排序时,同时考虑了测试用例的执行开销和组合权重[58,59].

回归测试中的测试用例选择和排序技术的有效性与已有测试用例集的质量密切相关.Qu 等人通过实证研究验证了组合测试可以为回归测试提供高质量测试用例集,同时,他们也提出了多种优先级排序准则[60].同样,在测试易配置软件时,这类软件的所有可能配置一般存在组合爆炸问题.Qu 等人在这类软件的回归测试中,借助覆盖矩阵(covering array)从中选择出一些典型配置,同时也给出了多种针对这些配置的优先级排序准则[61].Srikanth 等人对 Qu 等人的研究工作[61]进行扩展,在对测试用例排序时不仅考虑提高缺陷检测率,而且还额外考虑了配置开销和环境搭建时间[62].

5.2 事件驱动型应用测试

事件驱动型应用(event-driven application)是目前广泛使用的一类软件.它们以事件序列作为输入,随后经历内部状态变更,最终输出事件序列.常见的事件驱动型应用包括图形用户界面(GUI)、Web 应用和网络协议等.

在为传统软件设计测试用例时,仅需考虑对语句或分支的充分覆盖;而对图形用户界面测试时,不仅需要考虑对 GUI 源代码的覆盖,而且还需要考虑对事件的覆盖.目前,研究人员一般将 GUI 应用建模为事件流图(event flow graph).但事件序列存在组合爆炸问题,Bryce 和 Memon 基于该模型,在测试用例排序时考虑了对事件间的 *t*-way 交互的覆盖[63].Huang 等人则依据事件的重要程度进行权重设置,并构建出权重事件流图(weighted-event flow graph),基于该模型提出了 3 种 GUI 测试用例排序方法[64].

Web 应用也是一种重要的事件驱动型应用.在基于用户对话的 Web 应用测试中,当测试预算不足时,很难执行完所有的测试用例.一种解决方法是根据测试用例的重要程度对测试用例依次排序并执行.Sampath 等人提出了 3 类排序策略:(1) 依据测试用例中包含的用户请求个数;(2) 依据用户请求序列出现频率;(3) 依据对用户请求中参数取值间交互的覆盖.实证研究结果表明,这 3 种排序策略均优于随机排序策略,而且后两种策略在缺

© 中国科学院软件研究所 http://www.jos.org.cn

陷检测速率上更具优势[65]. Garg 和 Datta 则从构建 Web 应用的后台数据库入手,有效识别出数据库相关的代码修改,并通过测试用例的排序来提高对这类特定缺陷的检测速率[66].

Bryce 和 Memon 等人针对事件驱动型应用,提出可以同时适用于 Web 应用和 GUI 应用的统一抽象模型,并基于该模型提出了多种通用排序准则[67].

5.3 Web服务测试

目前,软件产业中的领军企业如 IBM、微软和 Oracle 等,倡导采用服务导向的业务流程来构建企业级应用.这种基于 Web 服务的新兴软件构建方式给传统的 TCP 问题研究提出了新的挑战.

张路等人发现,Web 服务组件因无法获取源代码,难以提出相应的测试充分性准则来指导测试用例的选择和排序.而 Web 服务接口契约可以描述组件提供者和使用者的权利和义务,所以他们针对 Web 服务接口契约执行变异测试分析,并以变异体检测率为依据来指导测试用例的选择和排序[68].随后,张路等人在对由 Web 服务组建的软件系统测试时,为了解决特定时间段内 Web 服务请求数受限问题,将整个测试过程划分为一系列时隙,在每个时隙内,在满足指定 Web 服务请求数时,应用整数线性规划来指导测试用例的选择和排序[69].

梅立军等人通过分析 WS-BPEL 中的 Web 服务接口规约,依据 XML 消息中的标签覆盖信息提出一系列 TCP 技术[70,71].随后,梅立军等人提出了多层次覆盖模型,可以分别考虑工作流、Xpath 和 WSDL 等软件制品,并基于该模型提出一种分层 TCP 技术[72].

陈林等人针对 WS-BPEL 应用进行控制流和数据流依赖关系分析.他们首先构造出加权图,然后通过修改影响分析识别出代码修改影响模块,并基于上述分析完成测试用例的排序[73].Nguyen 等人在对 Web 服务组合进行审计测试时,提出一种基于信息检索的 TCP 方法[74].基于位置的 Web 服务(location-based Web service)是目前一种流行的应用.该应用可以使移动网络的运营商和终端用户同时获得收益.一方面,网络运营商通过提供更多基于位置的增值服务可以为基础设置带来收益;另一方面,终端用户通过基于位置的 Web 服务可以获得更好的用户体验和更高质量的服务.翟可等人针对这类应用提出了一系列评测指标和相应的 TCP 技术[75,76].

