南京大學

研 究 生 毕 业 论 文

（申请工学硕士学位）

论 文 题 目 基于动静态程序分析结合的移动众包测试引导技术

作 者 姓 名 张欣

学科、专业名称 工学硕士（软件工程领域）

研 究 方 向 软件工程

指 导 教 师 陈振宇 教授

2019 年 5 月 27 日

学　　号：**MG1632011**

论文答辩日期：**2019 年 5 月 11 日**

指 导 教 师：　　（签字）

Fusing Dynamic and Static Program Analysis for Guiding Mobile Crowdsourced Testing

By

Xin Zhang

Supervised by

Professor **Zhenyu Chen**

A Thesis
Submitted to the Software Institute
and the Graduate School
of Nanjing University
in Partial Fulfillment of the Requirements
for the Degree of
Master of Engineering

Software Institute

May 2019

南京大学研究生毕业论文中文摘要首页用纸

毕业论文题目：基于动静态程序分析结合的移动众包测试引导技术

工学硕士（软件工程领域） 专业 2016 级硕士生姓名： 张欣

指导教师（姓名、职称）： 陈振宇 教授

摘 要

安卓移动应用以快速迭代开发模式占据大部分市场。安卓系统碎片化和应用场景多样化给移动应用质量保障提出新挑战。自动化测试和众包测试成为解决移动应用质量困境的两种有效互补手段。但当前众包测试依然存在流程无监管和专业素质参差不齐等问题，需要引入引导技术进一步提高效率。

本文提出一种基于动静态程序分析结合的移动众包测试引导技术。动态分析选用慕测平台自动化测试框架，基于程序执行日志和组件遍历信息，提取自动化测试过程中触发异常的场景，引导众包工人在不同设备和不同环境下复现异常。采用Android GUI静态分析工具GATOR，在不执行程序的情况下，遍历并分析上下文相关的程序控制流路径。该技术从代码端得到GUI状态的变化情况，提取自动化测试未覆盖的窗口跳转部分，引导众包工人探索新异常。跟踪测试任务完成情况，个性化推荐测试任务。推荐系统根据测试任务的不同类型，基于前置事件序列还原当前窗口下的异常触发路径、基于最短路径计算当前窗口下的未覆盖窗口跳转路径，使用截图和文本相结合的提示信息，引导众包工人以一次窗口跳转为单位，执行操作事件序列，引导快速完成测试任务。同时，测试任务推荐系统引导众包工人之间形成协作式关系，过滤已被多次完成的测试任务，均衡整体测试效果。

本文在“简豆”、“记乐部”和“QQ影音”三个常用Android应用上开展实验验证。实验包括无引导众包测试、动静态程序分析引导下的众包测试以及只包含自动化测试引导的众包测试。实验结果表明在相同时间内，动静态程序分析引导下的众包测试能够比无引导众包测试平均多触发约30%的异常，其中未充分验证的异常降低50%多。该技术对比于自动化测试引导的众包测试，能够提升代码覆盖率超过20%。结果表明引入Android GUI静态分析的必要性。面向众包工人的问卷调查结果显示，众包工人对该系统的满意度高达86%。

关键词： 众包测试，自动化测试，静态分析，推荐系统，测试引导

南京大学研究生毕业论文英文摘要首页用纸

THESIS: Fusing Dynamic and Static Program Analysis for Guiding Mobile Crowdsourced Testing

SPECIALIZATION: Software Engineering

POSTGRADUATE: Xin Zhang

MENTOR: Professor Zhenyu Chen

Abstract

Android mobile applications occupy most of the market with the rapid and iterative development mode. The fragmentation of Android system and the diversity of application scenarios pose new challenges to the quality of mobile applications. Automated testing and crowdsourced testing are two effective complementary approaches to solve the dilemma of mobile applications' quality. However, the current crowdsourced testing still has some problems, like unsupervised process and uneven professional quality, so it is necessary to introduce guiding technology to further improve efficiency.

This paper proposes a technology of guiding mobile crowdsourced testing based on fusing dynamic and static program analysis. Automated testing framework from MoocTest is used as dynamic analysis. Based on runtime logs and component traversal, application scenarios triggering exceptions are extracted to guide crowd workers to recurrence exceptions on different devices and in various circumstances. GATOR, an Android GUI static analysis tool, is selected to analyze the GUI changes from the aspects of context-dependent control flow path in source code, without running applications. Static analysis gets GUI changes from source code. Window transactions which are uncovered by automated testing can be extracted by comparison to guide workers to explore new exceptions. This technology tracks the completion of test tasks, and implements personalized recommendations for test tasks. The recommendation system use different approaches to calculate paths from current window according to different types of test tasks. Paths triggering exceptions are calculated based on pre events' sequence, while uncovered window transactions use the shortest path calculation. Using information combined with the screenshots and text, workers are guided to execute the

sequence of operation events in a single window transaction, quickly completing test tasks. At the same time, the test task recommendation system guides workers to form a collaborative relationship, filtering the test tasks that have been completed for several times and balancing the overall test effect.

This paper has carried out experiments on three Android applications, namely ”jiandou”, ”jilebu” and ”QQ vedio”. The experiments include unguided crowdsourced testing, dynamic and static program analysis guided crowdsourced testing, and crowdsourced testing only with guidance from automated testing. Results show that the crowdsourced testing under the guidance from the dynamic and static program analysis can trigger an average of about 30% more exceptions than the unguided crowdsourced testing, among which the unverified exceptions are reduced by 50% more. The technique proposed in this paper improves code coverage by more than 20% compared to crowdsourced testing only with guidance from automated testing. The results show the necessity of introducing static analysis of Android GUI. A survey of crowd workers shows that 86% of them are satisfied with the recommendation system.

Keywords: Crowdsourced testing, Automated testing, Static analysis, Recommendation system, Test guidance

目　录

表 目 录 ····· **ix**

图 目 录 ····· **xii**

第一章　引言 ····· **1**

1.1　项目背景 ····· 1

1.2　研究现状 ····· 1

1.2.1　移动应用测试 ····· 1

1.2.2　众包测试 ····· 3

1.3　论文组织与主要贡献 ····· 5

第二章　相关工作 ····· **7**

2.1　移动应用测试 ····· 7

2.1.1　自动化测试 ····· 7

2.1.2　安卓静态分析 ····· 8

2.1.3　众包测试 ····· 9

2.2　推荐系统 ····· 10

2.2.1　协同过滤推荐 ····· 11

2.2.2　冷启动 ····· 13

2.3　本章小结 ····· 14

第三章　众包测试引导技术 ····· **15**

3.1　基于GUI模型的测试 ····· 16

3.1.1　Android相关特性 ····· 17

3.1.2　普通窗口跳转 ····· 18

3.1.3　测试用例 ····· 21

3.1.4　未覆盖窗口跳转 ····· 22

3.2　测试任务推荐 ····· 26

3.2.1 冷启动 ····· 27
3.2.2 基于物品的协同过滤推荐 ····· 29
3.2.3 协作式测试任务分派 ····· 31
3.3 测试实时引导 ····· 31
3.3.1 跳转路径 ····· 32
3.3.2 实时引导 ····· 35
3.4 本章小结 ····· 35

第四章 实验设计与分析 ····· 37
4.1 实验目的 ····· 37
4.2 实验设置 ····· 37
4.2.1 实验应用选择 ····· 37
4.2.2 实验机型统计 ····· 38
4.2.3 实验步骤 ····· 38
4.3 实验结果预处理 ····· 41
4.3.1 异常数据统计 ····· 41
4.3.2 异常数据分析 ····· 41
4.4 实验结果分析 ····· 44
4.4.1 测试质量评估 ····· 44
4.4.2 代码覆盖率统计 ····· 47
4.4.3 推荐系统满意度 ····· 50
4.5 有效性威胁 ····· 51
4.6 本章小结 ····· 51

第五章 总结与展望 ····· 53
5.1 总结 ····· 53
5.2 展望 ····· 54

参考文献 ····· 55

简历与科研成果 ····· 61

致谢 ····· 63

版权及论文原创性说明 …………………………………………………… 65

表 目 录

3.1 自动化测试事件输出信息 ····· 19
3.2 GUI模型中事件的具体存储结构 ····· 20
3.3 Android应用程序常见异常类型及严重程度 ····· 28
3.4 事件类型及权重对应表 ····· 29
3.5 “简豆”部分测试用例相似度 ····· 30

4.1 实验设备情况统计 ····· 38
4.2 不重复异常平均统计数据 ····· 41
4.3 实验异常分布情况 ····· 42
4.4 未覆盖窗口比例统计 ····· 48
4.5 应用程序“简豆”未覆盖窗口的代码覆盖率 ····· 48
4.6 应用程序“记乐部”未覆盖窗口的代码覆盖率 ····· 48
4.7 应用程序“QQ影音”未覆盖窗口的代码覆盖率 ····· 48

图　目　录

1.1 众包测试流程 …… 3

3.1 众包引导技术的设计 …… 15
3.2 基于动静态程序分析结合的移动众包测试引导技术框架 …… 16
3.3 Android应用程序GUI模型构建流程 …… 17
3.4 普通窗口跳转处理流程图 …… 18
3.5 操作事件输出格式 …… 19
3.6 不同测试事件对应的程序运行截图 …… 21
3.7 自动化测试中程序日志输出格式 …… 22
3.8 测试用例处理流程 …… 23
3.9 WTG事件输出格式 …… 24
3.10 未覆盖窗口跳转的提取流程 …… 24
3.11 应用程序“简豆”的部分GUI模型 …… 25
3.12 测试任务推荐系统流程 …… 26
3.13 Android端的实时引导流程 …… 32
3.14 跳转路径计算流程 …… 33
3.15 应用程序“简豆”中弹出的豆瓣账号登录窗口 …… 34
3.16 程序运行截图 …… 36

4.1 集成该技术的应用程序“简豆” …… 38
4.2 测试用例类型统计情况 …… 39
4.3 应用程序“记乐部”部分GUI模型 …… 40
4.4 应用程序“QQ影音”部分GUI模型 …… 40
4.5 “简豆”在三个实验中类型异常分布变化图 …… 42
4.6 “记乐部”在三个实验中类型异常分布变化图 …… 43
4.7 “QQ影音”在三个实验中类型异常分布变化图 …… 43
4.8 “简豆”异常数目随时间变化图 …… 44
4.9 “记乐部”异常数目随时间变化图 …… 45

4.10 “QQ影音”异常数目随时间变化图 ····· 45
4.11 应用程序“简豆”的图书列表加载失败 ····· 46
4.12 实验异常复现情况统计图 ····· 47
4.13 应用程序“QQ影音”MainActivity对应的窗口 ····· 49
4.14 推荐系统满意度 ····· 50

第一章　引言

1.1　项目背景

随着4G时代的开启和移动设备的突显，移动互联网产业飞速发展，其便捷化及个性化特征使得移动互联网赶超传统互联网行业，成为互联网流量的最大入口。智能设备操作系统是移动互联网发展的核心，Google公司的Android操作系统以多于80%的占比，超过苹果的iOS系统和微软的Windows Phone系统，在智能设备操作系统市场中占据领导地位。Android系统自2008年9月发布第一版，逐步迭代，目前Android系统的用户使用版本主要集中于Android 4.4到Android 9之间。Android系统广泛应用的背后，是智能设备和安卓应用的发展。智能设备作为移动互联网发展的依托，2018年基于Android操作系统的智能手机出货量超过10亿，占据智能手机市场的85%[1]。据统计显示，Android应用管理中心Google Play，其应用商店下载量超过iOS应用商店的两倍[2]，目前在架Android应用覆盖娱乐、社交、阅读、交通等多个类别，基于Android系统的移动互联网逐步渗透生活的每一方面，并改变着用户工作、学习和生活的方式。

Android市场的发展也带来诸多问题。快速迭代的开发模式、井喷式的应用数量、复杂的使用场景、严重的碎片化问题……这些因素都向移动应用的质量提出挑战，Android应用健壮性亟待提高。

在软件开发过程中，软件测试是保证软件质量的必要环节，软件开发者希望从测试过程中快速获得产品反馈，修复软件异常，提高产品质量。和传统Web程序相比，Android应用程序在交互方式、运行环境、运行场景等方面，存在很大不同，传统软件测试不再能完全适用于移动应用。针对移动应用的测试研究在工业界和学术界逐步发展起来。

1.2　研究现状

1.2.1　移动应用测试

和所有应用一样，移动应用也需要经过充分的测试，才能保证程序运行的健壮性和安全性。随着移动业务增长，移动应用自动化测试技术的需求也在逐步增长。作为拥有移动市场最大份额的Android系统，其设备和版本的分布

[1] www.idc.com/cn/

[2] www.appannie.com/cn/

性、Android平台及其相关技术的开源性 [1]，吸引众多研究人员尝试与研究针对Android应用程序的自动化测试。

Android应用程序是基于事件驱动的GUI程序，主要由Java语言编写完成 [2]，但Android应用程序与纯Java语言编写的GUI程序存在很大不同：一方面是Android 应用程序的交互方式和应用场景变化，另一方面，Android应用程序架构发生变化，程序的运行过程中会产生Android系统特有异常。因此，基于Java的自动化测试工具不能直接用于测试Android应用程序。根据Android应用程序事件驱动的特征，研究人员提出基于GUI的自动化测试方案，通过模拟用户交互事件和系统事件，遍历安卓页面组件，产生测试输入，检测应用程序中存在的异常 [3]。

从安卓页面组件的遍历策略区分，安卓自动化测试可以分为以下三类：

1. 基于随机策略的自动化测试。顾名思义，这一类自动化测试通过点击、输入等事件向系统发送伪随机的用户事件流，随机策略只生成用户交互事件，很难生成系统事件。

2. 基于模型的自动化测试。通过构建应用程序的GUI模型，生成系统事件与用户交互事件，探索应用程序的状态转换。

3. 系统探索式的自动化测试。这一类的自动化测试通过一些复杂的测试技术，探索特定输入情况下的程序运行情况。

随着移动应用自动化测试的发展，该技术已经逐步运用于工业生产中。云测服务平台就是在移动应用自动化测试的基础上发展起来的，如AllTesting[3]，阿里MQC[4] 等。通过在云端部署并开放千万台测试终端，开发人员提交待测Android应用并选择测试环境，云测服务平台便可以在无人工干预的情况下，进行在线自动化测试，输出测试报告 [4]。云测平台中，自动化测试直接决定云测服务平台的测试质量。

云测服务平台一定程度上提高Android应用测试的充分性，但其与自动化测试的高耦合，也导致测试结果存在一定局限性 [5]。以众包工人代替云测平台，以自动化测试在单个终端的测试结果引导众包工人在不同环境下复现异常，同时利用众包工人在测试过程中的主动探索与反馈，在移动应用自动化测试优化众包测试的基础上，众包测试也完善自动化测试结果，两者优势互补，实现更加全面、高效、高质的移动应用测试。

[3]www.alltesting.cn

[4]mqc.aliyun.com/

1.2.2　众包测试

与传统Web应用相比，Android应用程序架构更加复杂、碎片化问题日趋严峻，而产品迭代与更新周期却越发紧凑，程序测试周期被急剧压缩。如何在短时间内获得大量产品反馈、特别是来自不同运行环境下的真实使用反馈？这是当前Android应用程序测试亟需解决的问题。

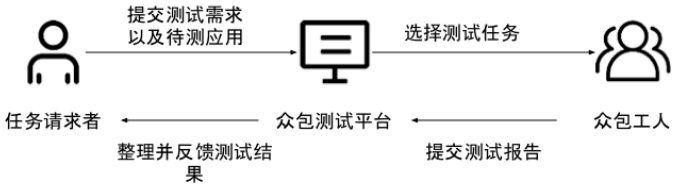

图 1.1: 众包测试流程

众包 [6]被提出解决移动应用测试的困境。众包，即一个公司或机构把过去由员工执行的工作任务，以自由自愿的形式外包给非特定的（而且通常是大型的）大众志愿者的做法 [7]。众包从用户真实使用感受出发，基于互联网完成产品调研。目前众包模式已经逐步应用于软件工程领域，尤其是软件测试领域。如图1.1所示，任务请求者（Requester）向众包测试平台提交测试需求，众包工人（Crowd worker）选择测试任务并向众包测试平台提交测试报告，众包平台整合测试结果并反馈给任务请求者 [8]。通过大量来自众包工人的真实产品使用反馈，众包测试以低成本、高回报的优点引起学术界与工业界的广泛关注，目前众包测试的研究覆盖GUI测试、QoE测试、性能测试、测试用例生成等领域，一大批众包测试平台也应运而生，比如MoocTest[5]，Testin[6]，百度众测[7]，BugTags[8]，uTest[9]，Pay4Bugs[10]。

但众包测试其模式本身存在着一些固有的问题：

(1)众包工人专业素养较低。由于众包测试平台对众包工人的招募是无门槛的，众包工人不具有专业的软件测试知识，与软件测试工程师相比，单个众包工人的测试效率低下，测试回报率低。

[5]www.mooctest.net
[6]www.ztestin.com
[7]test.baidu.com
[8]www.bugtags.com
[9]www.bugtags.com
[10]www.bugtags.com

(2)众包测试过程缺少管理。众包的奖励机制引导众包工人之间形成竞争关系而非协作关系，重复测试报告出现频率高，测试效率低下。

(3)测试报告整合困难。众包测试报告质量参差不齐，众包平台对测试报告的后期整合难度大、工作量大。

由此可见，大量优质众包工人的召集是高质量众包测试的基础 [9]，而提供对众包工人的引导可以在测试过程中完成测试培训，提高工人专业素养，缓解上述问题。对于没有测试经验的众包工人，测试过程中的引导与提示不仅可以帮助众包工人迅速熟悉测试流程、了解测试相关知识，而且通过合理规划测试任务的分派与调度策略，可以实现协作式众包测试流程，优化众包资源配置，提高整体测试效率。

因此，探索一种高效的众包工人引导机制，对于提升移动应用众包测试的质量具有重要意义。

1956年，人工智能的概念首次确立，以计算机模拟人类思考与认知 [10]。近年来人工智能逐步从学术界走向产业界，在国内外掀起发展浪潮，逐步覆盖家居、交通、医疗、零售、教育等方面。但随着技术的演进，传统人工智能面临新的挑战：

（1）社会需求的变更。传统人工智能的核心在于模拟人的智能行为，而无人驾驶、智能城市等概念的提出，使得人工智能的研究对象转向具体的运行情况与状态变更。

（2）信息环境的变更。人工智能以数据为驱动，在演进过程中带来数据形式的变化。大数据、多媒体数据、虚拟现实等多种多样的数据形式丰富了传统人工智能的信息环境。

（3）混合智能的发展。人工智能固然优秀，但并不能完全取代人类智能，将人类的学习推理能力与机器智能相结合，互相补充，可以构建一个更加强大的智能系统。

2017年，中国工程院发布人工智能2.0专题 [11]，“群体智能” [12]作为五大发展方向之一，以有效的方式协调人类智能与机器智能，在融合的基础上取长补短，拓展和深化人工智能。

受群体智能思想和众包改进思路的启发，本文提出一种基于动静态程序分析结合的移动众包测试引导技术，将机器智能与人类智能相结合，即将动静态程序分析与众包测试相结合，以程序分析的结果结果引导众包测试，以众包测试补充自动化测试结果，改进众包流程的组织流程。将两者结合，本文提出众包测试机制的改善与优化方案，实现更加充分、更高效率的移动众包测试。

1.3　论文组织与主要贡献

本文受群体智能思想的启发，结合众包测试机制存在的问题，提出一种基于动静态程序分析结合的移动应用众包测试引导技术，以自动化测试执行程序动态分析，对比程序日志和测试事件序列，还原异常被触发的场景，引导众包工人在不同设备、不同环境以及不同场景下复现异常。同时，在不执行程序的情况下，根据上下文相关的程序控制流路径，静态分析程序源码，得到GUI状态的变化情况，提取自动化测试未覆盖的窗口跳转，引导众包工人探索异常。该技术跟踪众包工人的测试任务完成情况，实现个性化测试任务推荐系统。推荐系统根据测试任务的不同类型，采用不同的方式计算窗口跳转路径。该技术基于前置事件序列还原当前窗口下的异常触发路径，而对于未覆盖窗口跳转，则基于最短路径的思想计算跳转路径。以截图和文本相结合的方式，该技术引导众包工人以一次窗口跳转为单位，执行操作事件序列，引导快速完成测试任务。同时，测试任务推荐系统记录测试任务的完成情况，过滤已被多次完成的测试任务，引导众包工人之间形成协作式关系，均衡整体测试效果。

