南京大學

研 究 生 毕 业 论 文

（申请工程硕士学位）

论 文 题 目 安卓自动化测试驱动的众包需求生成系统的设计与实现

作 者 姓 名 韦志宾

学科、专业名称 工程硕士（软件工程领域）

研 究 方 向 软件工程

指 导 教 师 陈振宇 教授 房春荣 助理研究员

2020 年 5 月 26 日

学　　号：MF1832173

论文答辩日期：2020 年 5 月 23 日

指 导 教 师：　　（签字）

The Design and Implementation of Crowdsourced Testing Requirement Generation System Driven by Android Automatic Testing

By

Zhibin Wei

Supervised by

Professor **Zhenyu Chen**

Research Assistant **Chunrong Fang**

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 2020

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：安卓自动化测试驱动的众包需求生成系统的设计与实现

工程硕士（软件工程领域） 专业 2018 级硕士生姓名： 韦志宾

指导教师（姓名、职称）： 陈振宇 教授 房春荣 助理研究员

摘 要

随着移动互联网的高速发展，移动应用的质量也越来越受到重视。为了解决移动应用快速迭代下的测试难题，自动化测试和众包测试在移动应用测试领域得到了广泛的应用。然而，自动化测试暂时无法对移动应用进行全面的覆盖，众包测试也存在着提取众包需求耗时耗力及众包工人缺乏测试经验等问题，两种测试手段的优势没有得到良好的结合。

本文设计并实现了安卓自动化测试驱动的众包需求生成系统。系统对目标应用的自动化测试数据进行处理，提取测试过程中触发的异常信息。基于测试操作序列和屏幕快照，提供复现步骤，引导众包工人在不同设备与环境下验证异常。使用反编译和静态分析技术，提取出源码中条件分支语句涉及的控件类型与自动化测试中未覆盖的窗口，引导众包工人探索新异常。系统从应用窗口层级模型、需求详情、机型设备、运行日志、引导信息等多维度展示了众包服务的概况。经本系统生成的众包需求可以发布到众包测试审核平台投入使用，组织和开展众包测试。系统根据众包工人历史信息，对其进行类 Top N 形式的众包需求推荐。此外，系统对众包测试的数据进行收集和汇总，实时反馈测试进展，使众包服务成为一个闭环。本系统基于前后端分离的架构进行实现，使用 Angular 作为前端框架，Spring Boot 为后端框架，两端数据格式统一，利用符合 RESTful API 规范的接口进行通信；结合 MongoDB 与阿里云 OSS 存储服务进行数据持久化处理；系统整体使用 GitLab 进行代码托管和版本控制，通过 Docker 容器技术进行打包，具有良好的可拓展性，降低系统部署与运维方面的开销。

系统上线后，本文选取 10 个安卓开源应用，在 15 台主流的手机设备上开展了对照实验。实验结果表明，众包工人参考本系统生成的众包需求进行测试，可比对照组多触发约 30% 的异常，整体覆盖率提高 15% 左右，系统提高了安卓移动应用众包测试的效率与质量。在此基础上，系统为生成众包需求至验收众测结果提供了一站式的服务，提高了众包任务发起者的工作效率。问卷调查的结果显示，80% 以上的众包任务发起者对本系统表示了肯定的态度。

关键词：安卓移动应用，众包测试，自动化测试，众包需求生成

南京大学研究生毕业论文英文摘要首页用纸

THESIS: The Design and Implementation of Crowdsourced Testing Requirement Generation System Driven by Android Automatic Testing

SPECIALIZATION: Software Engineering

POSTGRADUATE: Zhibin Wei

MENTOR: Professor Zhenyu Chen Research Assistant Chunrong Fang

Abstract

With the rapid development of the Mobile Internet, the quality of mobile applications has also received more and more attention. In order to solve the testing problems under the rapid iteration of mobile applications, automatic testing and crowdsourced testing have been widely used in the field of mobile application testing. However, automatic testing cannot provide comprehensive coverage of mobile applications. Crowdsourced testing also has the problems of costing tremendous effort to extract crowdsourced requirements and lacking professional crowdsourced workers. The advantages of automatic testing and crowdsourced testing are not well combined.