5.4 缺陷定位

与传统的软件调试技术不同,缺陷定位(fault localization)技术通过对源程序、测试结果以及各种程序行为特征信息的计算分析,给出缺陷在源代码中的可能位置.已有研究表明,TCP 技术可以有效提高缺陷检测速率,但在测试资源有限时,减少测试用例的执行数量会降低缺陷定位效果.姜博等人通过实证研究考察了 TCP 技术包含因素,如排序策略、覆盖粒度和时间开销等对缺陷定位效果的影响.他们考虑了 16 种 TCP 技术和 4 种缺陷定位技术.最终结果表明:

(1) 当测试资源有限时,TCP 技术是辅助缺陷定位的一种有效方法;

(2) 在完成测试用例排序后,若执行的测试用例数较少,则对缺陷定位效果的影响较大;

(3) 在已执行的测试用例中,若包含的未通过测试用例较多,则有助于提高缺陷定位效果;

(4) 在考虑的排序策略中,Random 策略效果最好,Additonal 策略和 ART 策略其次,Total 策略最差;

(5) 测试用例执行时搜集的程序实体粒度对缺陷定位效果的影响不显著[77−79].

Gonzalez-Sanchez 等人提出一种基于信息增益的 TCP 技术以提高缺陷定位效果.但该方法存在如下不足:(1) 对方法中的参数取值需要精确估算;(2) 算法复杂度呈指数级;(3) 采用在线方式完成测试用例的排序,即在选出一个测试用例时,需要分析已选择测试用例的执行结果[80].为了解决上述问题,他们提出了 RAPTOR 方法.该方法在测试用例排序时,考虑了对相似语句执行模式的约简[81].Kim 和 Baik 则通过引入缺陷定位技术,在测试用例排序时,同时考虑对程序实体的覆盖信息和测试用例的历史缺陷检测信息[82].

5.5 其他应用

变异测试(mutation testing)是评估和改进测试用例集充分性的一种有效手段.测试人员根据被测程序特征设计变异算子,通过应用变异算子到原有程序可以生成大量变异体,在识别出等价变异体后,若已有测试用例集不能检测出所有非等价变异体,则需额外设计新的测试用例来提高测试用例集的充分性.但变异测试因分析代

© 中国科学院软件研究所 http://www.jos.org.cn

价过高,难以适用于大规模复杂软件系统.Just 等人从变异体生成优化和变异体执行优化等角度出发提出了多种优化措施[83]:首先,通过移除 COR 类变异算子和 ROR 类变异算子中的冗余算子,可以在不影响变异分析效果的前提下减少生成的变异体数量;其次,通过分析测试用例的变异覆盖信息和执行时间差异,对测试用例进行排序以优化执行时间.实证研究结果表明:借助上述优化措施,可以将变异分析开销降低 65%.

测试用例 Oracle 模块提供了一种判断测试用例是否执行通过的验证机制.目前,已有研究工作集中关注的是测试用例 Oracle 模块的构造问题,而 Staats 等人则研究了测试用例 Oracle 模块对测试用例排序的影响[84].传统的测试用例排序方法假设提高程序实体的早期覆盖率有助于提高缺陷的早期检测速率,但研究者认为,要提高排序效果,仅考虑测试用例覆盖包含缺陷的语句并不充分,还需要测试用例能够造成程序内部状态出错,并通过传播影响到程序的实际输出.具体来说,他们通过执行测试用例,借助数据流分析来评估程序内部不同变量取值对 Oracle 模块相关变量取值的影响程度,这样可以计算出每个测试用例的与测试用例 Oracle 模块相关代码的覆盖率,最后借助上述信息来辅助测试用例的排序.基于 3 个同步反应系统的实证研究表明:与随机排序和基于结构覆盖排序的方法相比,他们所提方法在测试用例集的早期缺陷检测速率上具有一定优势.

6 未来研究展望

根据上述分析可以看出,TCP 问题已经得到学术界和工业界的广泛关注,并取得了大量研究成果.本文对 TCP 问题的背景、已有研究成果、评测数据集和评测指标等进行了系统的总结和分析.