为验证机制的有效性，我们选取并分析三个Android应用程序，开展无引导下的众包测试、程序分析引导下的众包测试以及只包含自动化测试引导的众包测试。我们分析对和对比实验结果，从测试效率和已确认异常数验证程序分析引导下的众包测试能够有效提升测试质量。从代码覆盖率对比程序分析下的众包测试和只包含自动化测试引导的众包测试，实验结果证明Android GUI静态分析的必要性。我们在众包工人中开展关于测试任务推荐系统的满意度调查，结果显示，86%的众包工人认可该技术中的测试任务推荐系统。

总结相关工作，本文的贡献主要有：

1. 提出一种众包测试流程的优化机制，并首次在众包测试的过程中，引入众包工人引导机制。

2. 设立两组对比实验，分别从测试效率、已验证异常和代码覆盖率的角度，利用真实数据评估该技术的有效性。

3. 开展问卷调查，了解众包工人对测试任务推荐系统的满意度。

本文的组织结构如下：

第一章是论文的引言部分，介绍该项目研究的背景，就当前众包测试平台的技术现状，提出引导性众包测试机制。

第二章是论文的相关技术部分，介绍框架相关的研究，包括众包测试，移动应用自动化测试、Android GUI静态分析以及推荐系统。

第三章详细介绍该技术的具体设计，从基于GUI模型的测试任务、测试任

务推荐机制以及实时引导机制三个模块详细介绍机制的整体流程。

第四章是对该技术的评估实验，通过设定实验目的，利用三个不同应用程序分别完成对比实验，并分析实验相关结果。

第五部分是总结和展望，总结在工作期间的经验，提出不足与改进之处。

第二章　相关工作

2.1　移动应用测试

2.1.1　自动化测试

基于Android应用程序事件驱动的特征，移动应用自动化测试面向GUI（图形用户界面），模拟用户交互事件和系统事件，探索应用程序状态，产生测试用例，检测应用程序中存在的异常 [13]。

从探索应用程序行为的策略区分，面向Android系统的移动自动化测试可以分为以下三类：

（1）基于随机策略的自动化测试随机生成UI事件。Monkey [14]作为目前最为常用的移动应用自动化随机测试工具，内部集成于Android开发工具包中。通过测试人员指定期望的测试事件数量，Monkey以黑盒测试实现基本的用户交互事件。Machiry等人 [13]提出的移动应用自动化测试框架Dynodroid也是基于随机策略，但与Monkey相比，Dynodroid从以下几个角度进行改进：一是Dynodroid在Android框架中注册监听器，也能够生成与应用程序相关的系统事件，二是与Monkey的完全随机相比，Dynodroid的随机策略更加优秀，更倾向于选择存在上下文关联的事件，三是Dynodroid允许测试人员提供测试用例，产生能够触发应用程序崩溃的测试用例，测试程序的健壮性。

（2）基于模型的自动化测试工具根据Android程序窗口变化，动态构建应用程序事件与状态之间的GUI变更模型。Amalfitano等人 [15]提出以深度优先遍历策略，探索程序运行的不同状态，完成自动化遍历框架GUIRipper，该框架允许在人工辅助的情况下，从不同启动状态探索应用程序GUI变更。Choi等人 [16]提出的SwiftHand 框架通过优化Android页面组件的探索策略，动态生成Android应用程序的状态模型，希望能够最大限度地扩大应用程序状态的覆盖范围。Hao等人 [17]则实现一个通用的UI自动化框架PUMA，其框架具有一定的扩展性，允许测试人员基于不同的探索策略，动态生成应用程序的状态模型。慕测平台[1]的自动化测试框架以人工测试脚本为辅助，在遍历过程中遇到输入控件时，主动寻找人工测试脚本中对应的输入参数，触发相应的状态变更，构建窗口跳转模型。通过调整人工脚本参数，慕测平台自动化测试框架力求更加全面和充分的测试结果。

[1]www.mooctest.net

（3）系统探索式的移动应用自动化测试。一些程序行为只有在特定输入下才能显示，因此，一些Android框架使用更加复杂的技术，如符号执行和进化算法等技术，关注一般探索策略未覆盖部分。Mahmood等人 [18]提出的EvoDroid框架，依赖进化算法生成测试用例，通过适应度函数实现覆盖范围的最大化。Anand等人 [19] 则基于符号测试，提出自动化测试框架ACTEve，跟踪应用程序在代码层的方法调用情况。

Choudhary等人 [20]选用10个Android应用程序，分别从框架易用性、系统兼容性、代码覆盖率以及故障检测能力这四个指标，评估和分析当前主流的一些移动应用自动化测试框架。研究发现即使是覆盖率较低的工具，在应用程序上的平均覆盖率也可以达到80%。Jiang等人 [21]则分析33个真实应用的异常，系统地研究影响基于GUI遍历的测试用例生成结果的因素。研究结果显示，一是对于GUI状态的定义以及判断两个GUI状态是否相等的指标影响错误检测率和代码覆盖率，二是受探索策略的影响。

从程序覆盖率的角度，对比总结上述针对Android应用的移动应用自动化测试，可以发现目前主流的自动化测试框架能够稳定、较高覆盖率地检验程序异常。因此，分析自动化测试过程，根据自动化测试触发的异常以及对应的窗口跳转情况，引导众包工人在不同环境下复现异常的思路是可行的。不仅如此，与传统众包测试中平台管理员人工划分测试任务、文本描述测试需求的模式相比，这样的测试任务设计机制具有更高的可信度与专业水平，能够更好地在测试过程中培训众包工人。

2.1.2　安卓静态分析

随着Android框架的不断演变，基于Android应用程序源码的GUI静态分析也在不断发展。与面向Web端的应用程序不同，Android应用程序依赖于Android Framework设计的交互模式，采用完全不同的项目框架搭建系统，因此传统的程序静态分析不能直接用于Android应用程序。

为了实现对Android应用程序的源码分析，首先需要了解Android应用程序的运行机制。Android应用程序响应来自用户的交互事件，交互事件作用对象以及事件的处理机制决定程序代码层的数据流与控制流。与移动应用自动化测试不同的是，Android程序的静态分析无需运行应用程序，通过理解代码程序中的语法、函数、接口等信息，为Android对象、对象间的交互以及对象与Android平台之间的交互建立模型。目前，针对Android应用程序源码分析的工作由于难度较大，相关研究并不是很多，在产业界也没有得到大规模的应用，但是这项工作的开展具有重要意义，高效的Android源码分析能够实现测试用例自动的生

成，优化真实应用场景中的测试方案。

Yang [22]等人提出静态分析框架GATOR[2]，构建窗口转换图WTG，图中节点代表安卓窗口，边记录窗口之间的转换信息，描述Android GUI控制流变化。Arzt 等人 [23]从程序安全的角度，提出用于Android应用程序的静态污点分析框架FlowDroid，通过程序上下文、流敏感、字段敏感和对象敏感分析，分析运行生命周期内的回调方法和处理GUI事件的程序回调方法，在[38]工作的基础上，派生、构建回调序列图（CSG）。Li等人 [24]则在FlowDroid的基础上进行了一些改进，将FlowDroid与Android应用程序组件之间的隐私泄露相结合，追踪从源头到接收器的任何敏感数据流，构建一个更加精确的静态分析框架IccTA。

Wang等人 [25]总结当前Android静态分析的工作，使用六个开源Android应用程序评估当前主流的静态分析工具FlowDroid，IccTA和GATOR，从回调函数覆盖率比较这三个Android静态分析框架的解决方案，同时Wang等人也分别分析了这三个静态分析框架目前没有达到100%覆盖率的原因。

2.1.3　众包测试

2006年，Jeff等人首次提出众包的商业概念 [26]：公司或者机构通过互联网，公开地将由员工完成的工作任务，以自由资源的方式外包给非特定的群体，共同完成分布式的问题求解。

Mao [27]等人全面总结与概述众包技术在软件工程中的应用，众包的思想已经在软件开发、移动应用开发、软件测试、移动应用测试、软件安全等软件工程领域进行过诸多尝试。2009 年，Chen等人 [28]创新性地提出将众包技术应用于QoE（quality of experience）测试中，关于将众包技术与软件测试相结合，如何使用众包技术提升测试效率与质量的问题，逐步成为软件测试的研究热点之一。Liu等人 [29]调研众包测试与可用性测试的结合，实验结果发现众包可用性测试可以显著降低测试成本，但众包可用性的质量略低于传统可用性测试的质量。Komarov等人 [30]从结果差异性分布、任务完成时间、错误率和一致性等方面对比众包GUI 测试与传统实验室测试，证明众包GUI测试的可行性，即众包技术能够实现有效开展GUI测试。众包与测试相结合的模式不再停留于学术界的研究，相关研究成果也逐步得到大规模的应用，工业界逐步采用众包的机制完成测试方案。Chrome公司以内置遥感装置的方式 [31]，收集来自众包工人的真实使用数据，取代模拟器测试结果，以更加多样化、更具代表性的方式反馈应用程序的性能问题。将众包测试与性能测试相结合，微软、Firefox 等公司已完成相应产品的研发 [32] [33]，产品也同样得到一定程度的应用。总体来

[2]web.cse.ohio-state.edu/presto/software/gator/

说，目前众包思想已经逐步应用于软件测试的QoE测试、可用性测试、GUI测试、性能测试与测试用例生成等多方面。

针对移动应用的众包测试也随着移动时代的爆发而受到关注，终端用户在使用移动智能设备的过程中，完成软件相关测试。Khan [34]在众包可用性的基础上，研究针对移动应用的众包可用性测试，调研结果发现，针对移动应用的众包可用性测试主要关注移动应用的功能与可用性，其测试结果的质量与众包工人的实际使用场景密切相关。Maria [35]从众包工人的实际使用过程中收集应用程序数据，识别异常信息，复现异常产生的上下文场景，帮助开发人员定位异常产生原因。

基于众包在软件测试中的优秀表现，越来越多的学者开始关注众包测试的改进与优化。Mantyla等人调查众包工人规模和时间约束对测试结果的影响，实验结果表明，众包人员的规模由测试结果的重复率与重复处理机制的效率决定，同时，受时间约束的众包工人能够更好地检验程序异常。王俊杰等人 [36]根据众包测试报告中的文本，提出一种基于聚类的众包测试报告分类方法。Yang等人 [37]则创新性地从测试报告的截图出发，利用图片分析技术分析截图、获取测试信息，辅助测试报告的优先级排序。而郝蕊等人 [38]将文本描述与截图相融合，利用文本技术分析测试报告特征、图片分析技术处理截图特征，提出一种基于特征融合的测试报告处理框架。Quoc等人 [39]则提出一种新的测试报告处理工具ERICA，该工具利用专家反馈，简化测试报告的验证过程，以专家预期的测试报告答案集，引导测试报告的评估，排除不符合标准的报告，提高后期的处理质量。面对众包测试相关技术的广泛应用与发展，章晓芳等人 [9]收集52篇与众包测试相关的文献，概述当前众包软件测试的技术发展情况，根据不同众包测试平台的实际使用情况，提出从众包工人、测试任务、报告处理与软件评估这四个角度，优化众包测试的流程。

由此看来，目前，研究者提出的众多众包测试优化工作大部分着眼于测试报告的处理方法，而这些繁重的后期数据处理工作都是源于不够完善的众包机制。基于这样的现状，本文首次提出一种众包测试流程的优化机制，改进众包工人的管理方式、优化测试任务的分派机制，希望能从问题的源头入手，减少不必要的融合与处理工作，实现高效高可用的移动应用众包测试。

2.2　推荐系统

互联网时代的到来带来信息的爆炸式增长，用户将会花费巨大的时间寻找自己喜欢、感兴趣的物品。如此低效率的信息获取过程无疑会流失大量的用户，基于这样的情况，推荐系统应运而生 [40]，将用户和信息相关联，根据用户的

需求、兴趣，实现真正有意义、高效率的信息获取。

一直以来，推荐系统都吸引着众多科研人员的关注，在推荐系统发展早期，Pazzani等人 [41]总结、归纳推荐系统常用的几种过滤算法，Breese等人 [42] 评价不同过滤算法对于协同推荐预测精度的影响，Herlocker等人 [43]从推荐的数据集类型、预测的评估角度等方面，分析不同协同过滤推荐算法。Adomavicius和Tuzhilin [44]总结目前推荐系统的发展情况，并指出推荐系统的未来研究方法，包括协同过滤推荐算法中的冷启动与稀疏性、基于内容推荐算法中对内容的提升以及推荐算法普适性等领域。

在近二十年的发展中，伴随着深度学习与机器学习的发展，推荐系统的健壮性、覆盖率都得到进一步的发展，电子商务、在线短视频、在线音乐以及在线阅读等，这些方面都是推荐系统在产业界的重要研究与应用场景。国内知名互联网公司字节跳动团队分别从内容分析、用户标签、评估分析以及内容安全等多个方面，对视频、文字以及图片等信息采用混合的推荐策略。

其中协同过滤推荐（Collaborative Filtering Recommendation）作为目前最为主流、应用最为深远的推荐算法，一直以来都吸引了学术界和工业级的关注

2.2.1　协同过滤推荐

协同过滤算法也称作基于用户行为的推荐，根据用户的历史行为数据，包括用户的浏览和搜索记录、用户的反馈等，寻找相类似的物品或者用户，计算当前用户对物品的兴趣程度 [45]。协同过滤算法分为以下三种：基于用户的协同过滤推荐，基于物品的协同过滤推荐以及基于模型的协同过滤推荐，下面将分别介绍这几种算法的步骤。

1.基于物品的协同过滤推荐（ItemCF）：以物品为中心，根据用户的历史偏好行为预测用户对其他物品的偏好程度。ItemCF假设用户偏好集中于某几种类型，物品A和物品B 相似度高的原因是，对物品A表示过偏好的大部分用户也对物品B表示过偏好，因此，物品A和物品B极有可能属于用户偏好的几种类型之中。当两个物品被越多用户表示过偏好，他们属于同一类型的可能性也就越高。基于物品的协同过滤推荐采用这样的设计思想，实现有限领域中的相关物品推荐。其计算步骤包括：

（1）用户偏好数据收集。根据用户行为判断用户偏好。评分、投票、转发、评论、点击、购买都是常见的用户行为，通过给不同行为赋值，获取用户对不同物品的偏好程度。

（2）计算物品之间的相似度。基于物品的协同过滤推荐算法采用如下公式定义物品i和物品j之间的相似度w_{ij} [46]：

$$w_{ij} = \frac{|N(i) \cap N(j)|}{\sqrt{|N(i) \cup N(j)|}} \tag{2.1}$$

其中N(i)表示喜欢物品i的用户集合，N(j)代表喜欢物品j的用户集合，同时喜欢物品i和物品j的用户集合则用N(i)∩N(j)代表。从公式可知，在喜欢物品i的用户中，越多人喜欢物品j，物品i和物品j的相似度则越高。

（3）预测对其他物品的喜欢程序。和当前用户历史喜欢的物品越相似，该物品则越有可能被推荐给用户。基于物品的推荐算法根据以下公式预测用户u对物品j的喜欢程度 [47]：

$$p_{uj} = \sum_{i \in N(u) \cap S(j,k)} w_{ji} r_{ui} \tag{2.2}$$

p_{uj}表示用户u对物品j的喜欢程度，N(u)代表用户u喜欢的物品集合，S(j,k)表示和物品j最相似的K个物品集合，w_{ji}表示物品j和物品i的相似度，r_{ui}表示用户u对物品i的喜欢程度。但很多情况下，用户没有对物品直接评分，而是通过诸如购买、收藏等行为反馈偏好情况，对此，推荐算法设置r_{ui} 的取值在{0,1}之间 [48]，若用户对物品表示过喜欢，r_{ui} 则为1。

（4）生成推荐列表。Top-N [49]推荐是常见的推荐形式，根据预测的喜欢程度对物品进行排名，选取排名前N个物品生成推荐列表。

2.基于用户的协同过滤推荐（UserCF）：与基于物品的协同过滤不同，基于用户的协同过滤推荐以用户为中心，计算与该用户兴趣相似的用户群体，将兴趣相似的用户群体所偏好的物品推荐给该用户。其计算步骤包括：

（1）用户偏好数据收集。同基于物品的协同过滤推荐的第一步。

（2）最近邻搜索（Nearest Neighbor Search, NNS）。选用一种相似度计算方法，寻找与当前用户A偏好最相似的K个用户 [50]。常见的相似度计算包括余弦相似度、皮尔森相关系数和杰卡德系数等。

（3）推荐物品。与基于物品的协同过滤算法一致。

3.基于模型的协同过滤算法。基于用户与物品之间的历史偏好数据，综合利用机器学习建模的思想，预测其他用户与其他物品间的偏好关系。将推荐系统与机器学习相结合，目前利用于推荐系统的主流机器学习思想包括：聚类算法、关联算法、回归算法、神经网络等。

2.2.2　冷启动

冷启动一直是协同过滤算法中的经典问题之一。协同过滤算法基于用户的历史行为数据完成对其他物品的推荐，那么如何为一个新用户完成个性化的推荐？同样，协同过滤算法又是如何将一个新物品推荐给其他用户？冷启动问题的解决方案直接影响协同过滤算法的质量。

常见的冷启动问题包括两类：

（1）用户冷启动：新用户不存在历史物品行为，因此无法获取新用户的兴趣，实现个性化推荐。在这种情况下，可以首先分析用户自身的信息。

（2）物品冷启动：在协同过滤算法，尤其是基于物品的协同过滤算法中，新物品不被推荐给用户，系统中就不会存在用户关于该物品的行为数据，因此物品冷启动利用物品的属性将新物品推荐给可能感兴趣的用户。

常见的解决方案包括：

（1）分析用户注册信息，解决用户冷启动问题。用户注册信息可以来自性别、年龄、专业、民族、籍贯等人口统计学信息，也可以来自用户对自己感兴趣类别的主动表达，或者系统通过授权信息，收集用户在其他网站上的历史行为。基于用户的注册信息，用户被划分为不同的类别，并根据所属类别中其他用户感兴趣的物品，完成向新用户的粗粒度推荐 [51]。

（2）以非个性化推荐解决用户冷启动问题。其中最常见的一种方式就是向新用户推荐热门产品，Netflix等人 [52]的调查显示，冷启动阶段的新用户对热门物品的接受度较高，长尾推荐更受老用户的喜欢。

（3）选择合适的物品启动用户兴趣。这种模式以部分物品为测试数据，记录新用户对这些物品的反馈，了解新用户的喜好，产生用户历史行为数据，实现从新用户向老用户的过渡，完成后续的物品推荐。因此，合理选择测试数据是该模式的核心，一般来说，启动用户兴趣的物品列表具有多样性、代表性、区分性以及一定的热门度。

（4）分析物品信息，解决物品冷启动问题。对于基于物品的协同过滤算法，基于用户对物品的历史行为计算物品之间相似度，物品冷启动是一个较为严重的问题，由于新物品尚未被推荐给用户，很难收集用户的行为数据。物品冷启动问题可以通过计算物品之间的相似度解决新物品的推荐问题，一般将物品转换为属性向量，采用余弦相似度的方式得到两个物品之间的相似程度 [53]。物品A和物品B的余弦相似度$similarity_{AB}$计算如下：

$$similarity_{AB} = \cos(\theta) = \frac{A \cdot B}{||A||||B||} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}} \tag{2.3}$$

使用不同的属性定义物品，将物品A和B转换为对应的属性向量，A_i和B_i分别代表向量A和向量B的各个分量。以向量空间中，两个向量夹角间的余弦值作为衡量两个个体之间差异的大小，余弦值接近1，夹角趋于0，表明两个向量越相似，余弦值接近于0，夹角趋于90度，表明两个向量越不相似。

