自动化测试用例测试失败类型分析

姜 文,刘立康
(西安电子科技大学 通信工程学院,陕西 西安 710071)

摘 要:随着计算机技术的发展,软件迭代开发模式在软件开发与测试过程中占的比重越来越大。软件迭代开发过程中大量采用自动化测试,在测试环境上进行测试脚本连跑;通常会有一定数量的测试脚本失败,需要对这些脚本进行失败分析。这是一项十分重要的工作,否则软件产品无法继续开发,也无法保证软件产品的质量。依据软件迭代开发和测试工作实践,归纳总结了自动化测试用例失败的类型,介绍了各种类型测试脚本失败的工作实例;叙述了自动化测试用例失败分析涉及的角色和软件迭代开发过程中自动化测试应用场景;详细叙述了自动化测试用例失败的定位和分析处理;最后叙述了测试工作的改进。工作实践表明做好软件自动化测试用例失败分析工作,有助于提高软件产品开发效率和提升软件产品质量。

关键词:迭代开发;自动化测试;测试用例;脚本;失败分析
中图分类号:TP311.5 **文献标识码**:A **文章编号**:1673-629X(2020)04-0008-06
doi:10.3969/j.issn.1673-629X.2020.04.002

Testing Failure Type Analysis of Automation Test Case

JIANG Wen,LIU Li-kang
(School of Telecommunication Engineering,Xidian University,Xi'an 710071,China)

Abstract:With the development of computer technology,the proportion of software iterative development mode is becoming larger in the process of software development and test. A large number of software automated test are used in the iterative development process when test scripts running in test environment; a certain number of test script failure will appear,so failure analysis of these scripts needed to be done. This is a quite important work,otherwise,the software products won't be able to continue to develop,and the quality of software products is unable to guarantee. On the basis of iterative development and software testing work practice,we summarize the failure types of automation test case,introduce the instance of all kinds of test script failure,describe the involving role in failure analysis of automated test cases and automation test application scenario in software iterative development process,and detail the failure positioning and analysis of automated test cases. Finally,we describe the improvement of testing work. The practice shows that the failure analysis of automation software test cases is helpful to improve the efficiency of software product development and the quality of software products.

Key words:iterative development;automation test;test case;script;failure analysis

0 引 言

随着计算机技术的发展,计算机软件的使用范围越来越广泛,迭代开发模式在软件开发与测试过程中获得广泛应用。软件测试[1-2]过程中大量采用自动化测试,通过自动化测试脚本连跑的方式对软件产品进行测试。

使用自动化连跑进行软件测试,不仅能覆盖到当前版本的新特性,也能够覆盖版本的历史特性,测试覆盖率高。这样可以节省大量的测试人力、物力,也提升了测试效率。

在测试环境上连跑测试脚本,通常会有一定数量的测试脚本失败,对这些脚本进行失败分析是一项十分重要的工作,否则软件产品无法继续开发,也无法保证软件产品的质量。

软件缺陷问题是测试脚本失败的原因之一,但是还有其他原因导致测试脚本失败,需要进行细致的定位、分析和处理。对于软件自动化测试而言,测试脚本的失败分析是重要的核心环节。

收稿日期:2019-06-15 **修回日期**:2019-10-18 **网络出版时间**:2019-12-18
基金项目:国家自然科学基金(61403291)
作者简介:姜 文(1986-),女,高级工程师,硕士,CCF 会员(E200032324M),研究方向为图像处理、软件工程和网络通信;刘立康,副教授,研究方向为数字通信、图像处理和软件工程。
网络出版地址:http://kns.cnki.net/kcms/detail/61.1450.TP.20191218.1115.074.html

1　软件迭代开发与自动化测试

1.1　软件迭代开发

迭代开发[3-10]也被称作迭代增量式开发,将整个软件项目开发分成多个小的迭代(一般 2 至 4 周),每一次迭代都由需求分析、设计、实现和测试在内的多个活动组成,每一次迭代都可以生成一个稳定和被验证过的软件版本。迭代开发可以及时应对软件开发过程中的各种技术变更;有助于早期发现各种技术问题,及时消除风险;持续从客户获得反馈,使产品更加符合客户的需求;有利于人力资源的调度安排,提高工作效率。