从本文的介绍中可以看出,TCP 问题是回归测试中的一个重要研究热点,具有丰富的理论价值和应用前景.虽然已有研究取得了一定的成果,但在该领域还存在一些值得进一步研究的问题,主要包括:

(1) 新颖、高效的 TCP 技术研究.传统 TCP 问题是一个典型的 NP 难问题.虽然研究人员提出了大量新颖有效的 TCP 技术,但研究结果表明,这些技术与最优解之间仍存在一定的差距[15].理想情况下,测试用例在排序时应以测试用例在修改后程序上的缺陷检测能力为依据,但上述信息却无法预先获知.已有 TCP 技术一般借助一些替代排序准则,例如依据测试用例在原有程序上的程序实体覆盖能力进行排序.但已有实证研究结果表明,并没有一个排序准则在不同程序的测试中总是最为有效[3,9,15].一个值得关注的研究方向是基于多目标优化的 TCP 技术,即在测试用例排序时,同时考虑多个不同的排序准则,对这些准则采用加权求和或为不同准则设定优先级等方式进行拟合,并借助多目标优化研究领域的最新研究成果以提高求解质量.同时,在排序过程中进一步融合专家知识,也有助于提高最终排序效果.

(2) 与其他回归测试技术相结合.TCP 技术的有效性不仅与排序准则和排序策略有关,而且还与测试用例集的质量密切相关.在软件的开发和维护过程中,部分代码修改会增加、修改或移除原有的测试需求,导致已有测试用例集难以对代码修改进行充分测试,并进一步影响随后的测试用例排序效果.针对上述问题,可以将回归测试中的测试用例集扩充、测试用例集缩减和测试用例选择技术的最新研究成果引入到 TCP 研究中,通过添加新的测试用例、移除失效和冗余测试用例来提高测试用例集的质量,并用于随后的测试用例排序.

(3) 对影响TCP技术有效性的内在影响因素进行深入分析.在实证研究中进一步充分研究不同内在因素对 TCP 技术有效性的影响.通过分析我们认为,可以考虑的影响因素包括程序对象特征、测试用例集特征、植入缺陷的特征和代码修改特征等.例如,若针对测试用例集特征,则可以进一步考虑如下属性:测试用例集规模、测试用例输入的多样性和测试用例的覆盖率等.在选择评测程序进行分析时,除了可以考虑第 4.2 节中总结的来自实验室和开源软件界的典型程序,更需要与软件企业开展深入合作,将企业中的大规模复杂软件系统纳入到实证研究中,以提高研究工作的说服力.

(4) 寻找新的应用场景.通过第 5 节可以看出,TCP 技术已经初步成功应用于 Web 应用测试、Web 服务测试、组合测试等特定领域,并取得了一定的研究成果[56−82].除了对这些特定测试领域的 TCP 技术展开更为深入的研究以外,我们还可以积极寻找可以使 TCP 技术发挥效果的新兴特定测试领域,例如并发程序测试、云计算平台和物联网应用等.结合这些新领域的场景特征和传统 TCP 技术的已有研究成果,有助于提出适用于这些特定领域内的行之有效的 TCP 技术.

© 中国科学院软件研究所 http://www.jos.org.cn

(5) 基于缺陷定位技术的 TCP 技术研究.缺陷定位是一种高效、自动的软件调试技术,并在近些年得到国内外学者的关注.已有研究成果表明,测试用例集的质量对缺陷定位效果存在重要影响.其下一步研究工作可以考虑从两个角度进行展开:首先,进一步考察不同 TCP 排序准则和排序策略对传统缺陷定位技术的影响;其次,传统 TCP 技术以提高测试用例集的早期缺陷检测率作为排序优化目标,若考虑缺陷定位应用,则 TCP 技术需要以提高缺陷定位效果作为排序优化目标,并需对传统的 TCP 排序准则和排序策略的设定进行深入的研究.

(6) 与企业内部测试流程相结合.将 TCP 的已有研究成果与企业的内部测试流程进行紧密结合,目前,研究成果存在一些不足:首先,已有 TCP 技术一般采用对照实验(controlled experiment)方式进行有效性评估,采用的评测程序规模较小,所得结论不一定适用于来自企业的大型软件产品;其次,已有 TCP 工具也未充分考虑到与软件企业目前已有的开发、测试和调试流程相结合.若要将学术界的研究成功应用到企业界,则需要进一步提高测试工具的自动化程度、改善工具的用户界面,并保证工具具有一定的鲁棒性.若对上述挑战提出有效解决方案,则将有效提高研究成果的应用价值,并大幅度提高企业内的测试和调试效率.

References:

[1] Beizer B. Software Testing Techniques. New York: Van No strand Reinhold, 1990.