2.3　本章小结

本章主要介绍与本文相关的工作，包括众包测试、移动应用自动化测试、安卓静态分析以及推荐系统等相关领域的研究。在学习中我们发现，目前关于众包测试机制的优化主要集中于后期的测试报告整合工作，对众包测试流程的优化，特别是测试过程中对众包工人的引导，目前还没有研究人员进行过相关研究工作。因此，可以说我们从众包测试机制的优化入手，创新性地提出基于步骤引导的众包测试流程。利用基于群体智能的思想，我们结合众包测试与工具分析，实现协作式的众包资源调度。

第三章　众包测试引导技术

这一章将会介绍基于动静态程序分析结合的移动众包测试引导技术的详细设计，整个框架的设计思想如图3.1所示，分别以动态和静态程序分析的结果引导众包测试的开展，以众包测试补充自动化测试的结果，将机器智能与人类智能互相融合，取长补短，实现高覆盖率、高效率、低投入的众包测试。

具体设计如下：使用自动化测试框架动态分析程序执行情况，提取自动化测试过程中触发异常的场景，并以此为主要测试任务，引导众包工人在不同设备、不同环境下验证异常。同时，为完善自动化测试结果，实现更加充分的应用测试，该技术在不执行程序的情况下，从Android GUI静态分析的角度，基于程序控制流分析程序GUI状态的变化，获取用户交互窗口的变化情况，并与自动化测试的窗口遍历情况对比，提取自动化测试未覆盖的窗口跳转，以其为辅助测试任务，引导众包工人探索新的异常。该技术在传统众包测试的基础上，完成整个测试机制的优化，提高测试覆盖率，实现Android应用程序更加全面与充分的测试。

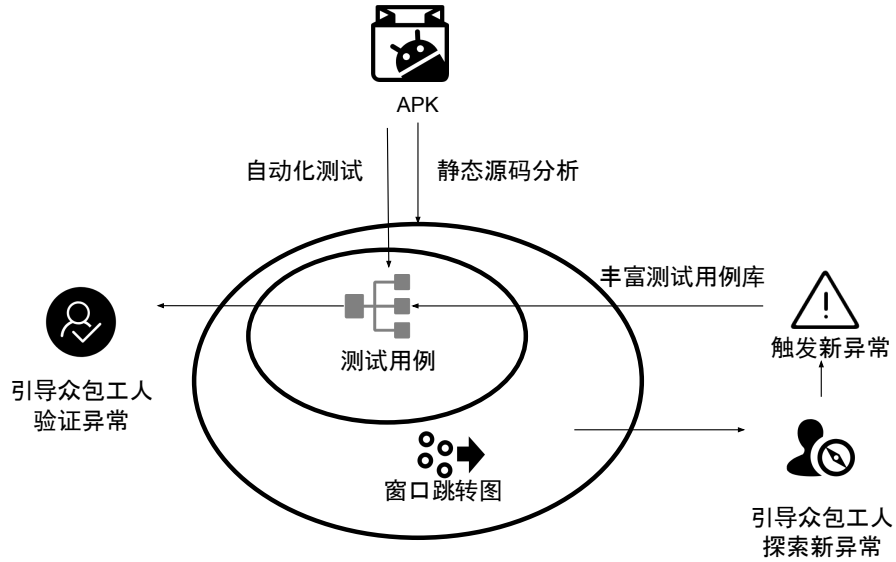

图 3.1: 众包引导技术的设计

结合测试任务分派机制和实时引导路径的计算，本文提出的众包技术具体流程如图3.2所示。任务请求者提交待测Android应用的apk后，该技术使用慕测

平台自动化测试框架和静态分析工具GATOR，分别从动态程序运行日志和程序控制流的角度，构建关于应用程序的GUI模型，描述不同事件对应的窗口跳转，该模型包含3个部分：自动化测试过程中触发异常的窗口跳转（下文简称测试用例）、自动化测试过程中的普通窗口跳转（下文简称普通窗口跳转）以及GUI静态分析对自动化测试补充的窗口跳转（下文简称未覆盖窗口跳转）。

以测试用例和未覆盖窗口跳转为两种测试任务，推荐系统根据不同类型的窗口跳转属性，分别解决测试任务推荐系统的冷启动问题，并根据协同过滤推荐算法，从众包工人的历史任务完成情况，预测当前众包工人对其他测试任务的偏好程度，基于Top-N的方式计算测试任务推荐列表。该技术为实现每个任务都能被测试的目标，筛选和过滤已被重复测试多次的任务，缓解传统推荐系统中的“马太效应”，实现对众包测试流程的管理与监听，促使众包工人之间形成协作关系。

该技术最终以Android SDK集成于待测Android应用程序中，根据当前测试任务的类型，基于众包工人的当前窗口，采用不同的方式计算窗口跳转路径，并以一次窗口跳转为单位，结合截图和文字信息，引导众包工人执行测试事件序列，分别实现测试用例的复现和新异常的探索。

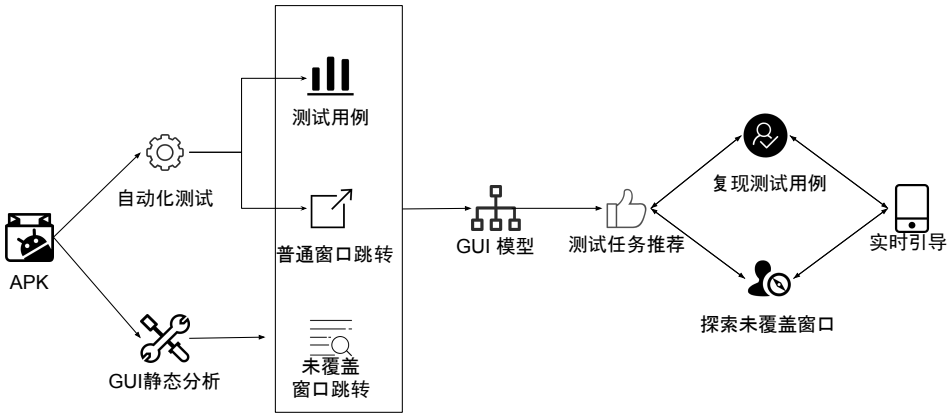

图 3.2: 基于动静态程序分析结合的移动众包测试引导技术框架

3.1　基于GUI模型的测试

传统众包测试流程中，众包平台管理人员往往根据任务请求者提供的需求文档，手工划分测试任务。这种传统的测试任务划分模式很大程度上局限于需求文档的文本描述以及众包平台管理者的专业水平。本文提出一种全新的测试任务划分模式：基于动静态程序分析的结果构建Android GUI模型，以模型中自

动化测试触发的测试用例为主要测试任务，引导众包工人在不同应用场景下复现异常，以静态GUI分析补充的未覆盖窗口跳转为辅助测试任务，引导众包工人探索新的异常。

因此，构建Android应用程序的GUI跳转模型是整个技术实现的基础。图3.3是模块处理的流程示意图，任务请求者在众包平台上传Android应用程序apk，该技术使用慕测平台自动化测试工具与静态源码工具GATOR，分别解析待测应用程序apk，分析并处理工具的解析结果，生成待测应用的GUI跳转模型，模型包括自动化测试过程中操作事件引起的普通窗口跳转、触发异常的测试用例以及静态分析对自动化测试补充的未覆盖窗口跳转。这三部分将在3.1.2，3.1.3以及3.1.4节详细介绍。

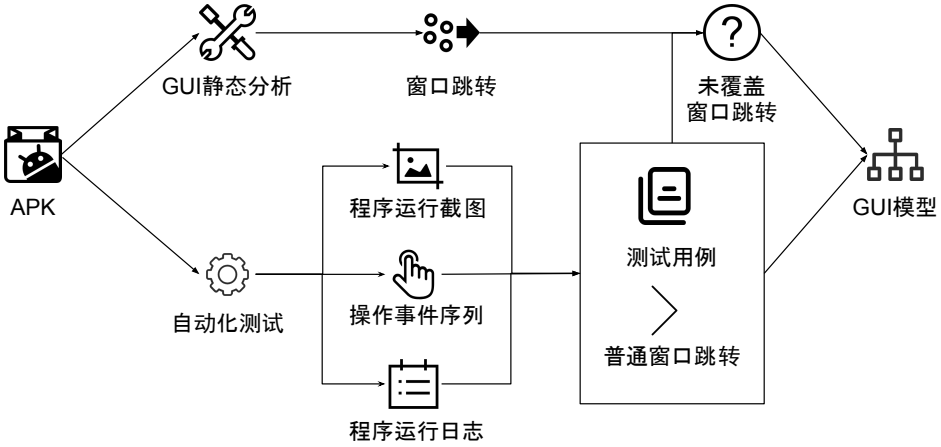

图 3.3: Android应用程序GUI模型构建流程

3.1.1　Android相关特性

为了更好地理解自动化测试和GUI静态分析的设计原理，方便后续工作的描述和介绍，本节将介绍一下Android系统的相关特性。

应用层窗口：Android应用程序由多个Activity互相堆积形成，Activity作为界面的载体，由不同的组件组成。对于一个Android界面，视图（View）管理界面中不同组件的生命周期，而Activity作为当前页面的父窗口，为每个界面定义一个界面管理器WindowManagerService（WMS），管理不同组件之间、组件与Activity之间的交流。其中Dialog和Menu作为两种常见的组件，区别于其他控件直接加载的形式，需要特定的用户行为，才能触发这两种控件在当前界面上的弹出与显示，Android将这两种组件定义为附着在父窗口Activity上的子

窗口，由父窗口的界面管理器管理其生命周期。本文以“窗口”代表应用程序中Activity，Dialog和Menu的实体类。

事件：Android应用程序事件包括系统事件和用户交互事件。系统事件发生于设备硬件端，可以分为以下几种：屏幕旋转、按返回键返回上级窗口、按Home键返回手机桌面以及程序低电量模式。基于GUI的用户交互事件则作用于当前屏幕中的窗口组件，接受来自用户的手机屏幕点击、滑动等交互操作。

回调函数：基于观察者模式，Android应用程序中的每个组件向系统注册接口、提供服务，系统监听来自用户的事件，当用户执行组件相关的操作时，则触发组件相关的接口，调用相关函数。在代码层，一个事件的发生对应一系列作用于不同对象上的回调函数。

3.1.2　普通窗口跳转

基于Android应用程序事件驱动的特征，面向Android应用程序的自动化测试框架模拟用户交互事件和系统事件，检测应用程序中存在的异常。其中，基于模型的自动化测试框架采用不同的策略遍历页面组件，跟踪应用程序相应的状态，动态构建应用程序事件与状态之间的GUI跳转模型。

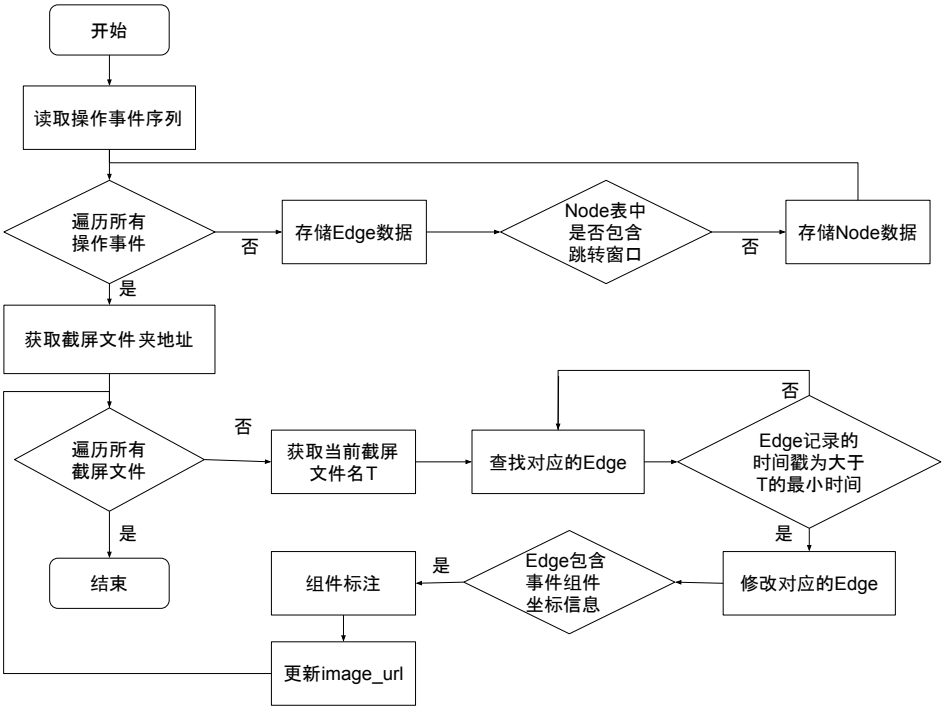

图 3.4: 普通窗口跳转处理流程图

本文选用慕测平台的自动化测试框架，该框架以人工测试脚本为辅助，辅

助遍历过程中的输入事件，采用深度优先遍历算法，希望能够遍历应用程序所有页面中的所有组件，提高应用程序的组件覆盖率。在测试过程中，自动化测试框架记录测试事件序列并截图保存测试事件发生前的Android页面运行图，这些页面图以截屏操作的当前时间戳命名，保存在指定文件目录下。该技术基于自动化测试信息的处理，以有向图结构存储每一条测试事件对应的信息，构建待测应用程序的GUI跳转模型。图3.4 显示普通窗口跳转的处理流程，下面将详细介绍普通窗口跳转的分析过程。

```
{
    "time":"2019-02-15 16:25:54,885",
    "type":"CLICK",
    "message":"Click widget //android.widget.ImageButton[contains(@index,'0')],
              bounds: '[0,72][168,240]'",
    "activityBeforeAction":".activity.MainActivity",
    "activityAfterAction":".activity.MainActivity"
}
```

图 3.5: 操作事件输出格式

首先，存储自动化测试结果。自动化测试框架以图3.5的格式记录测试过程中的事件序列：“time”字段代表操作事件发生的时间，“activityBeforeAction”和“activityAfterAction”记录事件在应用程序上触发的窗口变化，“type”字段代表事件类型，“message”字段则详细输出操作事件信息，对于不同的操作事件，自动化测试以不同的格式记录事件详细信息，表3.1总结message字段的输出格式及对应的事件信息。

表 3.1: 自动化测试事件输出信息

输出内容	事件信息	事件类型
Click Return button because this page has done	按返回键返回上一级窗口	系统事件
Click Home button has tries more than 3 times	程序无响应，按Home 键退出应用程序回到手机桌面	系统事件
以“Click widget”开头，类似“Click widget //android.widget.ImageButton, bounds:[0.72][168.240]”	事件作用的窗口组件名称及坐标	用户交互事件

遍历事件序列，该技术以有向图的结构存储自动化测试记录的GUI状态变化情况：$G =< N, E >$，其中：

节点集合N：Android应用层所有窗口的集合，

边集合E：事件触发的窗口跳转集合，对于E中任意一条边e_{ij}，其对应的起始节点和目标节点分别为n_i和n_j，这两个节点都属于节点集合N。以节点n_i为起始节点、n_j为目标节点，E 中可能存在多条有向边$E_{ij} = \{e_1, e_2, ...\}$，但集合E中每条边代表的事件信息都不同。

边的具体存储结构如表 3.2所示，source_node和target_node分别存储起始窗口和目标窗口的名称，event_type记录事件类型，event_handlers 对应自动化测试事件中的message，保存事件详细信息，image_url将操作事件与测试过程中的页面截图对应，对于触发异常的事件，message字段记录异常日志，create_time保存事件发生的时间。数据存储中设置data_type字段，用于区分GUI模型中的窗口跳转类型，其中0代表未覆盖窗口跳转，1代表普通窗口跳转，2代表触发异常的测试用例。

表 3.2: GUI模型中事件的具体存储结构

列名	类型	描述	备注
id	BIGINT	not null;PK	
source_node	varchar(255)	not null	
target_node	varchar(255)	not null	
event_handlers	varchar(1000)	not null	
event_type	varchar(1000)	not null	
data_type	INT(11)	default 1	
app_key	varchar(255)	not null	程序标识
image_url	varchar(1000)	not null	截图地址
create_time	TIMESTAMP	not null	时间戳
message	varchar(1000)	not null	异常详情

其次，是对截图信息的处理。为了帮助开发人员更快更准地定位产品异常，测试工程师往往需要提交异常触发的步骤，减少开发人员无效的查找与定位工作。由此可见，决定一个异常能否复现的因素有很多种，不仅仅来自当前触发异常的测试事件，和当前程序运行环境以及测试事件步骤都有一定的关系。在真实产业应用中，对于一些难以复现的异常，测试工程师提交异常发生情况下的程序运行图，辅助说明异常信息。

因此，该技术提出以截图信息辅助测试事件的说明。自动化测试框架在测试过程中截图保存应用程序当前运行状态，并以当前时间戳命名。结合用户交互事件相关组件的坐标，我们标注当前事件作用的窗口组件，辅助描述事件，实现用户交互事件作用组件的清晰定位，提供明确的测试引导。同时，程序运行截图中包含一定的前置测试事件信息，能够向众包工人提供相关的测试步骤提示。下面将介绍截图信息的处理步骤：

（1）将截图与事件对应：读取截图列表P = {p_1,p_1,...}，对于任意一张截图p_i，其文件名为n_i，都可以在应用程序边集合E中找到一条边e_i，满足e_i 对应时间戳t_i大于n_i且最接近n_i，从而建立p_i 与e_i的映射关系。

（2）标注用户交互事件的对应组件：一个Android应用窗口中包含多个组件，那我们如何清晰地向众包工人描述事件相关的组件呢？对于用户交互事件，自动化测试在“message”字段中输出对应组件坐标信息，我们根据截图对应的测试事件，寻找组件坐标信息，标注截图中相关组件，作为描述事件的辅助信息反馈给众包工人。而系统事件不涉及窗口组件，则直接保存自动化测试中的原截图，不予处理。

充分利用自动化测试过程中的程序运行截图能够准确、清晰地向众包工人描述测试环境，是测试事件重要的说明辅助。

图3.6展示应用程序“简豆”中，部分测试事件对应的截图。

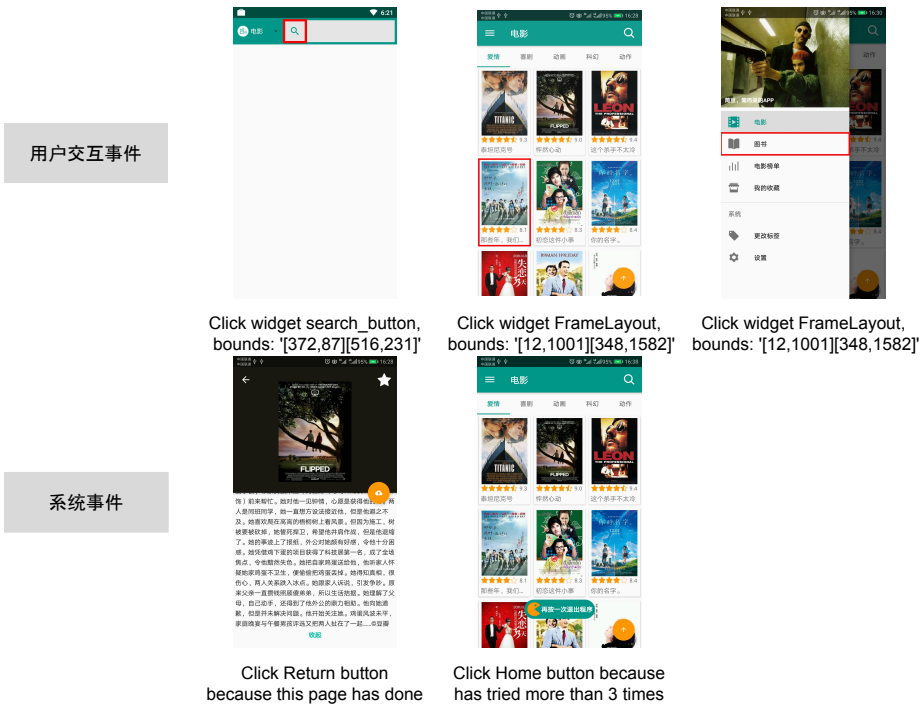

图 3.6: 不同测试事件对应的程序运行截图

3.1.3　测试用例

Android应用程序运行过程中存在三大异常，造成程序崩溃：

（1）RuntimeException：Java端的运行时异常导致jvm退出，如空指针异常等，程序日志将输出Java调用栈。

（2）ANR：安卓系统为提高用户体验而设置的保护机制，防止应用在主线程操作时间过长、程序长时间无响应产生。

（3）Native信号异常：底层Android框架中C/C++开发的问题导致进程收到错误信号，程序日志将输出汇编的堆栈信息，包括错误信号、寄存器快照以及调用栈信息。