1.2　软件自动化测试

自动化测试,简而言之就是将手工测试执行的工作改为自动执行。它通过使用某一自动化测试工具和自动化测试脚本来执行测试工作。自动化测试适合执行各种频繁的测试,使测试任务的执行比手动方式更高效。自动化测试可以避免因人为而造成的疏忽和错误。

自动化测试用例运行结束后,会出现两种可能情况:正常结束(Passed)、异常结束(Failed)。Passed 说明测试结果和预期结果相吻合,软件特性通过了测试;Failed 则说明测试失败,相关检查点出现错误或者测试脚本抛出异常而终止测试。

2　自动化测试用例失败分析

在测试环境上执行自动化测试脚本连跑,通常会有一定数量的测试脚本失败(Failed)。自动化测试用例失败分析就是对失败的脚本进行失败定位、分析和处理,查找测试脚本失败的原因。

2.1　自动化测试用例失败类型

依据软件迭代开发和测试实践,归纳总结出自动化测试用例失败类型。通常测试脚本失败类型分为五类,如表 1 所示。

表 1　自动化测试用例失败类型

序号	失败类型	导致的测试脚本失败原因
1	测试环境问题	软件测试环境搭建和配置存在问题
2	软件版本包问题	软件版本包选择错误,或者提交的软件测试版本包存在构建问题
3	测试脚本问题	测试脚本存在问题
4	脚本概率性失败问题	测试脚本中存在某种隐藏的问题,在一定的外部条件下触发,导致测试脚本失败
5	软件缺陷问题	软件存在缺陷,测试脚本检测到了软件存在的问题,需要修改软件代码

2.2　自动化测试用例测试失败的实例

以下是自动化测试工作中遇到的一些测试脚本失败实例。

2.2.1　测试环境问题

某软件产品在迭代 5 结束时,自动化工厂扩充了部分新的测试执行机,在新执行机上执行脚本连跑任务。由于自动化工厂测试工程师的疏漏,将一个通用变量的值删除之后,没有赋值,导致大量使用该变量的软件 S 特性脚本运行失败,处理了该问题后,所有脚本运行通过。

2.2.2　软件版本包问题

某软件产品在迭代 1 结束之后,连跑新增特性 S 的脚本,结果所有脚本均运行失败。测试工程师定位之后发现,测试环境上安装的版本包,并不支持特性 S,连跑失败是由于没有使用正确的版本包。自动化工厂测试工程师获取最新出的版本包,重新安装在测试环境中,所有脚本运行通过。

2.2.3　脚本存在配置残留问题

某软件产品在迭代 3 结束时,对软件 T 特性进行自动化安全测试,由于某个脚本存在配置残留问题,脚本执行之后未恢复环境,导致连跑过程中一系列脚本运行失败。测试工程师通过分析发现了存在配置残留的脚本,修改了存在配置残留的脚本;排查这一批脚本,如果有配置残留问题全部修改;之后脚本连跑通过,脚本全部合入脚本库。

2.2.4　脚本概率性失败问题

某软件产品在迭代 6 结束之后,软件 Q 特性部分脚本,会出现概率性失败的问题。测试工程师分析发现,某个脚本连跑过程中调用了一个自研工具,脚本运行结束之后,已经调用的自研工具进程不一定会自动释放,从而导致脚本概率性失败。修改该脚本,在脚本中添加自动杀死自研工具进程的功能。之后脚本连跑通过,再没发生概率性失败问题。

2.2.5　软件代码缺陷导致脚本失败

某软件产品在迭代 6 结束之后,测试软件历史特性 H 时,某些测试脚本在连跑过程中失败。测试工程师分析发现不是环境问题或者脚本问题导致的失败。将该问题反馈给软件开发工程师,软件开发工程师定位发现是由于迭代 6 新开发的特性 Q 中新合入的代码影响了原有特性 H 的功能,属于新引入的软件缺陷。

测试工程师提交问题单,软件开发工程师修改软件代码,完成验证之后将已修改的源代码提交到版本库中。持续集成工程师提交新的版本包之后,测试工程师完成问题单回归测试,关闭该问题单。

3 自动化测试用例失败分析涉及的角色

自动化测试连跑失败分析过程中,参与的人员包括:软件测试架构师、软件测试工程师、持续集成工程师、自动化工厂测试工程师、软件系统工程师、软件开发工程师。各角色分工协作共同完成自动化测试用例失败分析,如图 1 所示。