[2] Leung H, White L. Insights into regression testing. In: Proc. of the Int'l Conf. on Software Maintenance. ACM Press, 1989. 60−69. [doi: 10.1109/ICSM.1989.65194]

[3] Rothermel G, Untch RJ, Chu C. Prioritizing test cases for regression testing. IEEE Trans. on Software Engineering, 2001,27(10): 929−948. [doi: 10.1109/32.962562]

[4] Yoo S, Harman M. Regression testing minimization, selection and prioritization: A survey. Software Testing, Verification & Reliability, 2012,22(2):67−120. [doi: 10.1002/stvr.430]

[5] Harrold M, Orso A. Retesting software during development and maintenance. In: Proc. of the Frontiers of Software Maintenance. IEEE Press, 2008. 99−108. [doi: 10.1109/FOSM.2008.4659253]

[6] Wong W, Horgan J, London S, Agrawal H. A study of effective regression testing in practice. In: Proc. of the Int'l Symp. on Software Reliability Engineering. IEEE Press, 1997. 264−274. [doi: 10.1109/ISSRE.1997.630875]

[7] Elbaum S, Malishevsky AG, Rothermel G. Prioritizing test cases for regression testing. In: Proc. of the Int'l Symp. on Software Testing and Analysis. ACM Press, 2000. 102−112. [doi: 10.1145/347324.348910]

[8] Rothermel G, Untch RH, Chu C, Harrold MJ. Test case prioritization: An empirical study. In: Proc. of the Int'l Conf. on Software Maintenance. IEEE Press, 1999. 179−188. [doi: 10.1109/ICSM.1999.792604]

[9] Elbaum S, Malishevsky AG, Rothermel G. Test case prioritization: A family of empirical studies. IEEE Trans. on Software Engineering, 2002,28(2):159−182. [doi: 10.1109/32.988497]

[10] Jones JA, Harrold MJ. Test-Suite reduction and prioritization for modified condition/decision coverage. IEEE Trans. on Software Engineering, 2003,29(3):195−209. [doi: 10.1109/TSE.2003.1183927]

[11] Do H, Rothermel G, Kinneer A. Empirical studies of test case prioritization in a JUnit testing environment. In: Proc. of the Int'l Symp. on Software Reliability Engineering. IEEE Press, 2004. 113−124. [doi: 10.1109/ISSRE.2004.18]

[12] Do H, Rothermel G, Kinneer A. Prioritizing JUnit test cases: An empirical assessment and cost-benefits analysis. Empirical Software Engineering, 2006,11(1):33−70. [doi: 10.1007/s10664-006-5965-8]

[13] Zhang L, Zhou J, Hao D, Zhang L, Mei H. Prioritizing JUnit test cases in absence of coverage information. In: Proc. of the Int'l Conf. on Software Maintenance. IEEE Press, 2009. 19−28. [doi: 10.1109/ICSM.2009.5306350]

[14] Mei H, Hao D, Zhang LM, Zhang L, Zhou J, Rothermel G. A static approach to prioritizing JUnit test cases. IEEE Trans. on Software Engineering, 2012,38(6):1258−1275. [doi: 10.1109/TSE.2011.106]

[15] Li Z, Harman M, Hierons RM. Search algorithms for regression test case prioritization. IEEE Trans. on Software Engineering, 2007,33(4):225−237. [doi: 10.1109/TSE.2007.38]

[16] Jiang B, Zhang ZY, Chan WK, Tse TH. Adaptive random test case prioritization. In: Proc. of the Int'l Conf. on Automated Software Engineering. IEEE Press, 2009. 233−244. [doi: 10.1109/ASE.2009.77]

© 中国科学院软件研究所 http://www.jos.org.cn

[17] Voas J. PIE: A dynamic failure-based technique. IEEE Trans. on Software Engineering, 1992,18(8):717−727. [doi: 10.1109/32.153381]

[18] Jeffrey D, Gupta N. Test case prioritization using relevant slices. In: Proc. of the Annual Int'l Computer Software and Applications Conf. IEEE Press, 2006. 411−420. [doi: 10.1109/COMPSAC.2006.80]

[19] Jeffrey D, Gupta N. Experiments with test case prioritization using relevant slices. Journal of System and Software, 2008,81(2): 196−221. [doi: 10.1016/j.jss.2007.05.006]

[20] Elbaum S, Malishevsky A, Rothermel G. Incorporating varying test costs and fault severities into test case prioritization. In: Proc. of the Int'l Conf. on Software Engineering. IEEE Press, 2001. 329−338. [doi: 10.1109/ICSE.2001.919106]