我们收集应用程序运行异常，包括RuntimeException和ANR异常。Native而信号异常作为Android框架层面的异常，在针对应用程序的测试过程分析中，暂不收集相关异常信息。

```
{
  "id": 6913,
  "type": W,
  level": System.err,
  "LOG": Caused by java.net.SocketException: Socket is closed
  "time: 2019-02-15 16:26:06.246
}
```

图 3.7: 自动化测试中程序日志输出格式

自动化测试框架导出程序运行日志，每一条日志的格式如图3.7 所示。当应用程序启动时，Android系统为其开启一个新的执行线程，该执行线程作为主线程产生其他子线程，在子线程中运行应用程序中的其他窗口。线程之间以线程id作为唯一标识符区别。程序日志中以线程id为标识符，记录不同线程中资源加载过程与程序运行状态。“type”字段记录程序日志的调试信息，分为以下5类：“V”代表VERBOSE，不过滤地输出程序所有调试信息，“D”代表DEBUG，输出调试信息，“I”代表INFO，输出提示类信息，“W”代表WARNING，输出警告信息以及“E”代表ERROR，输出错误信息。“LOG”字段则接收来自程序具体代码层的具体信息，“time”存储异常产生的时间戳。

测试用例的处理流程如图3.8所示。我们遍历程序日志，读取LOG存储的程序运行信息，其中以“Caused by”开头的LOG输出Java端程序运行异常的详细信息，包括“Timeout”的LOG输出Android端无响应窗口信息，查找具有类似特征的日志，构成该应用程序的异常日志集合L = $\{l_1,l_2,...\}$。对于L中任意一条日志l_i，其对应时间为t_i，在应用程序边集合E中，必定存在一条边e_i，其发生时间在小于t_i的边中最靠近t_i。该技术为每条异常日志寻找其对应的测试事件信息，修改相应的窗口跳转信息，设置data_type 字段为2，并将日志中的LOG信息输出到message字段中。

3.1.4　未覆盖窗口跳转

Android应用程序的图形用户界面（GUI）响应来自用户的交互事件。受GUI驱动，交互事件作用的对象以及相关的事件处理机制决定程序代码

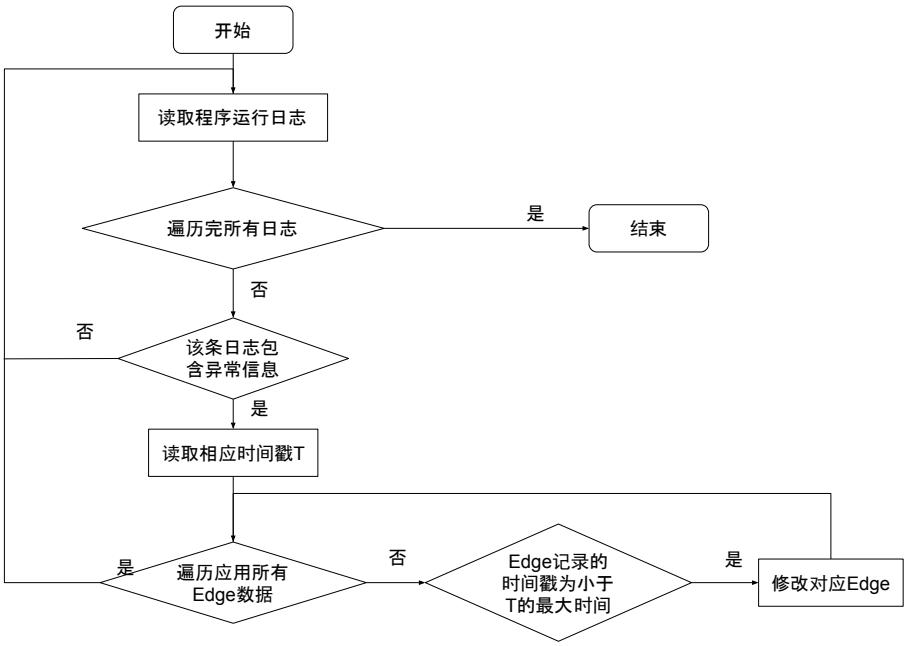

图 3.8: 测试用例处理流程

层的数据流和控制流。Android 应用程序的静态分析在不运行程序的基础上，通过检查代码中的语法、结构、接口等信息，建立Android程序对象、对象间交互、对象与Android框架间交互的模型。Yang [54]等人基于静态控制流分析，遍历并分析上下文相关的控制流路径，识别可能触发回调的语句，分析受用户事件驱动的组件，捕获从Android框架到程序代码层的回调序列。Yang [22]等人则在之前工作 [55]的基础上，开发客户端工具GATOR。该工具通过分析Android代码的分析，描述应用程序的GUI变化，生成窗口转换图（Windows Transaction Graph，即WTG），窗口转换图中的节点为程序窗口（window），包括Activity，Menu，Dialog这三个实体类，图中边记录触发窗口变化的程序回调信息，其每一条边包含的信息如图3.9所示，其中“source_node”和“target_node”分别存储图中边的起始节点与目标节点名称，即事件触发的GUI状态变化情况，“event_handlers”则存储回调函数发生的窗口以及回调函数的相关定义。GATOR通过解析回调函数的名称与定义，建立回调函数与用户交互操作的映射，将回调函数对应的事件信息存储在“type”字段中。

未覆盖窗口跳转的提取流程如图3.10所示，该技术认为，自动化测试框架的操作序列记录应用程序窗口间的多种跳转形式，而GATOR生成的窗口转换图中，边代表移动应用窗口之间是否存在跳转。因此，提取过程中首先遍

```
{
    "sourceNode": MainActivity,
    "targetNode": Launcher,
    "type": press_key
    "event_handlers": [<com.lhr.jiandou.activity.MainActivity: boolean
                        onKeyDown(int,android.view.KeyEvent)>]
}
```

图 3.9: WTG事件输出格式

历WTG中的边，根据起始节点与目标节点信息检索自动化测试的事件信息，判断自动化测试过程中是否已经覆盖此窗口跳转，如没有包含该条窗口跳转，该技术根据3.1.3中定义的普通窗口跳转格式存储来自WTG的边信息，并以data_type为1标识该条跳转是GATOR对自动化测试事件序列的补充。当然，窗口跳转的遍历过程，也是对自动化测试记录的窗口列表的不断补充。

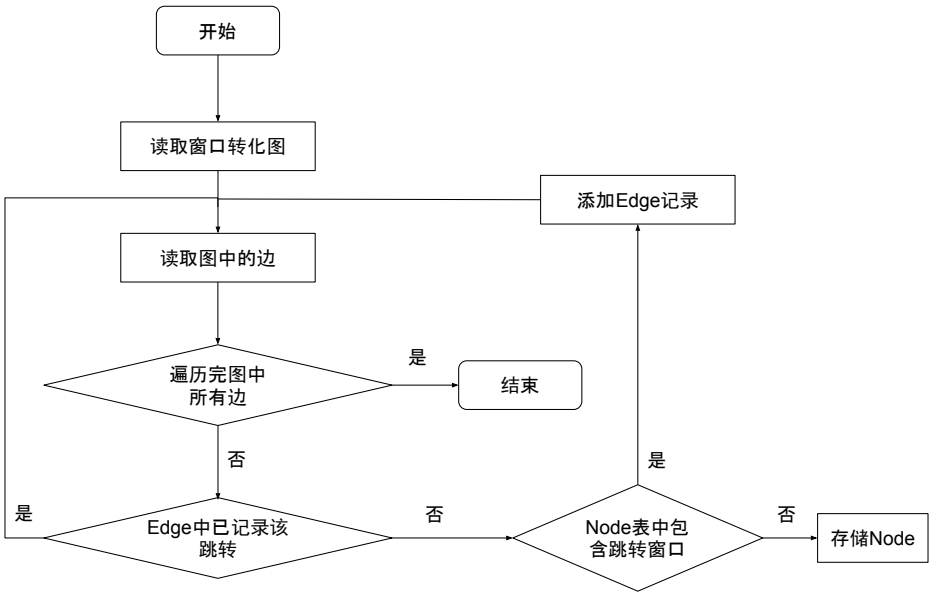

图 3.10: 未覆盖窗口跳转的提取流程

和测试用例一样，我们也为未覆盖窗口路径添加截图作为辅助引导信息。由于GATOR在应用程序不运行的情况下分析窗口跳转情况，遍历过程中无法获得应用程序窗口对应的截图，在这里我们以人工辅助的方式，手动截取未覆盖窗口的程序运行图并保存。

应用程序的GUI模型包含测试用例、未覆盖窗口跳转以及普通窗口跳转。其中测试用例和未覆盖窗口跳转作为测试任务，区别于传统众包测试中文本描述测试需求的模式，从更加专业化的角度划分测试任务，防止一定的主观性和局限性。同时，传统众包测试过程中，众包工人根据测试需求探索异常，该技

术则明确地指出哪些可能触发异常的行为，引导众包工人复现异常，很明显，和传统众包测试相比，基于动静态程序分析结合的众包流程更加简单，测试任务划分更加专业，对众包工人的门槛更低，但测试效率会更高。

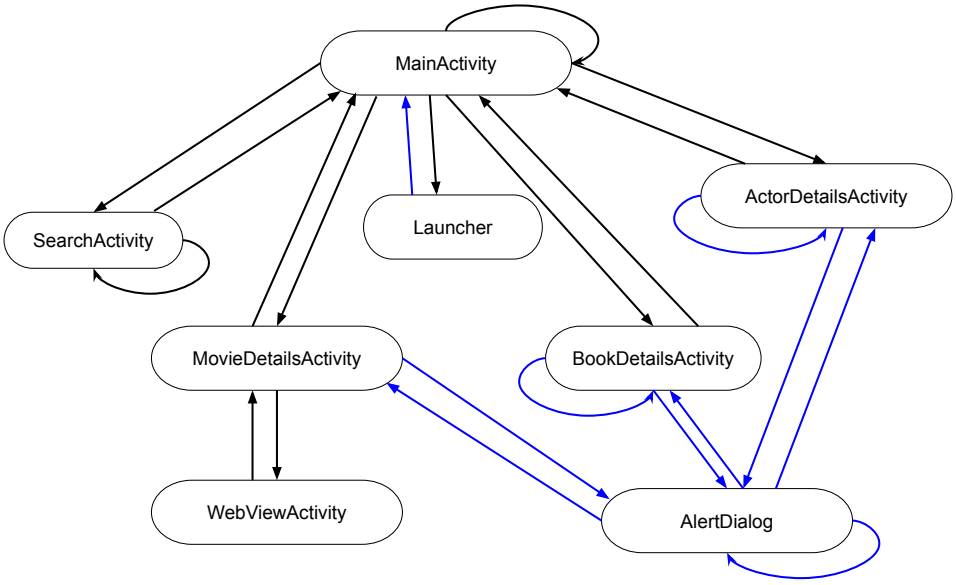

图 3.11: 应用程序“简豆”的部分GUI模型

普通窗口跳转作为应用程序GUI模型的重要补充，在后期的测试路径计算和测试事件提示中，起到了重要的辅助作用。

与自动化测试和GUI静态分析单独构建的GUI模型相比相比，该技术构建下的GUI变化模型更加全面。一般情况下，该模型具有以下特征：

（1）GUI模型呈树形结构，MainActivity一般作为程序首页、GUI模型的根节点，从MainActivity出发，用户可以跳转到应用程序的不同窗口。

（2）一般情况下，除首次安装时的产品介绍和宣传窗口，对于GUI模型中的其他所有窗口，若存在一条从窗口A跳转到窗口B的边，则必定存在另一条从窗口B跳转到窗口A 的边。

图3.11显示应用程序“简豆”的部分GUI模型，由于两个窗口之间可能存在多种跳转方式，GUI模型中两个节点同一方向上可能存在多条边，为了便于展示，对于某个方向上存在跳转的两个节点，图3.11只选取两个节点在该方向上的一条边。其中黑色边代表来自自动化测试的窗口跳转，蓝色边则来自GUI静态分析补充的窗口跳转。

3.2　测试任务推荐

在信息过载的背景下，用户很难从大量的物品中找到自己感兴趣的物品，推荐系统应运而生，根据用户的历史行为，定义用户与物品的之间的关联，预测用户对其他物品的喜好程度，满足用户的个性化需求。目前推荐系统已经得到广泛的应用，有机结合数据、算法、人机交互、架构等环节，逐步使用机器学习、信息检索等技术，推荐系统的构建技术已经较为成熟。协同过滤推荐系统是目前推荐系统中应用较为成熟和广泛地技术之一，该推荐系统一般统计用户历史偏好物品数据，计算用户之间或者物品之间的相似度，建立推荐模型，预测用户对其他物品的偏好程度。基于物品的协同过滤推荐系统基于物品相似度的计算，以较为稳定的方式生成当前用户的物品推荐列表。该推荐系统能够胜任大计算量、实时性要求较高的推荐系统，尤其适用于用户多于物品数量的情况，可以极大提高推荐系统的效率。

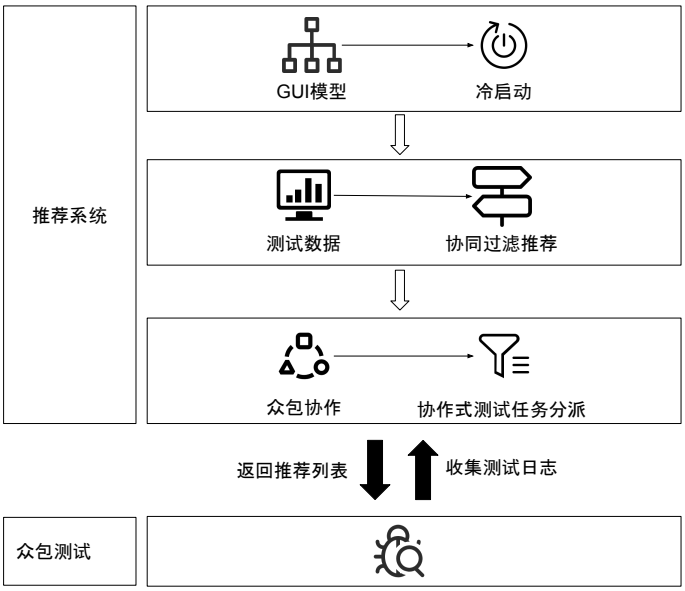

图 3.12: 测试任务推荐系统流程

我们采用基于物品的协同过滤算法，实现关于测试用例和未覆盖窗口跳转的测试任务推荐系统。但是推荐系统如何获取新用户的历史行为、如何计算新物品与其他物品的相似度？冷启动是基于物品的协同过滤推荐算法必须面临的问题，我们结合3.1中GUI模型，分析不同窗口跳转的属性特征，解决测试任务推荐系统冷启动问题。同时，将推荐系统与众包测试机制的优化相结合，从协

作式众包流程的角度出发，该技术通过跟踪不同测试任务的完成情况，提高推荐系统对物品的覆盖率，实现对不同测试任务的均匀推荐，优化众包资源配置。图3.12显示整个推荐系统的流程，下面将在3.2.1和3.2.2 节中详细介绍与众包测试相结合的推荐系统设计。

3.2.1　冷启动

基于协同过滤的推荐算法根据用户对物品的历史行为数据，预测用户对其他物品的喜好程度，完成个性化的物品推荐。但是当系统中新用户或者新物品加入时，即没有用户对物品的历史行为数据的情况下，如何对用户进行推荐呢？冷启动问题就此衍生。

冷启动问题可以分为以下两类：

1. 用户冷启动：针对新用户的个性化推荐，

2. 物品冷启动：将新物品推荐给可能感兴趣的用户，

在众包机制中，众包工人的管理是松散、无组织的，因此每开展一次众包测试，面向众包测试的推荐系统面临都会面临新的众包工人、新的测试任务，冷启动成为测试任务推荐系统必须解决的重要问题之一。

在面向未覆盖窗口跳转的测试任务推荐系统，由于未覆盖窗口跳转很难找出具有代表意义的跳转属性，因此该技术采用随机策略解决冷启动问题。下面将主要从用户冷启动和物品冷启动两个方面，介绍测试用例推荐系统冷启动阶段的解决方案。

针对新的众包工人，面向测试用例的测试任务推荐系统使用测试数据启动众包工人的兴趣。其步骤如下：

（1）异常分类：收集应用程序GUI模型中的异常信息，我们对Android 应用程序的常见异常进行分类，并手工完成对异常严重程度的评分，分数越高越严重，如表 3.3 所示。

（2）异常选择：根据当前应用程序在自动化测试过程中触发的异常分布情况，我们均匀地从每个类型中随机选取一定数目的异常，组成具有多样性和区分性的测试数据，推荐给每个新加入的众包工人。

至此，冷启动处理过程中的新用户问题已经解决，但新用户的冷启动阶段数据往往较为稀疏，推荐效果不佳，因此，在解决新用户冷启动问题的基础上，我们研究新物品的冷启动解决方案。

新物品的冷启动是测试用例推荐系统中的重要部分，因为在该技术中选用的是基于物品的协同过滤算法，这种基于用户的历史偏好行为计算物品相似度

的算法，当新物品加入时，物品的相似度关系中不存在关于新物品的信息，因此，新物品面临无法推荐给用户的困境。我们通过分析异常的属性，基于余弦相似度计算不同测试用例间的相似度，实现粗粒度的个性化推荐。物品冷启动阶段维护二维的物品相似度表，并频繁更新该表单，以获取准确的相似度。

表 3.3: Android应用程序常见异常类型及严重程度

类型	代表异常	严重程度
资源加载异常	*android.content.pm.PackageManager* *NameNotFoundException* *java.lang.FileNotFoundException* *java.lang.FileNotFoundException* *java.lang.IllegalAccessException* *android.system.ErrnoException*(*Nosuchdirectory*) *java.lang.NoSuchMethodException* ……	6
输入意图错误	*java.lang.NullPointerException* *java.lang.IllegalArgumentException* *java.lang.ArrayIndexOutOfBoundsException* *java.lang.ClassNotFoundException* *java.lang.ClassCastException* ……	5
网络通信异常	*java.net.ConnectException* *java.net.SocketException* *NetworkError* ……	4
ANR	*KeyDisPatchTimeout* *BroadcastTimeout* *ServiceTimeout*	3
线程间异常	*java.lang.SecurityException*(*Permissiondenied*) *android.os.DeadObjectException* ……	2
其他异常	*java.io.IOException* *ExecutionException* 引用外部包产生的异常 ……	1

基于以上分析，我们结合3.1构建的GUI模型，通过异常发生的起始窗口层级、测试事件类型以及异常类型这三个属性描述测试用例，将测试用例抽象为属性向量：对于测试用例t_i，其内容可以表示为一个属性向量：

$$t_i = \{(e_1, w_1), (e_2, w_2), (e_3, w_3)\} \tag{3.1}$$

其中e_1代表异常类型，其对应的权重w1依据表3.3中定义的异常严重程度。e_2代表异常发生的起始窗口，其对应的权重w2代表窗口页面层级：对于应用程序窗口集合W = {w_1,w_2,...}，对于其中任意一个窗口w_i，其对应的页面层级l_i = MainActivity到w_i的最短路径+ 1。以图3.11为例，MainActivity窗口对应的权重为1，SearchActivity等窗口对应的权重为2，WebViewActivity窗口对应的权重则为3，从提高测试覆盖率的角度出发，窗口嵌套越深，异常越不容易被发现，越要被推荐给众包工人，赋予的权重越高。e_3代表操作事件类型，该技术为不同的事件类型赋予不同权重，如表 3.4所示：