This thesis designs and implements a crowdsourced test requirement generation system driven by Android automatic testing. The system processes the automatic test data of the target application and extracts the exceptions triggered during the test. Based on test operation sequences and screen shots, workers are guided to find exceptions in different environments by reproduction steps. Using decompilation and static analysis techniques, the types of components involved in the conditional branch statements in the source code and the windows not covered in the automatic test are extracted to guide crowdsourced workers to explore new exceptions. The system shows the overview of crowdsourced services from multiple dimensions such as the application window hierarchy model, requirements details, models and equipment, operation logs, and guidance information.The crowdsourced requirements generated by this system can be posted to the crowdsourced inspection platform and put into use to organize and conduct crowdsourced testing.The system recommends crowdsourced requirements based on the historical information of crowdsourced workers in a Top N form. In addition, the system

collects and summarizes data for crowdsourced testing, and feedbacks test progress in real time, making the crowdsourced services a closed loop. This system is implemented using a separate client-server architecture. We utillize Angular as our front-end framework and Spring Boot is used as the back-end framework. The data format at both ends is unified, and the interface conforming to the RESTful API specification is used for communication. This system combines MongoDB and Alibaba Cloud OSS storage services to save data. In addition, the entire project uses GitLab for code hosting and version control, and is packaged through Docker container technology, which has good scalability and reduces the overhead of system deployment and maintenance.

After the system was launched, 10 Android open source applications were selected and controlled experiments were carried out on 15 popular mobile devices. The experimental results showed that by guiding the crowdsourced workers with our generated requirements, 30% more exceptions could be triggered and 15% more coverage rate was achieved than the control group. Therefore, the system improves the overall efficiency and quality of Android crowdsourced testing. In addition, the system provides a service for generating crowdsourced requirements and accepting crowdsourced test results, improving the working efficiency of crowdsourced sponsors. After the experiment, a survey showed that more than 80% of the crowdsourced sponsors expressed a positive attitude towards this system.

Keywords: Android mobile application, Crowdsourced testing, Automatic testing, Crowdsourced requirement generation

目录

表 目 录 ····· **x**

图 目 录 ····· **xii**

第一章 引言 ····· **1**

1.1 背景与研究意义 ····· 1

1.2 国内外研究现状 ····· 2

1.2.1 Android 自动化测试 ····· 2

1.2.2 Android 应用静态分析 ····· 3

1.2.3 众包测试 ····· 4

1.3 本文主要研究工作 ····· 5

1.4 本文的组织结构 ····· 7

第二章 相关技术综述 ····· **9**

2.1 前端相关技术 ····· 9

2.1.1 Angular ····· 9

2.1.2 ECharts ····· 9

2.1.3 Webpack ····· 10

2.2 后端相关技术 ····· 11

2.2.1 微服务 ····· 11

2.2.2 Docker ····· 11

2.2.3 Spring Boot ····· 12

2.2.4 MongoDB ····· 13

2.2.5 GitLab ····· 14

2.2.6 RESTful API ····· 14

2.3 推荐方式 ····· 16

2.3.1 协同过滤算法 ····· 16

2.3.2 冷启动 ····· 17

2.4 本章小结 ····· 17

第三章 系统需求分析与概要设计 ····· 19
3.1 系统需求分析 ····· 19
3.1.1 功能性需求 ····· 20
3.1.2 非功能性需求 ····· 22
3.2 用例描述 ····· 23
3.3 系统总体设计 ····· 27
3.3.1 “4+1”视图模型 ····· 29
3.4 核心模块设计 ····· 33
3.4.1 自动化测试数据处理模块设计 ····· 33
3.4.2 安卓静态分析模块设计 ····· 35
3.4.3 众包需求推荐模块设计 ····· 37
3.4.4 可视化模块设计 ····· 40
3.5 数据库设计 ····· 41
3.6 本章小结 ····· 45

第四章 系统详细设计与实现 ····· 47
4.1 自动化测试数据处理模块的详细设计与实现 ····· 47
4.1.1 提取自动化测试操作 ····· 49
4.1.2 构造窗口跳转模型 ····· 50
4.1.3 匹配屏幕截图 ····· 51
4.1.4 生成引导复现异常场景的众包需求 ····· 51
4.2 安卓静态分析模块的详细设计与实现 ····· 52
4.2.1 反编译 APK ····· 54
4.2.2 安卓源码扫描 ····· 56
4.3 众包需求推荐模块的详细设计与实现 ····· 57
4.3.1 冷启动 ····· 57
4.3.2 基于协同过滤的众包需求推荐 ····· 59
4.4 可视化模块的详细设计与实现 ····· 61
4.4.1 绘制窗口跳转模型 ····· 62
4.4.2 众包需求列表与详情 ····· 63
4.4.3 设备运行情况与测试信息 ····· 63