[21] Ma ZK, Zhao JJ. Test case prioritization based on analysis of program structure. In: Proc. of the Asia-Pacific Software Engineering Conf. IEEE Press, 2008. 471−478. [doi: 10.1109/APSEC.2008.63]

[22] Elbaum S, Rothermel G, Kanduri S, Malishevsky AG. Selecting a cost-effective test case prioritization technique. Software Quality Control, 2004,12(3):185−210. [doi: 10.1023/B:SQJO.0000034708.84524.22]

[23] Arafeen M, Do H. Adaptive regression testing strategy: An empirical study. In: Proc. of the Int'l Symp. on Software Reliability Engineering. IEEE Press, 2011. 130−139. [doi: 10.1109/ISSRE.2011.29]

[24] Li S, Bian N, Chen Z, You D, He Y. A simulation study on some search algorithms for regression test case prioritization. In: Proc. of the Int'l Conf. on Quality Software. 2010. 72−81. [doi: 10.1109/QSIC.2010.15]

[25] Mirarab S, Tahvildari L. A prioritization approach for software test cases based on bayesian networks. In: Proc. of the Int'l Conf. on Fundamental Approaches to Software Engineering. Springer-Verlag, 2007. 276−290. [doi: 10.1007/978-3-540-71289-3_22]

[26] Mirarab S, Tahvildari L. An empirical study on Bayesian network-based approach for test case prioritization. In: Proc. of the Int'l Conf. on Software Testing, Verification, and Validation. IEEE Press, 2008. 278−287. [doi: 10.1109/ICST.2008.57]

[27] Leon D, Podgurski A. A comparison of coverage-based and distribution-based techniques for filtering and prioritizing test cases. In: Proc. of the Int'l Symp. on Software Reliability Engineering. IEEE Press, 2003. 442−453.

[28] Carlson R, Hyunsook D, Denton A. A clustering approach to improving test case prioritization: An industrial case study. In: Proc. of the Int'l Conf. on Software Maintenance. IEEE Press, 2011. 382−391. [doi: 10.1109/ICSM.2011.6080805]

[29] Huang YC, Peng KL, Huang CY. A history-based cost-cognizant test case prioritization technique in regression testing. Journal of System and Software, 2012,85(3):626−637. [doi: 10.1016/j.jss.2011.09.063]

[30] Tonella P, Avesani P, Susi A. Using the case-based ranking methodology for test case prioritization. In: Proc. of the Int'l Conf. on Software Maintenance. IEEE Press, 2006. 123−133. [doi: 10.1109/ICSM.2006.74]

[31] Yoo S, Harman M, Tonella P, Susi A. Clustering test cases to achieve effective and scalable prioritisation incorporating expert knowledge. In: Proc. of the Int'l Symp. on Software Testing and Analysis. ACM Press, 2009. 201−212. [doi: 10.1145/1572272.1572296]

[32] Srivastava A, Thiagarajan J. Effectively prioritizing tests in development environment. In: Proc. of the Int'l Symp. on Software Testing and Analysis. ACM Press, 2002. 97−106. [doi: 10.1145/566172.566187]

[33] Sherriff M, Lake M, Williams L. Prioritization of regression tests using singular value decomposition with empirical change records. In: Proc. of the Int'l Symp. on Software Reliability. IEEE Press, 2007. 81−90. [doi: 10.1109/ISSRE.2007.20]

[34] Ramanathan MK, Koyuturk, M, Grama A, Jagannathan S. PHALANX: A graph-theoretic framework for test case prioritization. In: Proc. of the Symp. on Applied Computing. ACM Press, 2008. 667−673. [doi: 10.1145/1363686.1363848]

[35] Korel B, Tahat L, Harman M. Test prioritization using system models. In: Proc. of the Int'l Conf. on Software Maintenance. IEEE Press, 2005. 559−568. [doi: 10.1109/ICSM.2005.87]

[36] Korel B, Koutsogiannakis G, Tahat L. Model-Based test prioritization heuristic methods and their evaluation. In: Proc. of the Int'l Workshop on Advances in Model-Based Testing. ACM Press, 2007. 34−43. [doi: 10.1145/1291535.1291539]

[37] Korel B, Koutsogiannakis G, Tahat L. Application of system models in regression test suite prioritization. In: Proc. of the Int'l Conf. on Software Maintenance. IEEE Press, 2008. 247−256. [doi: 10.1109/ICSM.2008.4658073]