表 3.4: 事件类型及权重对应表

操作事件类型	权重
作用于手机按钮的系统事件	1
作用于手机屏幕中程序组件的用户交互事件	2

将测试用例转化为属性向量，该技术采用余弦相似性计算测试用例A与测试用例B之间的相似度$similarity_{AB}$，其中A_i代表测试用例A的某个属性向量：

$$similarity_{AB} = \cos(\theta) = \frac{A \cdot B}{||A||||B||} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}} \tag{3.2}$$

表 3.5显示在测试用例推荐系统冷启动阶段，对Android应用程序“简豆”的部分测试用例，计算的物品相似度，由于物品相似度的二维矩阵过大，表3.5只截取其中一部分。

3.2.2　基于物品的协同过滤推荐

协同过滤分析用户的历史行为数据，寻找与用户兴趣相投的用户，预测用户对其他物品的喜好程度。基于协同过滤的推荐系统已经成为当下的主流推荐算法，已经在电子商务、社交网络中得到了广泛的应用。基于物品的协同过滤推荐作为协同过滤的重要部分，以物品相似性为基础，向用户推荐和他历史偏好物品类似的物品。和基于用户的协同过滤相比，基于物品的协作过滤推荐更

表 3.5: “简豆”部分测试用例相似度

id	99	102	105	108	112	114	115	117
99	1	0.34	0.57	0.19	0.21	0.65	0.71	0.49
102	0.34	1	0.22	0.43	0.76	0.23	0.89	0.11
105	0.57	0.22	1	0.17	0.63	0.55	0.47	0.39
108	0.19	0.43	0.17	1	0.57	0.47	0.27	0.19
112	0.21	0.76	0.63	0.57	1	0.63	0.83	0.45
114	0.65	0.23	0.55	0.47	0.63	1	0.26	0.15
115	0.71	0.89	0.47	0.27	0.83	0.26	1	0.37
117	0.49	0.11	0.39	0.19	0.45	0.15	0.37	1

加稳定，可离线完成大量的计算工作，对于实时性要求较高的系统来说，推荐效率更高，推荐结果更稳定。

测试任务推荐系统以众包测试开始的前5分钟为冷启动阶段，在收集到一定的众包工人偏好数据后，采用基于物品的协同过滤算法完成测试过程中的推荐系统，以测试用例为例，已知当前众包工人u选择的测试用例集合，计算对其他未选择测试用例的兴趣度，计算步骤如下：

（1）计算测试用例之间的相似度。与冷启动阶段通过物品属性计算测试用例相似度的方式不同，该技术监听众包工人行为，记录不同测试用例在推荐列表中被众包工人选择的情况。基于该数据计算不同测试用例之间的相似度。测试用例i和测试用例j相似度w_{ij}采用如下公式计算：

$$w_{ij} = \frac{|N(i) \cap N(j)|}{\sqrt{|N(i) \cup N(j)|}} \tag{3.3}$$

其中N(i)表示在测试过程中选择测试用例i进行测试的众包工人集合，N(j)代表选择测试用例j进行测试的众包工人集合，$N(i) \cap N(j)$则代表同时选择测试用例i 和测试用例j的众包工人集合。在选择测试用例i的众包工人中，越多众包工人选择测试用例j，测试用例i和测试用例j的相似度越高。

（2）预测众包工人对其他测试用例的偏好程度。根据众包工人u选择的历史测试用例集合N(u)，测试用例推荐系统基于KNN的方式预测众包工人u对测试用例j的兴趣程度p_{uj}，其计算公式如下：

$$p_{uj} = \sum_{i \in N(u) \cap S(j,k)} w_{ji} r_{ui} \tag{3.4}$$

其中S(j,k)代表和测试用例j最相近的K个测试用例的集合，在众包工人u已经测试的测试用例集合N(u)中，对于每一个测试用例i，计算测试用例j 和测

试用例i之间的相似度w_{ji}，r_{ui}则代表众包工人u对物品i的偏好程度，在本技术中，r_{ui}取值在{0,1}之间，若众包工人u选择测试用例i，则代表众包工人对测试用例i表示过偏好，$r_{ui}=1$，否则，则为0。

（3）以Top-N的方式生成测试列表。根据当前众包工人对测试用例的兴趣度排名，选取兴趣度最高的N个测试用例，返回给当前众包工人。

未覆盖窗口跳转的推荐系统计算步骤同理，此处不再赘述。

3.2.3　协作式测试任务分派

社会学中有一种现象叫做“马太效应” [56]，即“强者越强，弱者越弱”，相应的，在推荐系统中也存在这样的现象：热门的物品就会越热门，而冷门的物品最终都会走向无人问津的地步。推荐系统设计的初衷则是为了消灭“马太效应”，让用户摆脱热门排行榜的干扰，根据自己的喜好找到真正感兴趣的物品。但是Hao等人 [57] 的研究显示，目前推荐系统都存在一定程度上的“马太效应”。在结合众包测试的推荐系统中，如何缓解推荐系统的“马太效应”，实现高质量的推荐结果呢？我们从协作式众包测试流程的角度出发，结合移动应用兼容性测试的相关知识，提出一种创新的推荐数据过滤策略，优化众包机制下的推荐系统，实现众包资源的合理配置。

传统Android应用程序往往通过设置测试设备集群，完成兼容性测试，集群中包含当前市场常用、而非所有的手机型号。因此，在参与人数为N的应用程序众包测试中，对应用程序的测试用例集T = {t_1,t_2,...}，我们定义“已验证异常”的概念：根据众包工人参与情况，设立兼容性测试阈值S，在众包测试的过程中，任意一个测试用例t_i被众包工人选为测试任务的次数超过阈值S，测试用例t_i 对应的异常则认为已经被验证，不会再出现在推荐列表中，推荐系统将引导众包工人验证和探索其他异常，提高推荐系统的整体覆盖率，缓解长尾效应 [58] 的影响。因此，从改进众包测试机制的角度出发，我们基于协作式众包的理念，跟踪众包测试过程中，测试任务的完成情况，在3.2.2中推荐列表生成的过程中，过滤和筛选测试任务，实现众包测试任务的实时分派与调度，优化众包资源配置。

而对于未覆盖窗口跳转，由于一次窗口跳转对应多种方式，即多种事件，因此，为鼓励众包工人探索更多应用程序可能存在的异常，未覆盖窗口跳转的推荐系统不包含此步骤。

3.3　测试实时引导

该技术最终以Android SDK方式集成于手机端，众包工人无需跳转到网页

端，在设备端可以直接查看相关测试任务、请求来自服务器端的步骤提示。图3.13 显示该技术在Android端引导众包工人完成测试任务的流程。查看3.2节推荐系统生成的列表，判断众包工人选择的测试任务类型（为表述方便，测试任务，即众包工人选中的窗口跳转在下文中称作目标窗口跳转），根据不同类型的测试任务，采用不同方式完成相应的测试目的：若测试任务为测试用例，则引导众包工人复现异常，若为未覆盖窗口跳转，则引导众包工人到达目标窗口挑战的起始窗口，探索新的异常。

因此，无论是哪种类型的测试任务，实时引导都包含两个步骤，首先引导众包工人从当前窗口到达目标窗口跳转的起始窗口，其次对于测试用例，完成相应的前置事件后，引导众包工人执行触发异常的操作事件，对于未覆盖窗口跳转，该技术则提示众包工人触发相应的窗口跳转。基于这样的分析，Android端的实时引导首先需要引导众包工人到达目标窗口跳转的起始窗口，计算众包工人从当前窗口到目标窗口跳转起始窗口的跳转路径。其次，以一次窗口跳转为单位，提示众包工人执行相关事件序列，引导众包工人遍历跳转路径，完成测试任务。3.3.1和3.3.2节将分别从跳转路径计算和实时提示两个部分介绍Android端的实时引导机制。

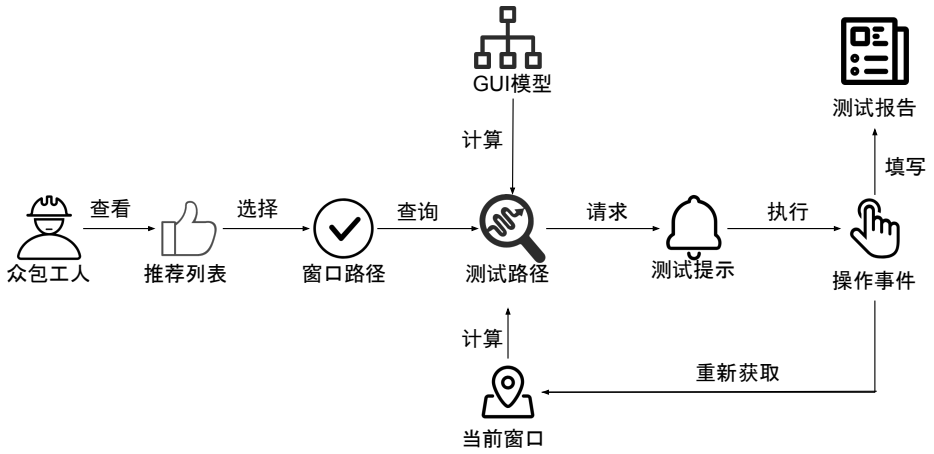

图 3.13: Android端的实时引导流程

3.3.1　跳转路径

对于选择测试用例的众包工人，其测试任务在于复现该异常。如何复现一个异常？仅仅执行该异常对应的操作事件是不够的，决定一个异常能否复现的一个重要因素就是测试事件步骤的还原，即触发异常的场景还原。因此，为了帮助开发人员更快、更准地定位异常，该技术针对异常的复现常见还原，提出

基于前置测试事件的跳转路径计算，其计算流程如图3.14 所示，其中具体处理细节如下：

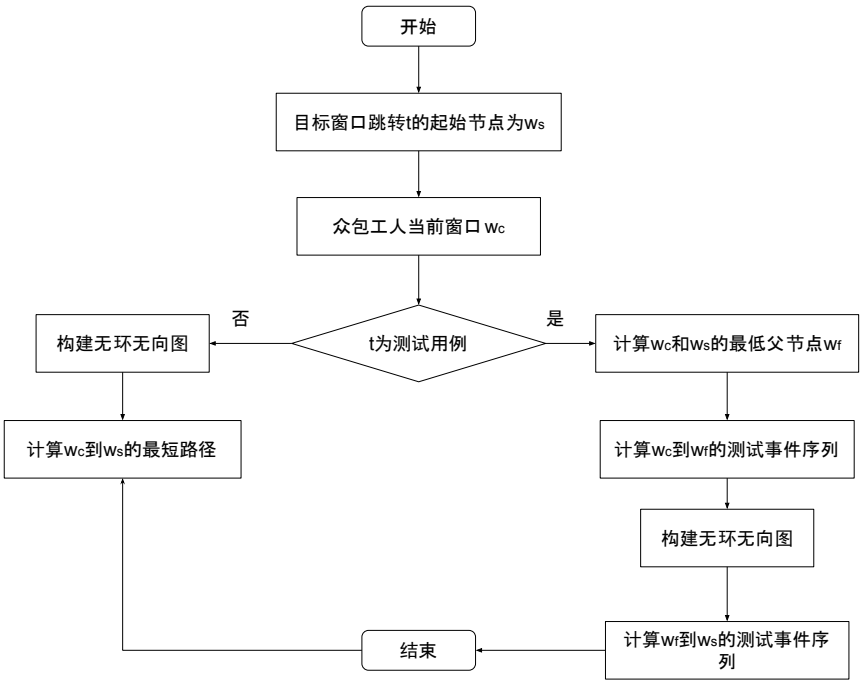

图 3.14: 跳转路径计算流程

众包工人选中一个测试任务，其对应的目标窗口跳转为t，t对应起始窗口为w_s，众包工人当前窗口为w_c，

（1）寻找w_s和w_c的最低父节点w_{father}。在Android应用程序中，一个组件可能对应多个内容不同的窗口，应用程序GUI模型中的环大多由此产生。在应用程序“简豆”中，MovieDetailsActivity和ActorDetailsActivity，这两个窗口都可以触发登录豆瓣的弹窗AlertDialog，如图3.15中，用户可以分别从查看泰坦尼克号电影详情和导演安东尼·罗素介绍的页面登录豆瓣，但是在实际使用中，和3.11中GUI模型不同的是，用户不可以在从泰坦尼克号电影详细的页面登录豆瓣后，到达安东尼·罗素的介绍页面。因此，计算两个节点的最低父节点时，以MainActivity为GUI模型的顶部方向，往顶部方向计算两个节点的最低父节点，避免环造成的错误信息干扰。

（2）寻找从w_s到w_{father}的前置测试事件序列。以目标窗口跳转t 为当前窗口跳转、以t对应的测试事件为当前事件，从目标窗口跳转t开始，查找以w_s为目标窗口的窗口跳转集合，并在集合中找到一条早于且最靠近当前事件发生时间的窗口跳转为前置测试事件。以前置事件为当前事件，采用同样的策略寻找当前

图 3.15: 应用程序“简豆”中弹出的豆瓣账号登录窗口

测试事件的前置事件……反向迭代查找，直到前置事件的起始窗口为w_{father}。

（3）寻找从w_c到w_{father}的测试事件序列。和步骤（2）不同，该技术认为从w_s到w_{father}的窗口跳转都是复现测试用例的前置事件，从w_c到w_{father}则是普通测试事件，因此，这里采用不同的计算方式获取相应的测试事件序列。对于Android应用程序，从产品设计的角度来说，任意两个窗口之间的跳转都是双向的，即若存在一条从w_a到w_b的边，则必定存在一条从w_b到w_a的边。因此，GUI模型转化为生成无环无向图：在两个节点只保留一条边，去除构成环的组件节点（即Dialog和Menu类型的窗口）及相应边，基于无向图计算从w_c到w_{father} 的最短测试路径。根据测试路径中的节点信息查找对应边，此时该技术尽量选择自动化测试中的普通窗口跳转，组成相应的测试事件序列。

而对于选择未覆盖窗口的众包工人，其测试路径的计算步骤则相对简单，因为我们只需要引导众包工人到达目标窗口跳转的起始节点，无需考虑前置测试事件，因此，其跳转路径的计算同基于前置测试事件的跳转路径中步骤（3），直接计算从w_s到w_c的最短路径。

无环无向图中两个节点间的最短路径计算计算过程如下所示：队列queue保存图中节点，基于广度优先遍历策略，处理其他节点的邻接节点。

Algorithm 1: 最短路径计算

```
queue ← currentWindow
distance ← 0
while queue is not empty do
    v ← queue.poll();
    foreach edge in v.adjEdges do
        if edge.targetVertex.distance is MAXVALUE then
            edge.targetVertex.distance ← v.dist+1;
            queue ← queue.offer(edge.targetVertex);
            edge.targetVertex.preVertex ← v;
```

3.3.2 实时引导

在测试过程中，不断以众包工人的当前位置查询测试路径，请求操作事件提示，根据众包工人的位置信息，我们向众包工人提供不同窗口跳转信息，实现多样化的测试提示与引导。

众包工人在测试路径中的状态分为以下几种：

（1）执行测试事件序列。以众包工人当前窗口w_s查询3.3.1中计算的测试事件序列，获取即将执行的测试事件，使用文本描述该测试事件信息的基础上，该技术以程序运行截图作为辅助说明，帮助众包工人理解事件发生场景。

（2）完成所有事件的执行。众包工人在执行完所有事件序列后，若仍然请求测试提示，此时我们认为当前测试任务已完成，该技术将提示众包工人选择新的测试任务。

3.4 本章小结

本章作为本文的核心章节，详细描述该技术的框架与设计。以程序分析引导众包测试的开展，其中主要以自动化测试和静态分析生成的结果为基础，完成测试任务的自动分解与专业化设计，构建关于应用程序的GUI模型。结合协作式众包测试机制的优化思想，针对测试任务完成基于物品的协同过滤推荐算法，实现个性化的测试任务推荐系统，保证众包资源的合理配置。最终该技术以Android SDK集成于在设备端，可以根据不同测试任务类型，选择不同的方案计算众包工人的操作步骤：针对测试用例，跟踪众包工人的当前窗口信息，尽可能还原测试用例的前置事件，引导众包工人准确复现测试用例，而针对未

覆盖窗口跳转，该技术则从效率的角度出发，基于GUI模型构建无环无向图并计算窗口间的最短路径。最后，以一次窗口跳转为单位，引导众包工人完成测试任务，实现众包测试对程序分析的补充，达到更全面、更充分的程序测试。

以针对测试用例的推荐系统为例，图3.16显示对应用程序“简豆”开展众包测试时，基于动静态程序分析结合下的移动众包测试技术下的众包工人测试流程。众包工人首先请求测试任务推荐，查看推荐的任务列表后从中选择一条测试用例作为测试任务，我们计算从当前窗口到测试任务的事件序列，以截图辅助事件的说明，引导众包工人完成测试任务。

图 3.16: 程序运行截图

第四章　实验设计与分析

4.1　实验目的

本文提出以下几个问题：

RQ1：基于动静态程序分析结合的移动众包测试引导技术是否有效提高众包测试的质量？

RQ2：静态分析对众包工人的引导是否有效提升众包测试的覆盖率？

RQ3：众包工人对测试任务的推荐结果是否满意？

RQ1的提出是为了探索该技术对传统众包测试质量的提升，包括自动化测试中测试用例、静态分析中补充的未覆盖窗口跳转对众包工人的引导。为了回答RQ1，我们从相同时间内触发的异常以及测试完成后，“已验证异常”数目，对比该技术的众包测试和传统众包测试。

RQ2的提出是为了验证静态分析在引导机制中的必要性，即当该技术不包含静态分析的时候，没有明确地向众包工人指出自动化测试未覆盖窗口跳转的情况下，众包工人能否很好地探索新异常。因此，为了验证RQ2，对于自动化测试未覆盖的窗口跳转部分，我们根据代码覆盖率，对比传统众包测试与静态分析引导下的众包测试。

RQ3的提出为了了解众包工人对测试任务推荐结果的满意度。用户作为推荐系统的参与者与实验数据产生者，其主观体验是评测推荐系统最重要的指标之一。我们采用问卷调查的形式，了解众包工人在测试过程中对测试任务推荐系统的使用体验。

4.2　实验设置

4.2.1　实验应用选择

实验选择三款Android应用程序：“简豆”、“QQ影音”以及“记乐部”。这三个应用程序的描述如下：

简豆：休闲娱乐类应用，基于豆瓣API的客户端，包括电影分类、图书分类、电影榜单以及搜索、收藏等功能。

QQ影音：影音播放器类应用，一款轻量级影音播放器，支持主流媒体格式，包括视频播放、视频下载、本地视频上传、投屏、用户互动等功能。

记乐部：办公类应用，专门针对俱乐部的团队管理应用，包括团队经费管理、人员管理、团队活动安排等功能。

每个应用在测试过程中设置三组实验：传统众包测试，只包含自动化测试引导的众包测试以及动静态程序分析共同引导下的众包测试。

4.2.2　实验机型统计

实验设置每个应用程序的参与众包工人数目为15，其中每场实验的众包工人参与数目为5。根据2018年度安卓手机的市场占比情况，我们选取5款热门测试设备，实验中各机型及Android系统的分布情况如表 4.1所示。

表 4.1: 实验设备情况统计

机型	系统版本
华为P20	基于Android8.1定制的EMUI8.1
三星Note9	基于Android9.0定制的Experience10
小米9	基于Android9.0定制的MIUI10
荣耀8	基于Android8.0定制的EMUI8.0
OPPO R11	基于Android7.1定制的Color3.1