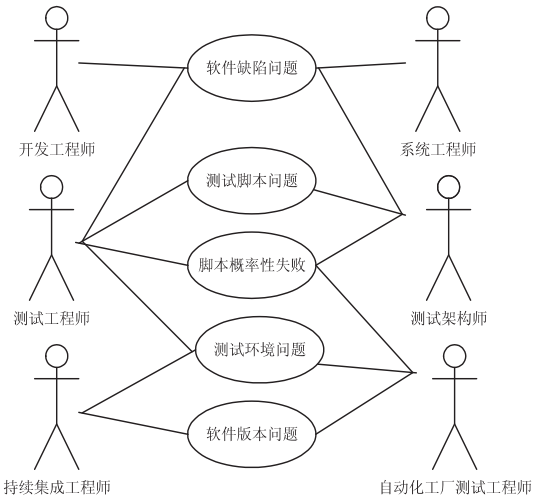

图 1 自动化测试用例失败分析涉及的角色

3.1 测 试

3.1.1 软件测试架构师

测试架构师(TSE)编写软件产品测试策略和软件测试用例。软件测试架构师与测试经理将软件产品根据功能划分若干个特性责任田,分配测试工程师担任责任田主。与软件测试工程师共同处理软件自动化测试过程中存在的各种问题。

3.1.2 软件测试工程师(责任田主)

软件测试工程师根据设计场景文档、测试策略和测试用例编写自动化测试脚本。搭建测试环境,安装软件版本包,进行软件测试。软件测试工程担任某个软件特性责任田主,在自动化脚本连跑完成之后,对失败的脚本进行失败分析,处理相关技术问题。对检测到的软件缺陷,提交问题单,跟踪问题单,直到问题单处理完毕。

3.1.3 持续集成工程师

软件迭代开期间,持续集成工程师从版本库更新源代码,搭建集成构建工程;进行静态检查、软件产品模块编译、软件版本包出包;向自动化工厂提交冒烟测试任务。与相关人员共同处理脚本失败问题,主要关注软件版本包相关问题。

3.1.4 自动化工厂测试工程师

管理自动化工厂,在自动化工厂中搭建软件测试环境,安装持续集成工程师提供的版本包,从版本库下载相关的自动化测试用例脚本;执行自动化脚本连跑。连跑结束之后,汇总测试结果,将失败脚本推送给测试工程师进行失败分析,参与脚本失败原因分析定位,关注测试环境的搭建和配置存在的问题。

3.2 开 发

3.2.1 软件系统工程师

系统工程师(SE)给出可行的设计方案;完成软件需求设计、架构设计和详细设计;提供给软件开发工程师进行软件编码。在软件迭代开发过程中,和软件开发工程师共同处理软件缺陷问题。

3.2.2 软件开发工程师

软件开发工程师根据软件产品设计场景文档编写代码。源代码调试、自验之后,将源代码合入版本库。处理持续集成工程师和测试工程师提交的软件缺陷问题,定位问题产生的原因,修改代码,重新将代码合入版本库。对于一些复杂的问题需要软件系统工程师参与定位处理。

4 软件迭代开发过程中自动化测试应用场景

4.1 基于持续集成的冒烟测试

冒烟测试[11]是对软件版本包进行详细测试之前的预测试,执行冒烟测试可以快速验证软件基本功能是否有缺陷。如果冒烟测试不能通过,则不必做进一步的测试。

持续集成中的冒烟测试是每日构建的一部分,通过自动化工厂执行自动化脚本连跑任务完成软件版本包预测试。

4.2 基于持续集成的覆盖率测试

迭代开发期间,持续集成中的冒烟测试任务调用测试用例脚本数量需要不断增加,从而保证测试用例对新增代码覆盖率[12]要求。通常要求测试用例对新增代码覆盖率大于 65%,对于整体的代码覆盖率,各软件产品根据产品自身情况可以灵活处理。

通常每周进行一次基于持续集成的软件覆盖率测试,可以快速完成软件代码覆盖率检查。测试架构师根据代码覆盖率及时补充测试用例。

4.3 迭代完成之后的自动化测试

每次迭代完成之后,需要对软件进行全面测试[13-14],执行全部自动化脚本连跑(预测试)。

4.4 软件验收前的自动化测试

软件全部迭代开发完成之后,软件进入验收交付阶段,需要对软件进行全面测试,首先执行全部自动化脚本连跑(预测试)。

4.5 软件交付后的自动化测试

软件交付后,通常需要处理软件的各种问题,有时还需要给软件制作补丁,问题处理之后,需要对软件进行全面测试,首先执行全部自动化脚本连跑(预测试)。

4.6　回归自动化测试

回归测试(Regression Testing)是指软件修改之后,重新执行先前的测试,以保证软件修改的正确性。在前五种测试中,出现测试用例失败,需要处理出现的问题,问题处理完成之后,进行回归测试,需要执行全部自动化脚本连跑。