[38] Kundu D, Sarma M, Samanta D, Mall R. System testing for object-oriented systems with test case prioritization. Software Testing, Verification and Reliability, 2009,19(4):297−333. [doi: 10.1002/stvr.407]

© 中国科学院软件研究所 http://www.jos.org.cn

[39] Srikanth H, Williams L, Osborne J. System test case prioritization of new and regression test cases. In: Proc. of the Int'l Symp. on Empirical Software Engineering. IEEE Press, 2005. 64−73. [doi: 10.1109/ISESE.2005.1541815]

[40] Qu B, Nie CH, Xu BW, Zhang XF. Test case prioritization for black box testing. In: Proc. of the Annual Int'l Computer Software and Applications Conf. IEEE Press, 2007. 465−474. [doi: 10.1109/COMPSAC.2007.209]

[41] Qu B, Nie CH, Xu BW. Test case prioritization based on test suite design information. Chinese Journal of Computers, 2008,31(3): 431−439 (in Chinese with English abstract).

[42] Zhang XF, Nie CH, Xu BW, Qu B. Test case prioritization based on varying testing requirement priorities and test case costs. In: Proc. of the Int'l Conf. on Quality Software. IEEE Press, 2007. 15−24. [doi: 10.1109/QSIC.2007.4385476]

[43] Krishnamoorthi R, Sahaaya Arul Mary SA. Factor oriented requirement coverage based system test case prioritization of new and regression test cases. Information and Software Technology, 2009,51(4):799−808. [doi: 10.1016/j.infsof.2008.08.007]

[44] Yu YT, Lau MF. Fault-Based test suite prioritization for specification-based testing. Information and Software Technology, 2012, 54(2):179−202. [doi: 10.1016/j.infsof.2011.09.005]

[45] Kim JM, Porter A. A history-based test prioritization technique for regression testing in resource constrained environments. In: Proc. of the Int'l Conf. on Software Engineering. ACM Press, 2002. 119−129. [doi: 10.1145/581339.581357]

[46] Walcott KR, Soffa ML, Kapfhammer GM, Roos RS. Time-Aware test suite prioritization. In: Proc. of the Int'l Symp. on Software Testing and Analysis. ACM Press, 2006. 1−12. [doi: 10.1145/1146238.1146240]

[47] Alspaugh S, Walcott KR, Belanich M, Kapfhammer GM, Soffa ML. Efficient time-aware prioritization with knapsack solvers. In: Proc. of the Int'l Workshop on Empirical Assessment of Software Engineering Languages and Technologies. ACM Press, 2007. 13−18. [doi: 10.1145/1353673.1353676]

[48] Zhang L, Hou SS, Guo C, Xie T, Mei H. Time-Aware test-case prioritization using integer linear programming. In: Proc. of the Int'l Symp. on Software Testing and Analysis. ACM Press, 2009. 213−224. [doi: 10.1145/1572272.1572297]

[49] You DJ, Chen ZY, Xu BW, Luo B, Zhang C. An empirical study on the effectiveness of time-aware test case prioritization techniques. In: Proc. of the Symp. on Applied Computing. ACM Press, 2011. 1451−1456. [doi: 10.1145/1982185.1982497]

[50] Do H, Mirarab S, Tahildari L, Rothermel G. An empirical study of the effect of time constraints on the cost-benefits of regression testing. In: Proc. of the Int'l Symp. on Foundations of Software Engineering. ACM Press, 2008. 71−82. [doi: 10.1145/1453101. 1453113]

[51] Do H, Mirarab S, Tahildari L, Rothermel G. The effects of time constraints on test case prioritization: A series of controlled experiments. IEEE Trans. on Software Engineering, 2010,36(5):593−617. [doi: 10.1109/TSE.2010.58]

[52] Do H, Elbaum S, Rothermel G. Supporting controlled experimentation with testing techniques: an infrastructure and its potential impact. Empirical Software Engineering, 2005,10(4):405−435. [doi: 10.1007/s10664-005-3861-2]

[53] Hutchins M, Foster H, Goradia T, Ostrand T. Experiments on the effectiveness of dataflow- and control- flow-based test adequacy criteria. In: Proc. of the Int'l Conf. on Software Engineering. ACM Press, 1994. 191−200.