4.2.3　实验步骤

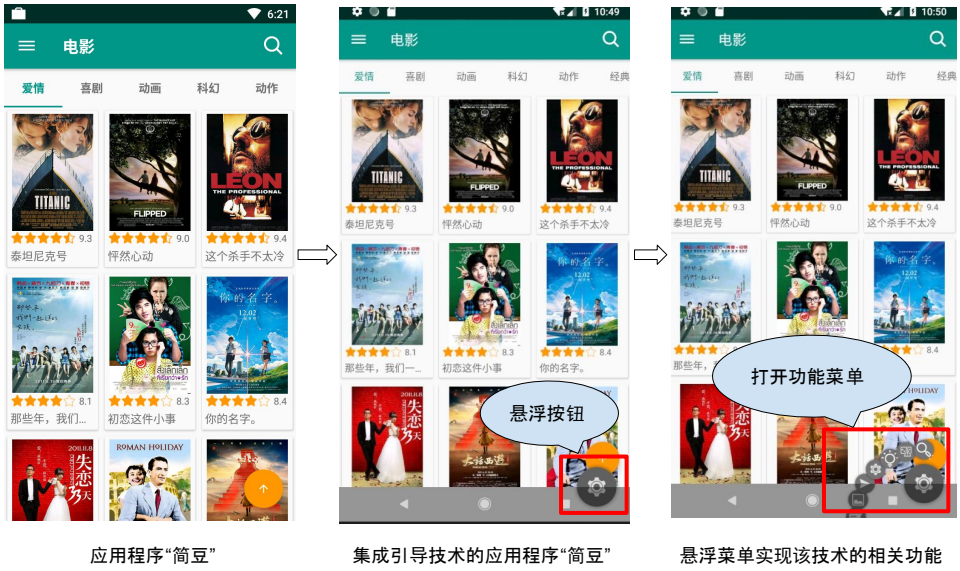

图 4.1: 集成该技术的应用程序“简豆”

任务请求者在网页版众包平台上传待测Android应用的apk，众包平台管理员解析apk，使用自动化测试和GUI静态分析分析待测应用，实现测试任务的专业划分与设计，构建关于待测应用程序的GUI模型。本文提出的技术以最终Android SDK的方式集成于应用程序中。以“简豆”为例，集成该技术的应用程序如图4.1所示。

以悬浮按钮组件实现相关众包测试功能，点击按钮展开对应的功能菜单，图4.1中第三张图中菜单从右到左依次实现“查看测试用例推荐列表”、“查看未覆盖窗口跳转推荐列表”、“测试提示”以及其他基本功能。根据自动化测试的分析结果，我们根据表 3.3中的异常分类设置，统计这三个应用程序的异常分布情况，图4.2显示“简豆”、“记乐部”和“QQ 影音”这三个应用程序的测试用例统计情况。

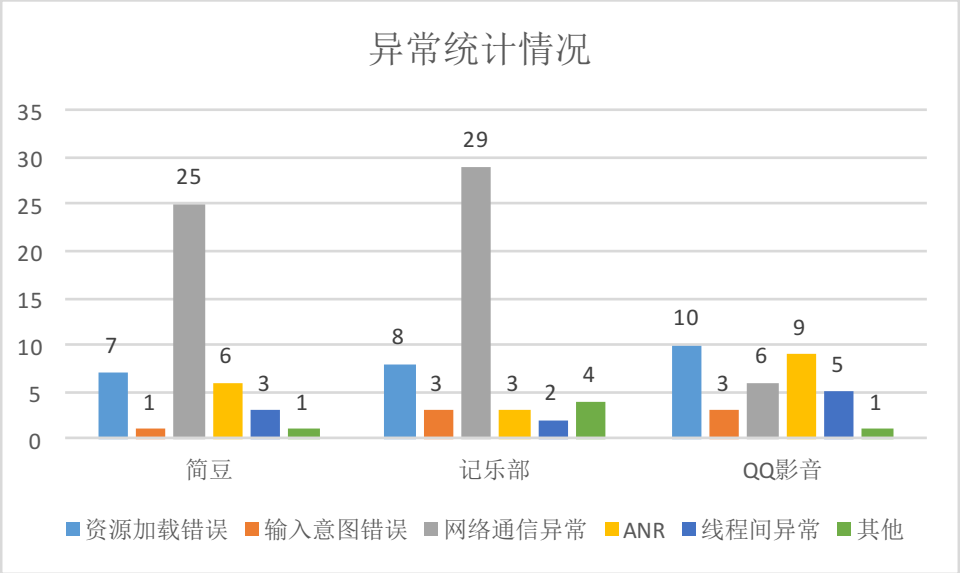

图 4.2: 测试用例类型统计情况

下面展示三个应用程序的部分GUI模型，由于两个窗口之间存在的跳转方式可能很多，我们只选取了GUI模型中的一部分：在两个窗口、一个方向上的窗口跳转中只选取其中一条记录。“简豆”的部分GUI模型如图3.11所示，“记乐部”和“QQ影音”的部分GUI 模型分别如图4.3和4.4所示，图中黑色边代表来自自动化测试的窗口跳转，蓝色边则来自静态分析的补充。

生成应用程序的GUI模型后，众包平台管理人员重新构建已集成该技术的应用程序，生成新的apk提供给众包工人进行测试。众包工人在测试平台上领取测试任务并在手机端安装新的apk，该技术摆脱传统众包测试需要众包工人在Web端查看测试任务、填写测试报告的方式，以内嵌于应用程序的形式，直

接在设备端引导众包工人在设备端查看推荐列表，完成测试任务。

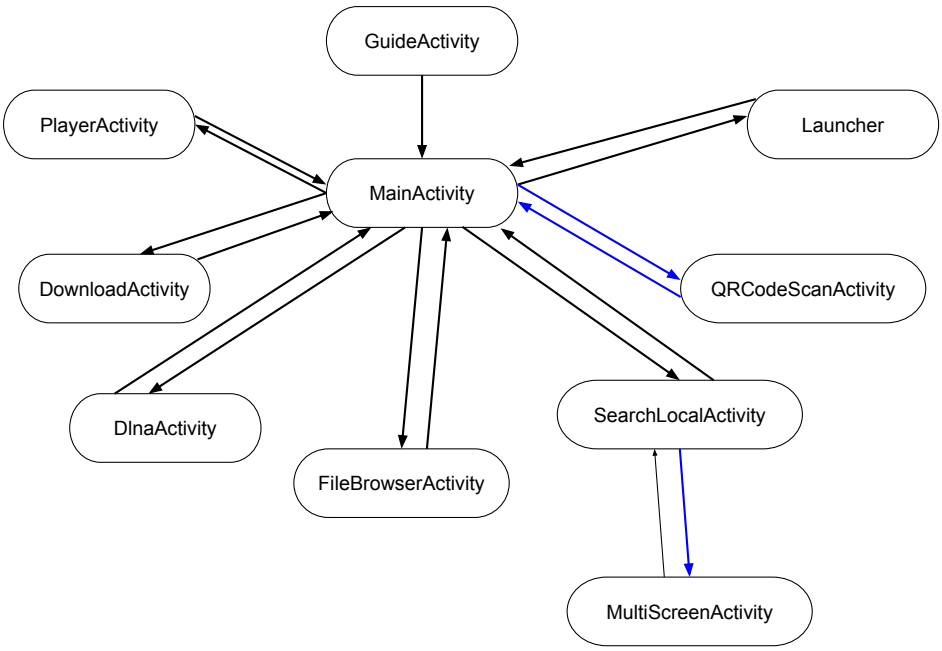

图 4.3: 应用程序“记乐部”部分GUI模型

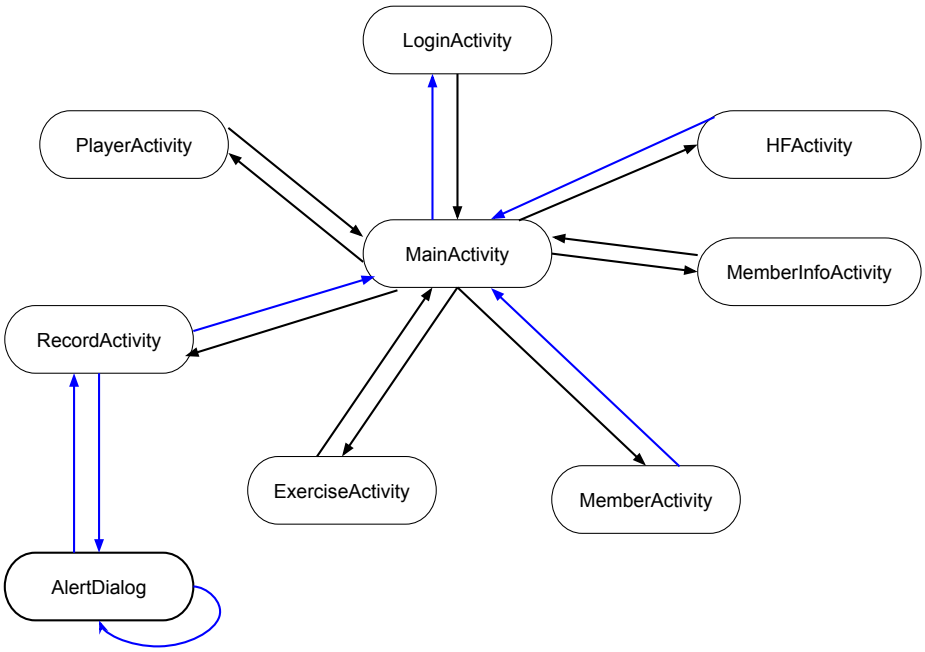

图 4.4: 应用程序“QQ影音”部分GUI模型

实验中推荐系统以Top5的方式选取兴趣度最高的5个测试任务，并设置众

包测试的兼容性测试阀值为3，即在测试过程中若某异常可复现且复现次数超过3次，则该异常被标识为“已验证异常”。开发人员可以直接根据程序运行日志处理“已验证异常”。而对于未确认的异常，测试结束后，开发人员需要首先探究异常不能完全复现的原因，选择相应的解决方案。

为了完成RQ2的回答，实验开始前，在三个应用程序项目中集成Android原生的测试框架Instrumentation和覆盖率工具Jacoco，动态统计众包工人手工测试过程中的代码覆盖率。在测试开始前，我们基于Jacoco工具，对未覆盖窗口跳转对应的起始窗口和目标窗口，完成离线插桩，使用Instrumentation框架监听测试过程中的代码执行情况，在测试中离开某个Activity窗口时，则统计该窗口覆盖代码，最后生成覆盖率报告。

4.3　实验结果预处理

4.3.1　异常数据统计

实验监听众包工人的操作事件，记录事件触发的窗口变化，收集服务器端的异常日志，这三个部分作为测试日志，每个众包工人都维护一份当前应用程序的测试日志。提取测试日志并查找对应的操作事件，实验统计三个应用程序中不重复的异常，其平均数据如表4.2所示。

表 4.2: 不重复异常平均统计数据

	简豆	记乐部	QQ影音
$\|C\|$	32.4	36.6	38.6
$\|A\|$	25.6	28.4	34.6
$\|T\|$	45.4	54.6	45

表中$|C|$代表传统众包测试过程，$|A|$代表不包含静态分析、只有自动化测试引导的众包测试机制，其中未覆盖窗口跳转部分仍采用传统众包测试的模式，$|T|$则代表该技术下的众包测试。

4.3.2　异常数据分析

为了更好地分析实验结果，我们根据表 3.3统计各个实验中的异常情况分布，不同实验中的程序异常分布情况如表 4.3所示。

我们利用折线图，更加直观地对比不同类型的异常在不同实验中的分布，了解三个应用程序不同众包测试实验中的异常类型分布变化情况，如图4.5、4.6和4.7所示。

表 4.3: 实验异常分布情况

	资源加载错误	输入意图错误	网络通信异常	ANR	线程间异常	其他
简豆|*C*|	7	10.6	0	4	5.8	5
简豆|*A*|	8.2	3.4	0	5.4	6.2	2.4
简豆|*T*|	9.6	13.8	0	6.6	8.2	7.2
记乐部|*C*|	9.2	12.6	0	5.8	4.6	4.4
记乐部|*A*|	8.8	4.2	0	5.6	6	3.8
记乐部|*T*|	9.2	24.8	0	7.4	9	4.2
QQ影音|*C*|	13.2	5.2	3.2	5.6	8	3.4
QQ影音|*A*|	12.6	4.4	0	6.6	7.4	3.6
QQ影音|*T*|	14.2	6.4	0	9.8	6.2	8.4

总结这三张图的异常变化趋势，我们可以发现：

现象1：在自动化测试触发的异常中，网络通信异常出现频率较高，但在众包测试实验中基本不出现。

现象2：在自动化测试触发的异常中，由于输入意图错误触发的异常不常出现，但在众包测试实验中经常出现。

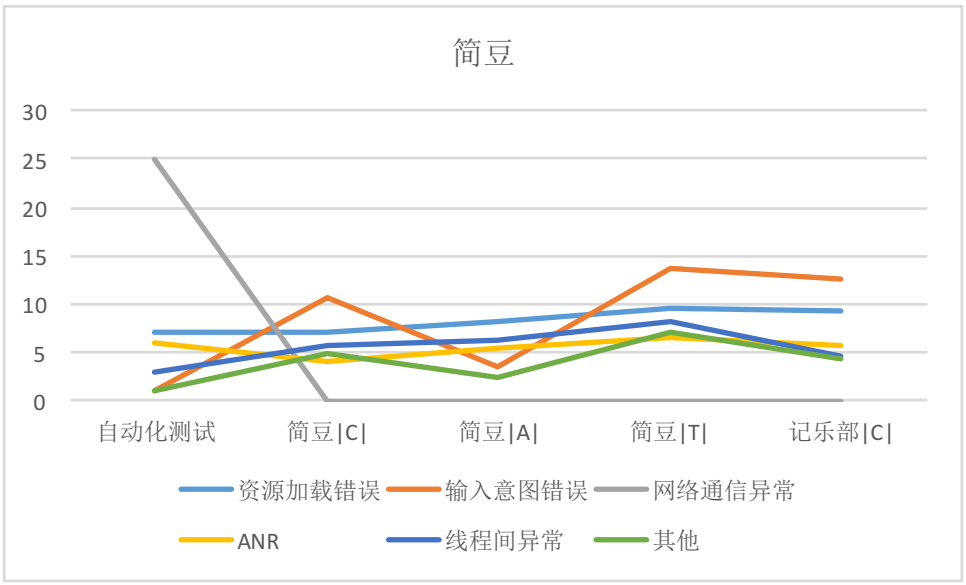

图 4.5: “简豆”在三个实验中类型异常分布变化图

我们向众包工人了解相关测试场景，解释这两个变化趋势产生的原因。

（1）现象1产生的原因：自动化测试通过在手机模拟器上运行程序，完成测试。由于电脑端手机模拟器的网络连接状态极不稳定，从而产生大量的网络通信异常，而实际实验处于网络状况良好的情况，基本不存在网络异常问题。

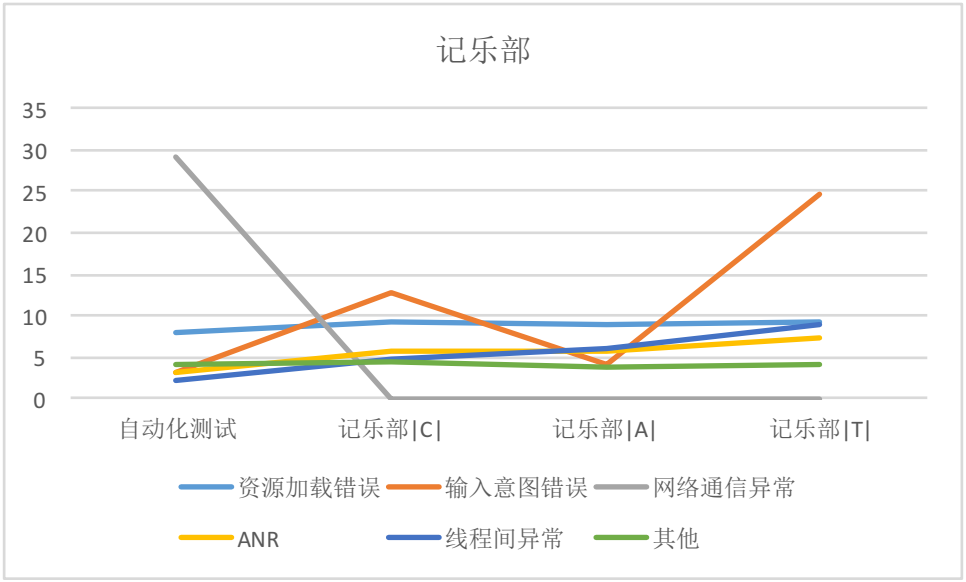

图 4.6: “记乐部”在三个实验中类型异常分布变化图

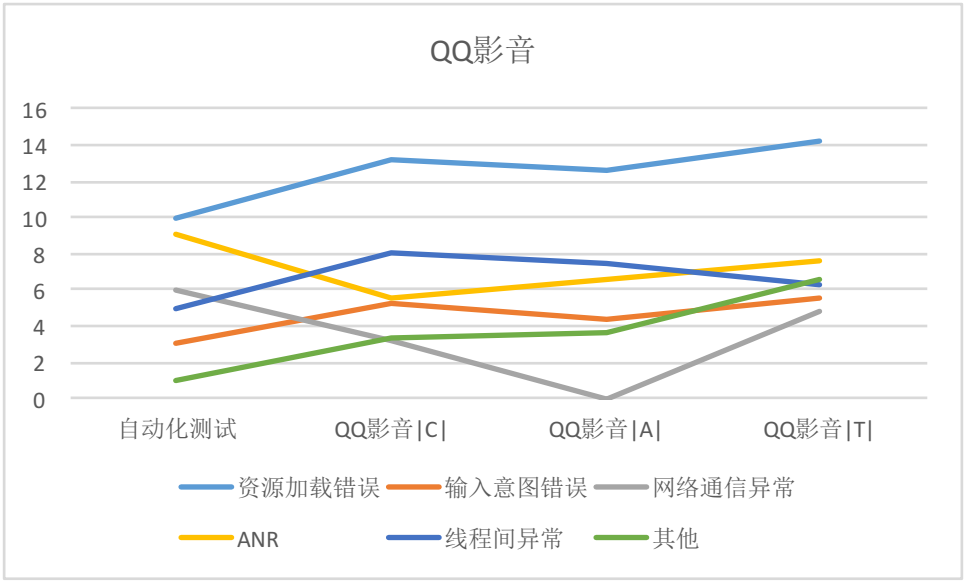

图 4.7: “QQ影音”在三个实验中类型异常分布变化图

而QQ影音|C|实验中产生的3.2个网络异常，是因为众包工人在测试“视频下载”功能时，关闭网络而导致。

（2）现象2产生的原因：自动化测试以人工测试脚本为辅助，在遇到需要输入的控件时，主动去人工测试脚本中查找相关参数，完成输入事件，实现组件的遍历。在测试脚本中，每个输入事件对应其中一个输入参数。而在实验过程中，众包工人往往通过多种输入内容、多种内容格式，实现输入事件的边界

值测试、非常规字符输入等多种测试，所以自动化测试中的输入意图错误远低于实验中的输入意图错误。

综合上述分析，不考虑网络异常的情况下，我们可以得出以下结论：无论是传统众包测试还是包含引导的众包测试，众包测试都比自动化测试的结果更加全面，众包测试是一种有效的移动应用测试方式。

4.4　实验结果分析

4.4.1　测试质量评估

以相同测试时间内触发的异常数代表众包测试效率，测试开展相同时间时，应用程序被触发的异常数目越多，则该实验的众包测试效率更高。

我们从两个方面回答RQ1：众包测试效率以及已验证测试异常数目。

记录三个应用分别在实验|*C*|和实验|*T*|中的程序运行情况，每隔一分钟统计当前众包工人已经发现的不重复异常数目，得到三个应用的异常数目随时间变化图，如图4.8，4.9，4.10所示。