5　自动化测试用例失败的定位、分析和处理

5.1　失败分析工具—DAC

DAC(Data Analysis Center)是一款软件缺陷跟踪、缺陷管理、数据分析的管理工具[15]。通过 DAC 的智能分析中心可以查看日志的历史版本,查看任务列表,查看分析报告,查看测试用例详细信息。

可以和自动化工厂的软件模块相互关联,通过跳转到测试信息管理中心 TMSS(Test Management Service System),查看 TMSS 中测试用例的相关信息。自动化工厂测试工程师也可以通过 DAC 工具,处理自动化测试中出现的缺陷问题。

测试工程师可以登录 DAC 工具,通过特性过滤出自己需要分析的失败脚本,查看各种相关信息,处理脚本失败问题。

5.2　软件产品特性责任田

软件产品根据功能划分为若干个特性责任田,测试用例根据软件产品特性归属某个责任田。测试架构师梳理出产品特性列表,划分特性责任田。测试经理和测试架构师根据目前测试工程师的人数、能力将特性责任田划分给测试工程师。测试工程师担任这些特性的责任田主,并以交叉方式将确定每个脚本所属责任田的备份分析田主,以保证责任田在每次分析时均不会落空。自动化脚本的责任田主名单由测试经理提交给自动化工厂测试工程师,连跑之后的失败脚本分析就按照这个名单进行脚本推送。

5.3　自动化测试用例连跑失败分析流程

自动化测试用例连跑失败分析的流程通常分为以下几个步骤:搭建测试环境、自动化测试脚本连跑、将失败脚本提交相关人员、脚本失败原因的定位分析、处理存在的问题、提交脚本失败分析报告。自动化测试用例失败分析流程如图 2 所示。

5.3.1　搭建测试环境

软件测试环境通常在云环境中搭建,首先需要申请相关的虚拟网络资源,自动化工厂测试工程师负责测试环境的搭建、配置和管理;持续集成工程师提交软件测试版本包。

5.3.2　自动化测试脚本连跑

自动化工厂测试工程师,回收由测试工程师和开发工程师在 TICC(Test Integration Control Center,集成测试控制中心)上占用的环境,清理测试环境,卸载原有版本包。安装从指定路径下获取的新版本包,加上业务配置之后下发测试任务,在自动化工厂的测试环境上执行测试脚本连跑。

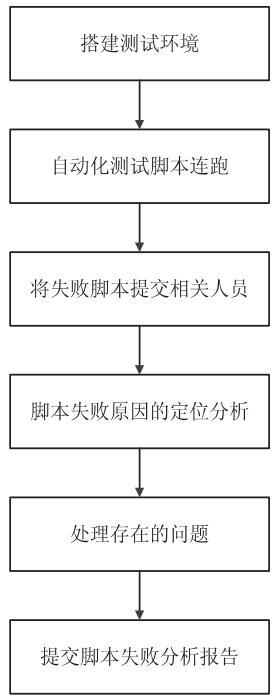

图 2　自动化测试用例失败分析流程

5.3.3　将失败的脚本提交相关人员

自动化脚本连跑完成后,自动化工厂的测试工程师汇总自动化脚本的运行结果,根据产品特性责任田田主名单,将失败脚本通过 DAC 工具推送给测试工程师(脚本的责任田主),通过邮件方式将连跑失败的测试脚本运行结果、运行日志以及运行脚本的执行机所在 DAC 网页地址发送给各责任田主进行分析。基于持续集成的自动化测试,需要同时将测试脚本运行结果通知持续集成工程师。同时将相关信息通知测试经理。

5.3.4　自动化脚本失败原因的定位分析

测试工程师收到推送过来的失败脚本,通过特性过滤出自己需要分析的失败脚本,在 DAC 工具上查看已收集到的连跑失败日志。通过日志确认脚本失败问题点,将脚本放到运行失败的执行机上,再次运行确认脚本失败的问题点。