[54] Andrews JH, Briand LC, Labiche Y. Is mutation an appropriate tool for testing experiments? In: Proc. of the Int'l Conf. on Software Engineering. ACM Press, 2005. 402−411. [doi: 10.1145/1062455.1062530]

[55] Do H, Rothermel G. On the use of mutation faults in empirical assessments of test case prioritization techniques. IEEE Trans. on Software Engineering, 2006,32(9):733−752. [doi: 10.1109/TSE.2006.92]

[56] Bryce RC, Colbourn CJ. Test prioritization for pairwise interaction coverage. In: Proc. of the Int'l Workshop on Advances in Model-based Testing. ACM Press, 2005. 1−7. [doi: 10.1145/1083274.1083275]

[57] Bryce RC, Colbourn CJ. Prioritized interaction testing for pair-wise coverage with seeding and constraints. Information and Software Technology, 2006,48(10):960−970. [doi: 10.1016/j.infsof.2006.03.004]

[58] Wang ZY, Xu BW, Chen L, Chen ZY. Cost-Effective combinatorial test case prioritization for varying combination weights. In: Proc. of the Int'l Conf. on Software & Knowledge Engineering. Redwood: Knowledge Systems Institute Graduate School, 2010. 273−278.

[59] Wang ZY, Chen L, Xu BW, Huang Y. Cost-Cognizant combinatorial test case prioritization. Int'l Journal of Software Engineering and Knowledge Engineering, 2011,21(6):829−854. [doi: 10.1142/S0218194011005499]

© 中国科学院软件研究所 http://www.jos.org.cn

[60] Qu X, Cohen MB, Woolf K. Combinatorial interaction regression testing: A study of test case generation and prioritization. In: Proc. of the Int'l Conf. on Software Maintenance. IEEE Press, 2007. 255−264. [doi: 10.1109/ICSM.2007.4362638]

[61] Qu X, Cohen MB, Rothermel G. Configuration-Aware regression testing: An empirical study of sampling and prioritization. In: Proc. of the Int'l Symp. on Software Testing and Analysis. ACM Press, 2008. 75−86. [doi: 10.1145/1390630.1390641]

[62] Srikanth H, Cohen MB, Qu X. Reducing field failures in system configurable software: Cost-based prioritization. In: Proc. of the Int'l Conf. on Software Reliability Engineering. IEEE Press, 2009. 61−70. [doi: 10.1109/ISSRE.2009.26]

[63] Bryce RC, Memon AM. Test suite prioritization by interaction coverage. In: Proc. of the Workshop on Domain Specific Approaches to Software Test Automation. ACM Press, 2007. 1−7. [doi: 10.1145/1294921.1294922]

[64] Huang CY, Chang JR, Chang YH. Design and analysis of GUI test-case prioritization using weighted-based methods. Journal of System and Software, 2010,83(4):646−659. [doi: 10.1016/j.jss.2009.11.703]

[65] Sampath S, Bryce RC, Viwanath G, Kandimalla V, Koru AG. Prioritizing user-session-based test cases for Web applications testing. In: Proc. of the Int'l Conf. on Software Testing, Verification, and Validation. IEEE Press, 2008. 141−150. [doi: 10.1109/ICST.2008.42]

[66] Garg D, Datta A. Test case prioritization due to database changes in Web applications. In: Proc. of the Int'l Conf. on Software Testing, Verification and Validation. IEEE Press, 2012. 726−730. [doi: 10.1109/ICST.2012.163]

[67] Bryce RC, Sampath S, Memon AM. Developing a single model and test prioritization strategies for event-driven software. IEEE Trans. on Software Engineering, 2011,37(1):48−64. [doi: 10.1109/TSE.2010.12]

[68] Hou SS, Zhang L, Xie T, Mei H, Sun JS. Applying interface-contract mutation in regression testing of component-based software. In: Proc. of the Int'l Conf. on Software Maintenance. IEEE Press, 2007. 174−183. [doi: 10.1109/ICSM.2007.4362630]

[69] Hou SS, Zhang L, Xie T, Sun JS. Quota-Constrained test-case prioritization for regression testing of service-centric systems. In: Proc. of the Int'l Conf. on Software Maintenance. IEEE Press, 2008. 257−266. [doi: 10.1109/ICSM.2008.4658074]

[70] Mei LJ, Chan WK, Tse TH, Merkel RG. Tag-Based techniques for black-box test case prioritization for service testing. In: Proc. of the Int'l Conf. on Quality Software. IEEE Press, 2009. 21−30. [doi: 10.1109/QSIC.2009.12]

[71] Mei LJ, Chan WK, Tse TH, Merkel RG. XML-Manipulating test case prioritization for XML-manipulating services. Journal of System and Software, 2011,84(4):603−619. [doi: 10.1016/j.jss.2010.11.905]