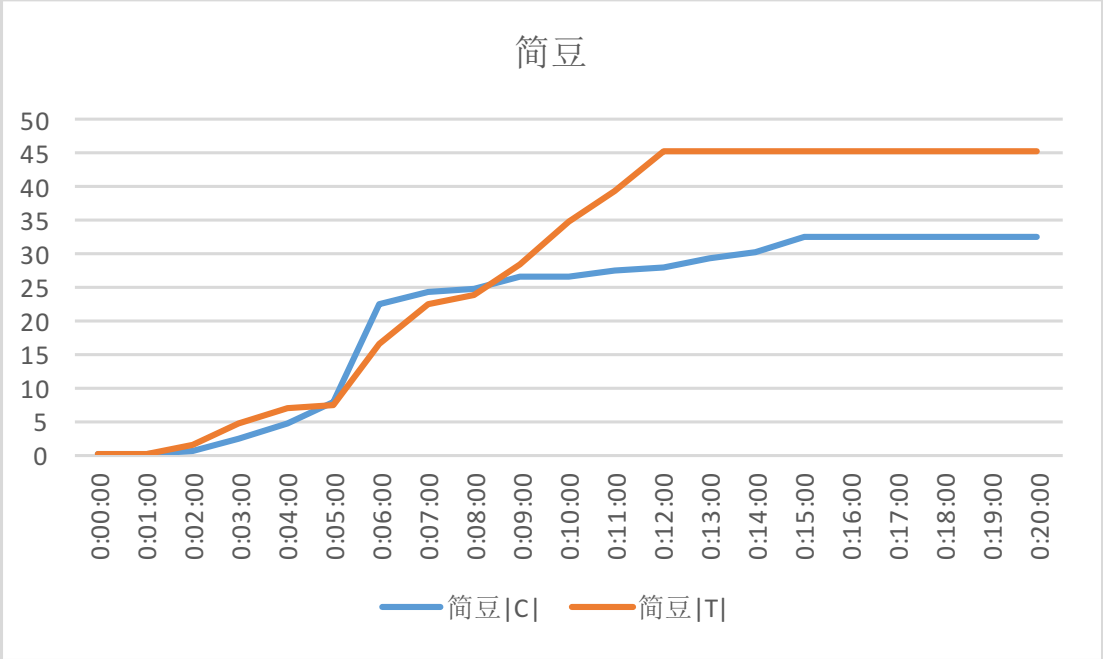

图 4.8: “简豆”异常数目随时间变化图

从三个图的整体趋势来看，程序分析引导下的众包工人，能够更快、更多地触发应用程序异常。在实验中，我们设置每次众包测试的开展时间为20分钟，测试完成后，实验|*T*|中触发的程序异常数目比实验|*C*|平均提升28.34%。值得注

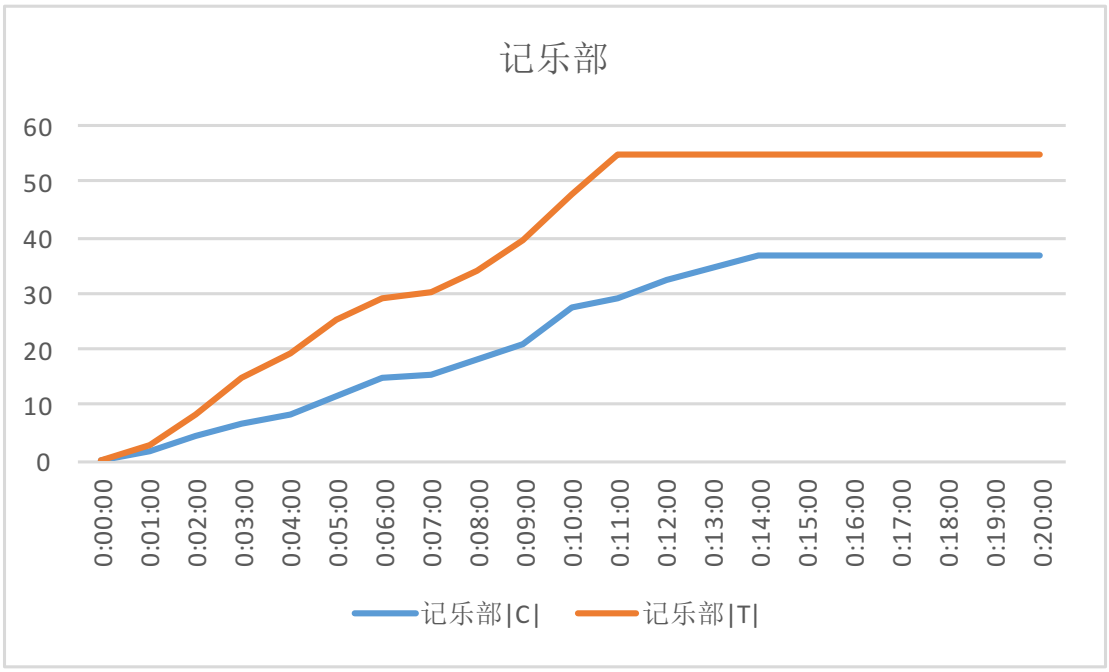

图 4.9: “记乐部”异常数目随时间变化图

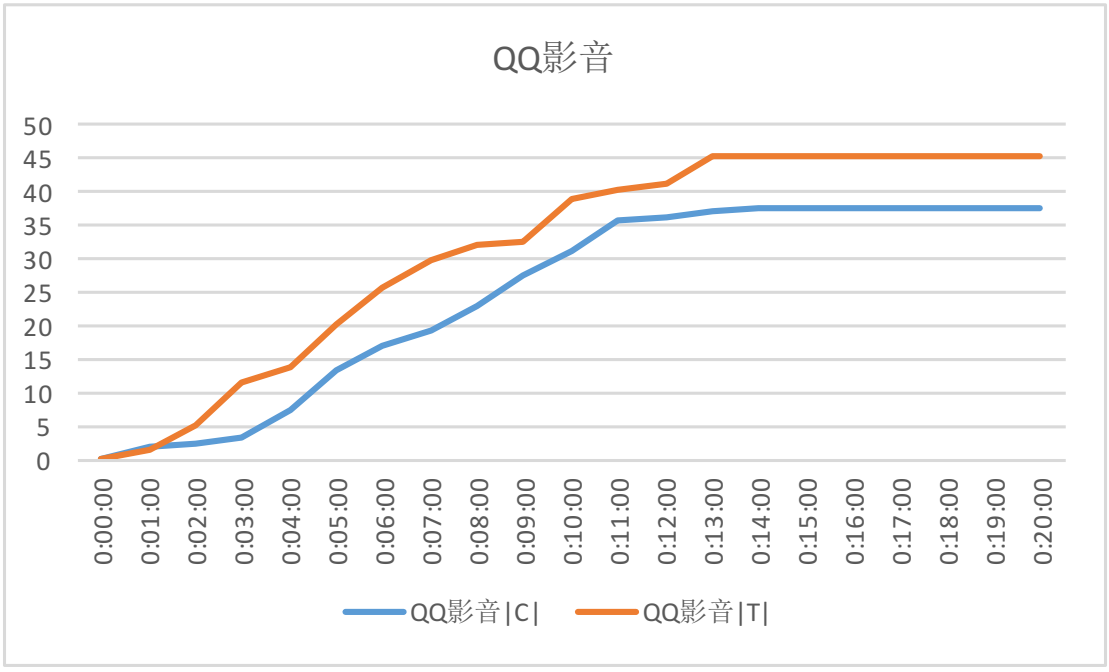

图 4.10: “QQ影音”异常数目随时间变化图

意的是，在图4.8里，应用程序“简豆”的异常变化图中，从第5分钟到第8分钟，实验|*C*| 触发的异常数高于实验|*T*|，我们调查当时的应用程序场景，发现这是由于应用程序“简豆”出现严重异常：“图书”资源获取失败，列表不能加载图书

列表，如图4.11所示。对于这种明显的功能异常，众包工人能够更快地发现异常，传统众包测试表现更好。

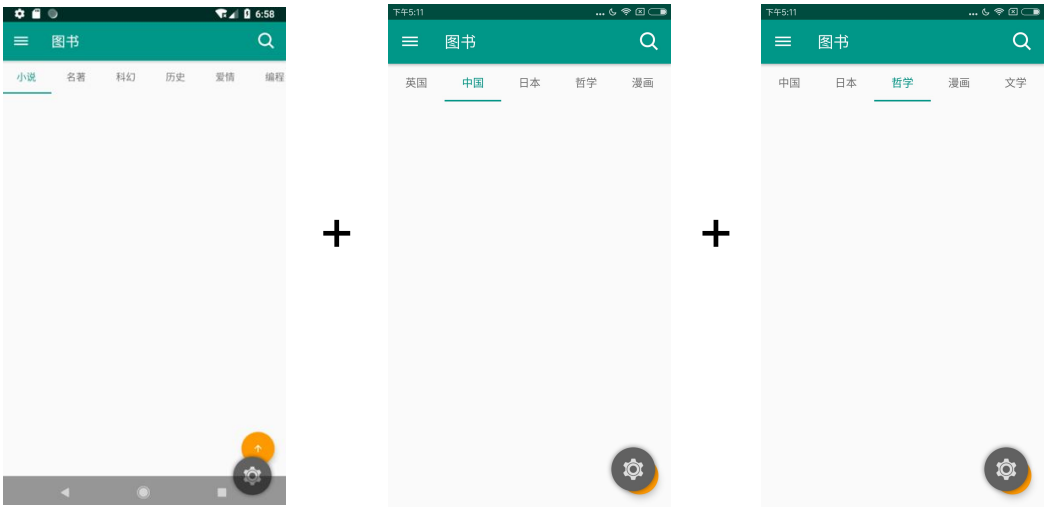

图 4.11: 应用程序“简豆”的图书列表加载失败

仅仅根据众包测试效率对比并不能完全说明该技术对众包测试质量的有效提升。众包测试开展的最终目的是为了向开发人员反馈程序异常、帮助开发人员解决异常。开发人员往往根据测试步骤的描述还原异常发生场景，了解异常产生原因。面对来自传统众包测试的大量未处理测试数据，开发人员将浪费大量的时间确认重复异常。针对这样的现象，在测试任务划分中，我们引入“已验证异常”的概念，在测试过程中了解不同异常的复现情况，实时调度测试任务的分派，防止众包资源的浪费。我们统计实验$|C|$和实验$|T|$中不同异常的复现次数，如图4.12，探讨“已验证异常”机制的合理性。

在众包工人数目为5，复现超过3次被认为“已验证异常”的众包测试中，我们分别统计复现1-3次、4-5次以及5次以上的异常数目。对于复现1-3的异常，该异常尚不能被确认，测试结束后开发人员需要研究不能完全复现的原因，复现4-5次的异常被认为“已验证异常”，在测试过程中可以同步给开发人员解决，而复现超过5次的异常，即该异常被每个众包工人测试超过1次，则被认为是众包资源的浪费。在统计中我们发现:

a. 测试过程中出现的所有线程类异常，被复现的次数都不超过3次，这是由于不同手机内存、手机当前在后台运行的程序决定的，该类异常很难被复现，导致实验过程中，复现1-3次的异常数目偏多。

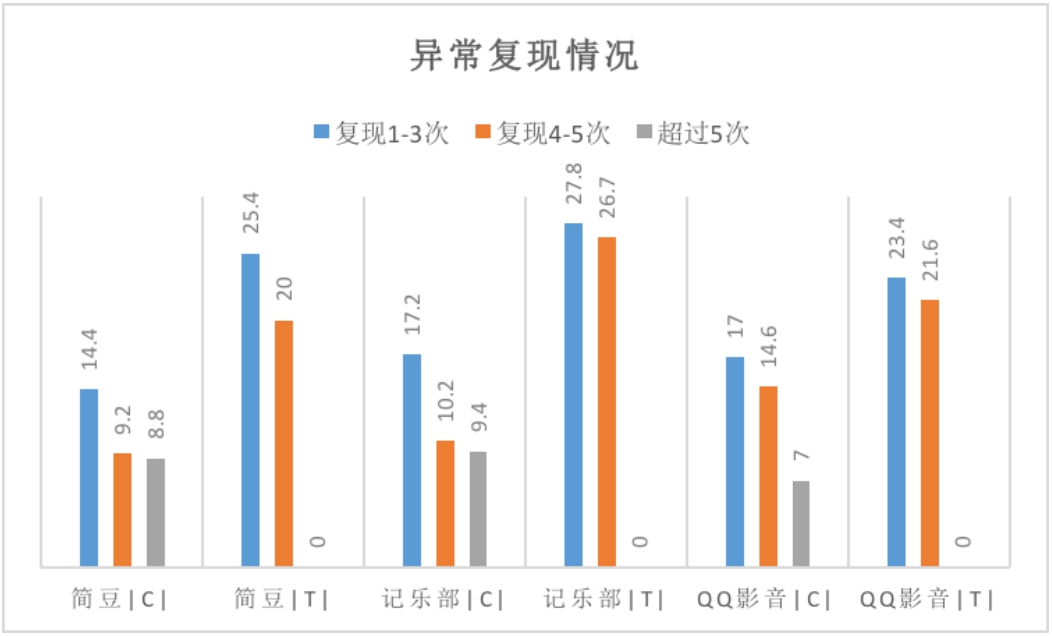

图 4.12: 实验异常复现情况统计图

b. 实验|*C*|中，即传统众包测试下，通过分析测试日志，我们发现这些众包工人没有明确的测试目标，存在很多的重复测试状态，导致同一异常的多次触发，实验|*C*|中普遍存在复现超过5次的异常。这种无监管的测试流程无疑是一种时间和众包工人测试精力的浪费。

c. 和实验|*C*|相比，实验|*T*|平均减少56.24%的未验证异常数目，这是对开发人员的一个重大帮助，极大减轻开发人员的修复成本。

因此，我们可以得出结论，“已验证异常”机制是有效的众包测试优化策略，能够优化众包资源配置。

从测试效率和“已确认异常”的分析，我们可以得出回答RQ1：本文提出的众包技术能有效提升众包测试质量。

4.4.2　代码覆盖率统计

为了回答RQ2，我们首先统计这三个应用程序未覆盖窗口比例，如表4.4所示。从未覆盖窗口的平均比例来看，自动化测试仍存在很多未能遍历的窗口，由此可见，自动化测试对应用程序的测试并不完全，因此，众包工人对未覆盖窗口的探索工作是有必要的。

其次，我们对比实验|*T*|与实验|*A*|，统计未覆盖窗口跳转中起始窗口和目标

表 4.4: 未覆盖窗口比例统计

简豆	Q记乐部	QQ影音	AVERAGE
10/22	3/17	7/17	20/56

窗口的平均代码覆盖率，验证引导机制对于未覆盖窗口跳转的有效性。

表 4.5: 应用程序“简豆”未覆盖窗口的代码覆盖率

元素	简豆$\|A\|$覆盖率	简豆$\|T\|$覆盖率
BookDetailsActivity	5.4/12	5/12
ActorDetailsActivity	5.8/12	6/12
AlertDialog	0.2/6	4/6
TOTAL	10.2/30	15/30

表 4.6: 应用程序“记乐部”未覆盖窗口的代码覆盖率

元素	记乐部$\|A\|$覆盖率	记乐部$\|T\|$覆盖率
RecordActivity	7.2/10	6/10
AlertDialog	5/5	5/5
MemberActivity	5.4/6	5.2/6
MemberInfoActivity	3/5	3.2/5
HFActivity	5/5	5/5
TOTAL	25.6/31	23.4/31

表 4.7: 应用程序“QQ影音”未覆盖窗口的代码覆盖率

元素	QQ影音$\|A\|$覆盖率	QQ影音$\|T\|$覆盖率
SearchLocalDevicesActivity	0.8/5	5/5
QRCodeScanActivity	0.4/2	2/2
MultiscreenHelpActivity	0.4/10	8/10
TOTAL	1.6/17	15/17

我们统计和计算未覆盖窗口在不同实验下的平均代码覆盖率，实验数据显示，对于未覆盖窗口跳转部分，与不包含静态分析的众包测试相比，本文提出的技术平均提升20.51%的代码覆盖率。

详细比较不同窗口间的代码覆盖率，我们发现三个应用程序的代码覆盖率并没有都得到提高，其中只有两个应用程序的代码覆盖率得到一定的提升，而甚至在一些Activity中，实验$|A|$的代码覆盖率超过实验$|T|$。我们结合众包工人的操作事件分析产生原因：

图 4.13: 应用程序“QQ影音”MainActivity对应的窗口

（1）实验|*A*|中的部分Activity代码覆盖率超过实验|*T*|：本文提出的技术以静态分析的未覆盖窗口引导众包工人探索异常，即提示众包工人自动化测试未覆盖的窗口跳转，众包工人根据提示进入相应页面，探索异常的过程本质为传统众包测试过程。若实验|*A*|中的众包工人也能进入到相关页面，则实验|*A*|和实验|*T*|都是对未覆盖窗口的传统众包测试，其测试结果存在不确定性，实验|*A*|中可能存在部分Activity的代码覆盖率超过实验|*T*|。

（2）实验|*A*|的部分Activity覆盖率过低：一个Activity可能对应多个页面，如图4.13所示，应用程序“QQ影音”的MainActivity窗口实际对应3个应用页面，用户可以通过下方按钮切换。在没有静态分析提示的时候，众包工人会存在对下方按钮的忽略情况，只完成一个页面的测试，未能对其他页面的组件进行测试探索。而我们使用静态分析划分的测试任务，虽然不能明确向众包工人指出执行点击按钮事件、实现页面切换的目的，但是众包工人查看提示后会主动寻找如何进入相关页面，探索更多的异常。

基于以上对于代码覆盖率的分析，我们可以得出这样的结论，在只包含自动化测试引导的众包测试中，众包工人极有可能会忽略一些页面，而本文提出的众包测试技术借助静态分析的结果，向众包工人明确未探索窗口列表，防止众包工人对页面的忽略，提高整体应用程序的测试覆盖率。

4.4.3　推荐系统满意度

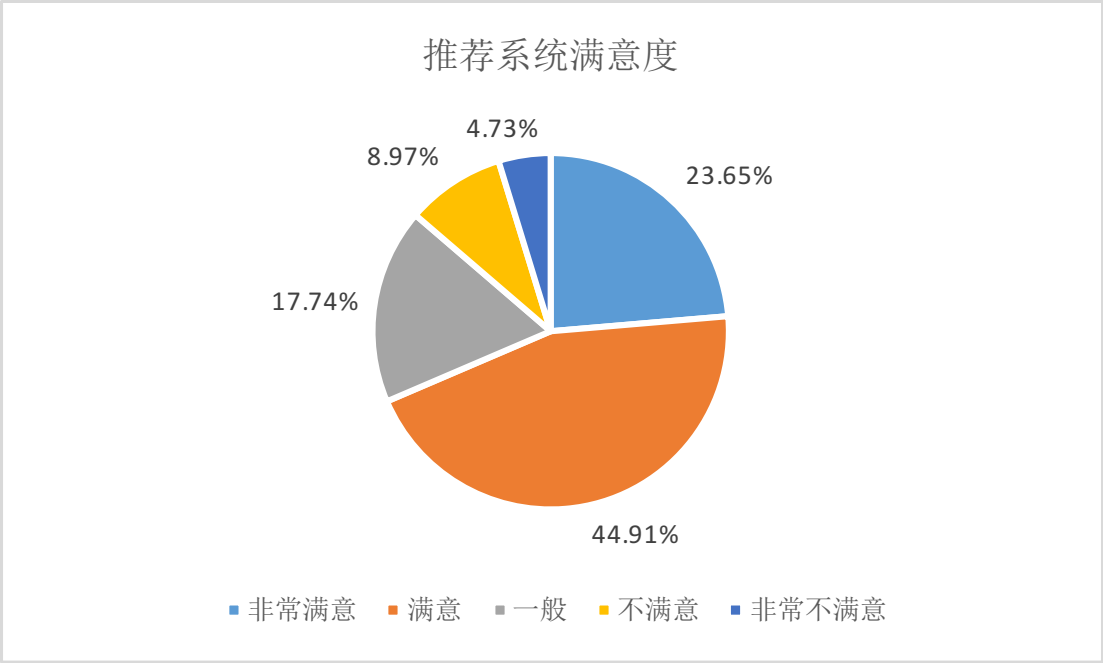

图 4.14: 推荐系统满意度

为了回答RQ3，我们对实验|*A*|和实验|*T*|中的众包工人进行关于测试任务推荐系统的问卷调查。问卷内容包括众包工人的专业程度、对推荐系统的了解程度、对冷启动阶段测试用例的多样性评估以及对测试任务推荐平台的满意度。我们收到30份来自问卷，其中参与实验的众包工人主要分布于大四到研三之间，都来自软件工程专业，对推荐系统的有一定的了解。收集这样一批专业素养较高的众包工人对我们测试系统的评价，其统计结果具有更高的参考价值，推荐系统满意度结果如图4.14所示：