(a)环境问题导致的失败,将环境问题导致失败的理由写在分析结论中。

(b)脚本问题导致的失败,测试工程师修改脚本后将脚本重新提交到 Git 的脚本库中。

(c)产品缺陷问题导致的脚本失败,测试工程师将脚本失败问题提交给软件开发工程师进行定位,如果开发工程师确认是由产品缺陷导致的失败,测试工程师提交问题单,在 DAC 工具的对应失败脚本中补充

问题单号。

(d)软件版本包问题导致的失败,联系持续集成工程师与软件开发工程师共同处理该问题。

(e)概率性失败的脚本通常表现为有时测试成功,有时测试成功。通常是测试脚本中存在某种隐藏的问题,在一定的外部条件下触发,导致的测试脚本失败。测试工程师通过在脚本失败的执行机上多次连续运行,查找问题的根源。最后将处理结果反馈给自动化工厂测试工程师,并继续对该脚本持续跟踪,分析错误根因。

各责任田主将脚本失败的原因,分析结论填写到 DAC 中失败脚本的分析结论中,反馈给自动化工厂测试工程师。每个责任田主需要在规定的时间内闭环脚本失败分析。

5.3.5 处理存在的问题

根据测试脚本失败的原因,相关人员处理各种测试脚本失败问题。

5.3.6 提交脚本失败分析报告

将脚本失败的原因,定位分析过程,解决问题方案,处理结果写入脚本失败分析报告,提交 DAC 备案,为以后的测试工作提供借鉴和帮助案例。

6 测试工作的改进

6.1 自动化工厂的工作改进

自动化脚本连跑与失败分析结束之后,自动化工厂测试工程师对各责任田主反馈的脚本失败分析报告,进行分析汇总。改进自动化工厂工作,提升自动化脚本连跑成功率。对于环境问题导致的失败,自动化工厂需要重点关注以下问题:

(a)关注脚本对应的环境组网类型标签,修改自动化脚本给出的环境组网类型错误。

(b)测试执行机由于网络问题导致脚本失败,需要确认执行机与测试环境上业务网络是否相通。如果网络不通,先排查实验室网络环境是否正常,仅是单个执行机出现网络不通的情况,可以通过在执行机上添加路由的方式使执行机上的网络与测试环境的业务网络互通。

(c)对于脚本配置残留问题,自动化工厂测试工程师可以在每个脚本运行之前加上配置残留检测,发现有配置残留的脚本之后,知会责任田主或分析田主确认和处理配置残留问题。

(d)对于概率性失败的自动化测试脚本,自动化工厂的测试工程师与脚本责任田主在后续的脚本连跑中持续关注脚本的运行情况,对能够复现的失败及时定位脚本运行失败的原因。

6.2 测试组提升测试效率

每一轮迭代完成之后,测试经理与测试架构师对自动化脚本连跑发现的问题进行汇总分析。对于问题较多的产品特性,测试架构师需要和该特性责任田主,一起分析脚本失败较多,问题较多的原因。责任田主完成质量补充测试和脚本优化重构整改。对于新开发特性及时补充测试用例,测试工程师测编写脚本,脚本评审通过之后完成脚本入厂。

6.2.1 测试环境中添加业务配置提升测试效率

软件产品 M 进行自动化连跑测试时,出现大量的 K 特性测试脚本失败。K 特性责任田进行失败分析后,发现测试环境需要增加部分业务配置。自动化工厂测试工程师在测试环境中添加了相关的业务配置后,K 特性的自动化测试脚本连跑成功率显著提升。

6.2.2 通过问题单分析提升测试效率

测试经理与测试架构师对 M 产品的所有问题单进行汇总分析,发现这一段时间的问题单中,有 55 张问题单来自自动化脚本失败分析。进一步深入分析发现,软件新特性的自动化脚本连跑出现的问题,多来自于软件特性交互。测试架构师为特性交互部分补充了测试用例。测试工程师为补充的测试用例编写测试脚本。再次进行测试脚本连跑,软件交互部分通过了测试脚本连跑,从而提高了 M 产品的自动化测试效率。