[72] Mei LJ, Zhang ZY, Chan WK, Tse TH. Test case prioritization for regression testing of service-oriented business applications. In: Proc. of the Int'l Conf. on World Wide Web. ACM Press, 2009. 901−910. [doi: 10.1145/1526709.1526830]

[73] Chen L, Wang ZY, Xu L, Lu HM, Xu BW. Test case prioritization for Web service regression testing. In: Proc. of the Int'l Symp. on Service Oriented System Engineering. IEEE Press, 2010. 173−178. [doi: 10.1109/SOSE.2010.27]

[74] Nguyen CD, Marchetto A, Tonella P. Test case prioritization for audit testing of evolving Web services using information retrieval techniques. In: Proc. of the Int'l Conf. on Web Services. IEEE Press, 2011. 636−643. [doi: 10.1109/ICWS.2011.75]

[75] Zhai K, Jiang B, Chan W, Tse T. Taking advantage of service selection: A study on the testing of location-based Web services through test case prioritization. In: Proc. of the Int'l Conf. on Web Services. IEEE Press, 2010. 211−218. [doi: 10.1109/ICWS.2010.98]

[76] Zhai K, Chan WK. Point-of-Interest aware test case prioritization: Methods and experiments. In: Proc. of the Int'l Conf. on Quality Software. IEEE Press, 2010. 449−456. [doi: 10.1109/QSIC.2010.63]

[77] Jiang B, Chan WK. On the integration of test adequacy, test case prioritization, and statistical fault localization. In: Proc. of the Int'l Conf. on Quality Software. IEEE Press, 2010. 377−384. [doi: 10.1109/QSIC.2010.64]

[78] Jiang B, Chan WK, Tse TH. On practical adequate test suites for integrated test case prioritization and fault localization. In: Proc. of the Int'l Conf. on Quality Software. IEEE Press, 2011. 21−30. [doi: 10.1109/QSIC.2011.37]

[79] Jiang B, Zhang ZY, Chan WK, Tse TH, Chen TY. How well does test case prioritization integrate with statistical fault localization? Information Software Technology, 2012,54(7):739−758. [doi: 10.1016/j.infsof.2012.01.006]

[80] Gonzalez-Sanchez A, Piel E, Gross H, van Gemund A. Prioritizing tests for software fault localization. In: Proc. of the Int'l Conf. on Quality Software. IEEE Press, 2010. 42−51. [doi: 10.1109/QSIC.2010.28]

[81] Gonzalez-Sanchez A, Abreu R, Gross H, Gemund A. Prioritizing tests for fault localization through ambiguity group reduction. In: Proc. of the Int'l Conf. on Automated Software Engineering. IEEE Press, 2011. 83−92. [doi: 10.1109/ASE.2011.6100153]

© 中国科学院软件研究所 http://www.jos.org.cn

[82] Kim S, Baik J. An effective fault aware test case prioritization by incorporating a fault localization technique. In: Proc. of the Int'l Symp. on Empirical Software Engineering and Measurement. ACM Press, 2010. 1−10. [doi: 10.1145/1852786.1852793]

[83] Just R, Kapfhammer GM, Schweiggert F. Using non-redundant mutation operators and test suite prioritization to achieve efficient and scalable mutation analysis. In: Proc. of the Int'l Symp. on Software Reliability Engineering. IEEE Press, 2012. 11−20. [doi: 10.1109/ISSRE.2012.31]

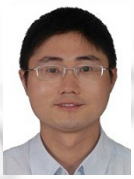
[84] Staats M, Loyola P, Rothermel G. Oracle-Centric test case prioritization. In: Proc. of the Int'l Symp. on Software Reliability Engineering. IEEE Press, 2012. 311−320. [doi: 10.1109/ISSRE.2012.13]

附中文参考文献:

[41] 屈波,聂长海,徐宝文.基于测试用例设计信息的回归测试优先级算法.计算机学报,2008,31(3):431−439.

陈翔(1980－),男,江苏南通人,博士,讲师,CCF 会员,主要研究领域为软件测试,程序分析.

E-mail: xchencs@ntu.edu.cn



鞠小林(1976－),男,博士生,讲师,CCF 会员,主要研究领域为软件测试,缺陷定位.

E-mail: ju.xl@ntu.edu.cn

陈继红(1966－),男,副教授,主要研究领域为软件测试.

E-mail: chen.jh@ntu.edu.cn

顾庆(1972－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件质量保障.

E-mail: guq@nju.edu.cn

© 中国科学院软件研究所 http://www.jos.org.cn