从统计结果发现，绝大多数众包工人对测试任务推荐系统还是持有积极的态度，少部分众包工人对测试任务推荐系统不是十分满意，我们采访并调查这部分众包工人的评分原因，总结如下：

1. 未覆盖窗口跳转推荐系统的数据更新不及时。

2. 在众包测试过程中新触发的异常没有加入到测试用例推荐系统中。

针对问题1，由于未覆盖窗口跳转的数量偏少，三个应用程序的未覆盖窗口跳转平均数量不超过10，推荐列表会存在重复的现象。而问题2的提出也是我们下面工作的方向，将众包测试过程中新触发的异常加入测试用例，引导其他众包工人验证该异常，形成迭代式的众包测试机制。

4.5　有效性威胁

本节我们讨论一些可能影响实验结果的因素：

（1）众包工人的选取。参与实验的众包工人是来自软件学院的学生，主要分布于大四到研三之间。这和一般的众包测试不同，一般众包测试中的众包工人来自不同背景，我们实验中的众包工人背景则相对狭窄，可能对实验结果产生影响。Salman等人 [59]研究以学生作为专业测试人员的可行性，结果表示，在明确描述测试任务的情况下，学生和专业测试人员的能力具有一定的相似性。因此，学生作为众包工人，其专业素养远超一般众包工人，对于众包测试来说，选用学生可以提高众包测试效率。

（2）应用程序的选取。本文提出的技术以Android SDK的方式集成于应用程序的源码中，因此，选择实验的应用程序还需要来自企业界的配合。同时，目前关于Android GUI静态分析的工作较少，现有工具中的静态分析工具GATOR仍不够稳定，因此，选择一个可以满足实验的应用程序需要花费较大精力。本次实验我们只选取3个应用程序，但是根据自动化测试分析，这些应用程序的平均异常数超过30，最重要的是，这些异常类型多样，能够满足我们的实验要求，同时，在分析实验的同时，这些多样化的异常也给我们带来一定的改进启发。

4.6　本章小结

在这一章，我们讨论对基于动静态程序分析结合的移动众包引导技术的评估实验，包括两组对比实验的市值以及众包工人对测试任务推荐系统的满意度调查。在此设计下，我们介绍该技术的实际使用过程，开展传统众包测试、只包含自动化测试引导的众包测试以及该技术下的众包测试三组实验，收集测试结果，我们以测试效率和已验证异常验证该技术对传统众包测试质量的提升。在Jacoco和Instrumentation的辅助下，我们讨论未覆盖窗口跳转对应窗口的代码覆盖率，验证GUI静态分析的必要性。而对于测试任务推荐系统，我们采用问卷调查的方式了解众包工人的反馈，并收集来自众包工人的建议。

第五章　总结与展望

5.1　总结

移动互联网飞速发展带来基于安卓系统的第三方应用程序的快速发展，Android应用程序以“短平快”的开发模式快速涌现，覆盖移动互联网产业链各方面的需求。这样的快速发展，也对安卓应用程序的质量提出更高的要求。和传统Web应用程序不同，安卓App面临着多样化的运行环境，传统的人工测试已经不能满足移动应用测试。众包的思想被提出运用于解决移动应用测试：众包平台将测试任务分发给广大的测试用户，大量用户反馈测试过程中，真实出现的异常信息，实现全面且充分的产品测试。但是众包机制其本身存在测试流程无监管、众包工人无门槛的特点，众包测试质量亟待提升。因此，有效地管理众包测试过程，引导没有经验的众包工人熟悉测试流程，可以从源头上提升众包测试质量，优化众包测试机制。

本文基于群体智能的思想，创新性地提出一种改进众包测试机制的方案，将动态程序分析中的自动化测试、静态程序分析与众包测试相结合，以程序分析的结果引导众包测试的开展。根据自动化测试日志提取异常的触发场景，该技术以此为主要测试任务，引导众包工人在不同环境下验证异常。对比自动化测试和静态分析的GUI变化情况，该技术提取自动化测试未能覆盖的窗口跳转，引导众包工人探索静态分析补充的窗口跳转中，可能存在的新异常。摈弃传统人工划分测试任务的方式，该技术首次提出基于程序分析的结果完成测试任务的专业化设计。同时，我们根据众包工人的历史测试行为，将测试任务推荐给具有类似测试经验的众包工人，完成众包资源的配置优化。该技术主动筛选已被测试过多次的测试任务，引导众包工人之间形成协作关系，努力实现“所有测试任务都得到充分测试”的目标，优化众包资源配置。最后，当众包工人选择一个测试任务，该方法根据不同的测试任务类型，基于GUI模型，计算众包工人从当前窗口到达测试任务起始窗口的最短路径，生成测试事件序列，以截图标注和文本信息相结合的方式，形象而直观地向众包工人提供对应测试事件的引导。在测试过程中以这种方式完成对众包工人的培训，提高众包工人的专业素养与众包测试的质量，减轻后期的处理与分析工作。我们在“简豆”、“QQ影音”以及“记乐部”这三个应用程序上开展实验，收集测试数据并分别从测试效率、异常覆盖率以及众包工人对测试任务推荐系统的反馈这三个方面，证明我们的引导机制能够有效地提升众包测试质量。

5.2 展望

本文提出的引导机制还有很多优化和改进的部分。

（1）该技术基于测试用例被选择的次数、而非异常被触发的次数完成协作式众包测试，为了实现更充分的兼容性测试，在下面的工作中，我们将会实现高并发高效率的测试监听机制，实现对异常信息的实时监听与处理。

（2）基于测试监听机制，我们可以将众包工人在测试过程中触发的新异常加入测试用例集，引导其他众包工人验证新异常，实现迭代式的众包测试引导机制，形成高可用、高质量的测试生态系统。

（3）我们以窗口跳转为单位，计算众包工人的测试路径，忽略自动化测试在当前页面的前置测试事件。为了实现更加准确的异常复现机制，下面的工作将研究如何准确还原自动化测试的事件序列，提取异常的前置测试事件，这是准确定位异常的工作重点。

（4）对于“已验证异常”，我们为每一个测试用例设置相同的测试阈值作为判断标准，但是在实际生产中，为了进一步提高众包测试效率，我们可以根据异常属性，为不同类型的异常赋予不同的测试阈值。

参考文献

[1] P. S. Kochhar, F. Thung, N. Nagappan, T. Zimmermann, D. Lo, Understanding the test automation culture of app developers, in: 2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST), IEEE, 2015, pp. 1–10.

[2] C. Hu, I. Neamtiu, Automating gui testing for android applications, in: International Workshop on Automation of Software Test, 2011.

[3] Y.-M. Baek, D.-H. Bae, Automated model-based android gui testing using multi-level gui comparison criteria, in: Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, ACM, 2016, pp. 238–249.

[4] 李乔, 柯栋梁, 王小林, 云测试研究现状综述, Ph.D. thesis (2012).

[5] 曹咏春, 刘小君, 云测试综述, Ph.D. thesis (2011).

[6] 夏恩君, 赵轩维, 李森, 国外众包研究现状和趋势, 技术经济1 (2015) 28–36.

[7] 张利斌, 钟复平, 涂慧, 众包问题研究综述, 科技进步与对策29 (6) (2012) 154–160.

[8] M. Nebeling, M. Speicher, M. Grossniklaus, M. C. Norrie, Crowdsourced web site evaluation with crowdstudy, in: International Conference on Web Engineering, Springer, 2012, pp. 494–497.

[9] 章晓芳, 冯洋, 刘峘, 陈振宇, 徐宝文, 众包软件测试技术研究进展, 软件学报 (1) (2018) 4.

[10] N. J. Nilsson, Principles of artificial intelligence, Morgan Kaufmann, 2014.

[11] Y. Pan, Heading toward artificial intelligence 2.0, Engineering 2 (4) (2016) 409–413.

[12] 王玫, 朱云龙, 何小贤, 群体智能研究综述, 计算机工程31 (22) (2005) 194–196.

[13] A. Machiry, R. Tahiliani, M. Naik, Dynodroid: An input generation system for android apps, in: Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, ACM, 2013, pp. 224–234.

[14] T. M. U. android testing tool, http://developer.android.com/tools/help/monkey.html (2019).

[15] D. Amalfitano, A. R. Fasolino, P. Tramontana, S. De Carmine, A. M. Memon, Using gui ripping for automated testing of android applications, in: Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering, ACM, 2012, pp. 258–261.

[16] W. Choi, G. Necula, K. Sen, Guided gui testing of android apps with minimal restart and approximate learning, in: Acm Sigplan Notices, Vol. 48, ACM, 2013, pp. 623–640.

[17] S. Hao, B. Liu, S. Nath, W. G. Halfond, R. Govindan, Puma: programmable ui-automation for large-scale dynamic analysis of mobile apps, in: Proceedings of the 12th annual international conference on Mobile systems, applications, and services, ACM, 2014, pp. 204–217.

[18] R. Mahmood, N. Mirzaei, S. Malek, Evodroid: Segmented evolutionary testing of android apps, in: Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, ACM, 2014, pp. 599–609.

[19] S. Anand, M. Naik, M. J. Harrold, H. Yang, Automated concolic testing of smartphone apps, in: Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering, ACM, 2012, p. 59.

[20] S. R. Choudhary, A. Gorla, A. Orso, Automated test input generation for android: Are we there yet?(e), in: 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE), IEEE, 2015, pp. 429–440.

[21] B. Jiang, Y. Zhang, W. Chan, Z. Zhang, Which factor impacts gui traversal-based test case generation technique most? a controlled experiment on android applications, in: 2017 IEEE International Conference on Software Quality, Reliability and Security (QRS), IEEE, 2017, pp. 21–31.

[22] S. Yang, H. Wu, H. Zhang, Y. Wang, C. Swaminathan, D. Yan, A. Rountev, Static window transition graphs for android, Automated Software Engineering 25 (4) (2018) 833–873.

[23] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau, P. McDaniel, Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps, Acm Sigplan Notices 49 (6) (2014) 259–269.

[24] L. Li, A. Bartel, T. F. Bissyandé, J. Klein, Y. Le Traon, S. Arzt, S. Rasthofer, E. Bodden, D. Octeau, P. McDaniel, Iccta: Detecting inter-component privacy leaks in android apps, in: Proceedings of the 37th International Conference on Software Engineering-Volume 1, IEEE Press, 2015, pp. 280–291.

[25] Y. Wang, H. Zhang, A. Rountev, On the unsoundness of static analysis for android guis, in: Proceedings of the 5th ACM SIGPLAN International Workshop on State Of the Art in Program Analysis, ACM, 2016, pp. 18–23.

[26] J. Howe, The rise of crowdsourcing, Wired magazine 14 (6) (2006) 1–4.

[27] K. Mao, L. Capra, M. Harman, Y. Jia, A survey of the use of crowdsourcing in software engineering, Journal of Systems and Software 126 (2017) 57–84.

[28] K.-T. Chen, C.-C. Wu, Y.-C. Chang, C.-L. Lei, A crowdsourceable qoe evaluation framework for multimedia content, in: Proceedings of the 17th ACM international conference on Multimedia, ACM, 2009, pp. 491–500.

[29] D. Liu, R. G. Bias, M. Lease, R. Kuipers, Crowdsourcing for usability testing, Proceedings of the American Society for Information Science and Technology 49 (1) (2012) 1–10.

[30] S. Komarov, K. Reinecke, K. Z. Gajos, Crowdsourcing performance evaluations of user interfaces, in: Proceedings of the SIGCHI conference on human factors in computing systems, ACM, 2013, pp. 207–216.

[31] C. telemetry, http://www.chromium.org/developers/telemetry (2019).

[32] M. LYNC, http://office.microsoft.com/lync (2019).

[33] F. telemetry, https://telemetry.mozilla.org (2019).

[34] A. I. Khan, Z. Al-Khanjari, M. Sarrab, Crowd sourced testing through end users for mobile learning application in the context of bring your own device, in: 2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), IEEE, 2016, pp. 1–6.

[35] M. V. Mäntylä, J. Itkonen, More testers–the effect of crowd size and time restriction in software testing, Information and Software Technology 55 (6) (2013) 986–1003.

[36] J. Wang, S. Wang, Q. Cui, Q. Wang, Local-based active classification of test report to assist crowdsourced testing, in: 2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE), IEEE, 2016, pp. 190–201.

[37] Y. Feng, J. A. Jones, Z. Chen, C. Fang, Multi-objective test report prioritization using image understanding, in: 2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE), IEEE, 2016, pp. 202–213.

[38] R. Hao, Y. Feng, J. A. Jones, Y. Li, Z. Chen, Ctras: Crowdsourced test report aggregation and summarization, in: Proceedings of the 41th International Conference on Software Engineering, ACM, 2019.

[39] N. Quoc Viet Hung, D. Chi Thang, M. Weidlich, K. Aberer, Erica: Expert guidance in validating crowd answers, in: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2015, pp. 1037–1038.

[40] H.-L. Xu, X. Wu, X.-D. Li, B.-P. Yan, Comparison study of internet recommendation system, Journal of software 20 (2) (2009) 350–362.

[41] M. J. Pazzani, A framework for collaborative, content-based and demographic filtering, Artificial intelligence review 13 (5-6) (1999) 393–408.

[42] J. S. Breese, D. Heckerman, C. Kadie, Empirical analysis of predictive algorithms for collaborative filtering, in: Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence, Morgan Kaufmann Publishers Inc., 1998, pp. 43–52.

[43] J. L. Herlocker, J. A. Konstan, L. G. Terveen, J. T. Riedl, Evaluating collaborative filtering recommender systems, ACM Transactions on Information Systems (TOIS) 22 (1) (2004) 5–53.

[44] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions, IEEE Transactions on Knowledge & Data Engineering (6) (2005) 734–749.

[45] Z. Huang, D. Zeng, H. Chen, A comparison of collaborative-filtering recommendation algorithms for e-commerce, IEEE Intelligent Systems 22 (5) (2007) 68–78.

[46] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: International Conference on World Wide Web, 2001.

[47] A. L. Deng, Y. Y. Zhu, B. L. Shi, A collaborative filtering recommendation algorithm based on item rating prediction, Journal of Software 14 (9) (2003) 54–65.

[48] B. McFee, L. Barrington, G. R. Lanckriet, Learning similarity from collaborative filters., in: ISMIR, 2010, pp. 345–350.

[49] G. Karypis, Evaluation of item-based top-n recommendation algorithms, in: Proceedings of the tenth international conference on Information and knowledge management, ACM, 2001, pp. 247–254.

[50] J. Elseberg, S. Magnenat, R. Siegwart, A. Nüchter, Comparison of nearest-neighbor-search strategies and implementations for efficient shape registration, Journal of Software Engineering for Robotics 3 (1) (2012) 2–12.

[51] B. Lika, K. Kolomvatsos, S. Hadjiefthymiades, Facing the cold start problem in recommender systems, Expert Systems with Applications 41 (4) (2014) 2065–2073.

[52] P. Cremonesi, Y. Koren, R. Turrin, Performance of recommender algorithms on top-n recommendation tasks, in: Proceedings of the fourth ACM conference on Recommender systems, ACM, 2010, pp. 39–46.

[53] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, L. Schmidt-Thieme, Learning attribute-to-feature mappings for cold-start recommendations., in: ICDM, Vol. 10, Citeseer, 2010, pp. 176–185.

[54] S. Yang, D. Yan, H. Wu, Y. Wang, A. Rountev, Static control-flow analysis of user-driven callbacks in android applications, in: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, Vol. 1, IEEE, 2015, pp. 89–99.

[55] S. Yang, D. Yan, H. Wu, Y. Wang, A. Rountev, Static control-flow analysis of user-driven callbacks in android applications, in: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, Vol. 1, IEEE, 2015, pp. 89–99.

[56] M. K. Heinemann, The matthew effect., Thoracic And Cardiovascular Surgeon 64 (02) (2016) 087–087.

[57] W. Hao, Z. Wang, W. Zhang, Quantitative analysis of matthew effect and sparsity problem of recommender systems, in: IEEE International Conference on Cloud Computing And Big Data Analysis, 2018.

[58] Anderson, The long tail: Why the future of business is selling less of more, Hyperion 24 (3) (2006) 274–276.

[59] I. Salman, A. T. Misirli, N. Juristo, Are students representatives of professionals in software engineering experiments?, in: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, Vol. 1, IEEE, 2015, pp. 666–676.

简历与科研成果

基本情况 张欣，女，汉族，1994 年 9 月出生，江苏省扬州市人。

教育背景

2016.9～2019.6 南京大学软件学院 硕士

2012.9～2016.7 大连理工大学软件学院 本科

1. **Zhang X**，Chen Z，Fang C，et al.“Guiding the crowds for Android testing[C]// Proceedings of the 38th International Conference on Software Engineering Companion. ACM, 2016: 752-753.”
2. 陈振宇，**张欣**，房春荣，张智轶“一种基于步骤提示的安卓众包测试”，申请号：201711200460.9，已受理。
3. 房春荣，**张欣**，刘嘉，李玉莹，陈振宇“一种基于群体智能的移动应用测试报告提交及处理方法”，申请号：201810092970.7，已受理。
4. 国家自然科学基金项目：基于可理解信息融合的人机协同移动应用测试研究（61802171）, 2019-2021
5. 中央高校基本科研业务费专项资金资助：4-3基于可理解信息融合的人机协同移动应用测试研究，021714380017，2019.4-2019.12

致　　谢

时光荏苒，三年的研究生求学生活即将结束，站在毕业的门槛上回忆这三年，我向所有关心、爱护和帮助过我的人们表示最诚挚的感谢。

首先，我要感谢我的导师陈振宇老师和房春荣老师。三年以来，陈振宇老师渊博的专业知识、严谨的科研态度、永不懈怠的工作激情以及幽默风趣的人格魅力给我留下了深刻的印象，引领着我完成三年的学术研究。房春荣老师创新的思路和细致踏实的学术态度让我受益良多，在毕业论文完成期间，从研究方向、项目最初计划以及后续的实施过程中，他的建议总是让我茅塞顿开，毕业论文倾注了他大量的心血。

感谢实验室的同学们，在遇到困难时你们总会伸出援手，和你们一起讨论学术、工作、生活的时候总是轻松快乐。感谢郝蕊、邹卫琴、时清凯、冯洋等很多师兄师姐们，你们一直是我前进的方向和努力的标杆。感谢同届的李玉莹，你是我的宝藏女孩，是我三年来快乐与忧愁的共同见证与分担者。感谢实验室后来加入的学弟学妹们，可爱的你们给我带来很多欢乐。在你们的陪伴下，三年生活异常精彩。

最后也是最重要的，谢谢我的爸爸、妈妈、郭玉晨，是你们在我失意的时候陪伴着我，使我有了再次前进的动力，你们永远是我坚实的后盾和温馨的港湾。

求学生涯暂时告一段落，但求知的道路永无止境。三年的求学生活带给我很多珍贵的财富，更教会我许多难能的品质，我将带着这三年沉甸甸的收获，砥砺前行，不忘初心。

版权及论文原创性说明

任何收存和保管本论文的单位和个人，未经作者本人授权，不得将本论文转借他人并复印、抄录、拍照或以任何方式传播，否则，引起有碍作者著作权益的问题，将可能承担法律责任。

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含其他个人或集体已经发表或撰写的作品成果。本文所引用的重要文献，均已在文中以明确方式标明。本声明的法律结果由本人承担。

作者签名：

日期：　年　月　日