7 典型案列

在软件产品的迭代过程中,自动化测试的应用场景比较多,在案例中仅提供软件迭代开发完成之后,对软件进行验收前,执行测试脚本连跑的统计数据。

7.1 软件产品自动化测试脚本连跑结果

W 产品与 S 产品在迭代开发结束之后,自动化测试脚本连跑结果如表 2 所示。

表 2 自动化测试脚本连跑结果

产品名称	连跑测试脚本总数	运行通过测试脚本数	运行失败测试脚本数	运行时长	连跑涉及的执行机数目
W 产品	9 350	9 241	109	11h21m58s	24
S 产品	12 741	12 672	69	16h29m23s	16

从表中可以看出,S 产品的自动化脚本数量比 W 产品多,失败脚本数量却比较少。由此可见 S 产品的自动化测试脚本与脚本执行机环境稳定性更高。

7.2　测试脚本失败的分类统计

对 S 产品与 W 产品的失败脚本进行分类统计,分类统计数据如表 3 所示。

表 3　测试脚本失败的分类统计

产品名称	失败测试脚本总数	环境问题导致失败	脚本问题导致失败	测试脚本概率性失败	软件缺陷问题导致失败
W 产品	109	25	47	10	27
S 产品	69	13	24	12	20

注:对于软件验收的自动化测试通常不存在版本包问题。

从表 3 中的数据,分析结论如下:

(a)自动化脚本测试环境需要加强配置管理工作,提升自动化脚本连跑的成功率。

(b)对于脚本问题,责任田主需要及时将修改后的脚本合入脚本库。

(c)对于概率性失败的脚本,需要责任田主和自动化工厂测试工程师跟踪这些失败问题,逐步发现概率性失败的原因,解决脚本失败问题。

(d)对于软件缺陷问题,测试工程师均已提单跟踪,由开发工程师处理软件缺陷问题。

8　结束语

软件自动化测试在软件开发过程中获得广泛应用,自动化测试通过测试脚本连跑来实现。自动化脚本连跑提升了软件测试效率,为软件迭代开发和测试提供了良好的基础。自动化测试脚本连跑过程中必然有某些测试脚本失败,需要对这些测试脚本进行失败分析。对于软件自动化测试而言,测试脚本的失败分析是其中重要的核心环节。长期的工作实践表明做好软件自动化测试用例失败分析,有助于提高软件产品迭代开发的效率和提升软件产品的质量。

参考文献:

[1] STEPHENS R. 软件工程入门经典[M]. 明道洋,曾庆红,译. 北京:清华大学出版社,2016.

[2] 肖利琼. 软件测试之魂核心测试设计精解[M]. 第 2 版. 北京:电子工业出版社,2013.

[3] LARMAN C. 敏捷迭代开发:管理者指南[M]. 北京:人民邮电出版社,2013.

[4] BITTNER K,SPENCE L. 迭代软件开发项目管理[M]. 罗景文,译. 北京:清华大学出版社,2010.

[5] HODA R,NOBLE J,MARSHALL S. Self-organizing roles on agile software development teams[J]. IEEE Transactions on Software Engineering,2013,39(3):422-444.

[6] KETTUNEN P,MAARIT L. Combining,agile software projects and large-scale organizational agility[J]. Software Process Improvement and Practice,2013,13(2):183-193.

[7] DINGSØYR T,NERUR S,BALIJEPALLY V G,et al. A decade of agile methodologies:towards explaining agile software development[J]. Journal of Systems & Software,2012,85(6):1213-1221.

[8] 吴　俊. 敏捷测试在 S 银行软件项目中的应用研究[D]. 上海:东华大学,2017.

[9] CARRERA A,IGLESIAS C A,GARIJO M. Beast methodology:an agile testing methodology for multi-agent systems based on behaviour driven development[J]. Information Systems Frontiers,2013,16(2):169-182.

[10] 姜　文,刘立康. 软件调试问题研究[J]. 计算机技术与发展,2017,27(11):1-6.

[11] 姜　文,刘立康. 基于持续集成的冒烟测试[J]. 计算机技术与发展,2018,28(8):53-57.

[12] 姜　文,刘立康. 基于持续集成的 C/C++软件覆盖率测试[J]. 计算机技术与发展,2018,28(3):37-41.

[13] 姜　文,刘立康. 应用软件项目的迭代开发与测试[J]. 计算机技术与发展,2019,29(4):7-12.

[14] 姜　文,刘立康. 基于 Selenium 的 Web 软件自动化测试[J]. 计算机技术与发展,2018,28(9):47-52.

[15] 王平升. 软件测试错误报告分析处理工具的设计与实现[D]. 西安:西安电子科技大学,2017.