4.4.4 众测数据收集与汇总 ····· 64
4.5 系统案例与运行截图 ····· 65
4.6 本章小结 ····· 71

第五章 系统测试与实验分析 ····· **73**
5.1 测试环境 ····· 73
5.2 功能测试 ····· 73
5.3 性能测试 ····· 75
5.3.1 页面性能测试 ····· 75
5.3.2 接口性能测试 ····· 76
5.4 实验分析 ····· 78
5.4.1 实验目标 ····· 78
5.4.2 实验设置 ····· 78
5.4.3 实验结果 ····· 79
5.5 本章小结 ····· 81

第六章 总结与展望 ····· **83**
6.1 总结 ····· 83
6.2 展望 ····· 83

参考文献 ····· **85**

简历与科研成果 ····· **91**

致谢 ····· **93**

表 目 录

2.1 RESTful API 请求示例表 ····· 15

3.1 可视化模块功能性需求表 ····· 20
3.2 自动化测试数据处理模块功能性需求 ····· 21
3.3 静态分析模块功能性需求表 ····· 21
3.4 众包推荐模块功能性需求表 ····· 22
3.5 系统非功能性需求列表 ····· 22
3.6 系统用例表 ····· 23
3.7 创建任务用例表 ····· 23
3.8 查看任务用例表 ····· 24
3.9 查看众包概况用例表 ····· 24
3.10 查看需求详情 ····· 25
3.11 查看设备详情用例表 ····· 25
3.12 发布众包需求用例表 ····· 26
3.13 查看众测数据用例表 ····· 26
3.14 众包需求推荐用例表 ····· 27
3.15 测试事件属性表 ····· 34
3.16 Android 跳转控制语句示例表 ····· 36
3.17 Android 常见的控件类型 ····· 37
3.18 安卓移动应用常见异常类型及严重程度 ····· 38
3.19 用户信息表 ····· 43
3.20 任务信息表 ····· 43
3.21 应用信息表 ····· 44
3.22 众包需求表 ····· 44
3.23 用户选择表 ····· 45
3.24 设备信息表 ····· 45

5.1 测试环境配置表 ····· 73

5.2 自动化测试数据处理模块测试用例表 ····· 74
5.3 静态分析模块测试用例表 ····· 74
5.4 众包需求推荐模块测试用例表 ····· 75
5.5 可视化模块测试用例表 ····· 75
5.6 页面加载平均时间 ····· 76
5.7 实验设备表 ····· 78
5.8 实验结果统计表 ····· 79

图 目 录

2.1 Webpack 功能示意图 ····· 10
2.2 SpringBoot 关键架构图 ····· 12
2.3 基于 git 的项目开发流程示意图 ····· 14

3.1 系统用例图 ····· 19
3.2 系统整体架构设计图 ····· 28
3.3 逻辑视图 ····· 30
3.4 进程视图 ····· 31
3.5 物理视图 ····· 31
3.6 开发视图 ····· 32
3.7 自动化测试数据处理模块功能示意图 ····· 33
3.8 窗口跳转与邻接表示意图 ····· 35
3.9 安卓静态分析模块功能示意图 ····· 35
3.10 众包需求推荐模块功能示意图 ····· 37
3.11 可视化模块时序图 ····· 40
3.12 系统实体关系设计图 ····· 42

4.1 自动化测试数据模块流程图 ····· 47
4.2 自动化测试数据模块核心类图 ····· 48
4.3 提取自动化测试操作关键代码 ····· 49
4.4 构造窗口跳转模型关键代码 ····· 50
4.5 匹配屏幕截图关键代码 ····· 51
4.6 生成引导复现异常场景的众包需求关键代码 ····· 52
4.7 安卓静态分析模块流程图 ····· 53
4.8 安卓静态分析模块核心类图 ····· 54
4.9 Jadx 图形化界面工具使用效果图 ····· 55
4.10 反编译 APK 关键代码 ····· 55
4.11 源码扫描关键代码 ····· 56

4.12 冷启动实现关键代码 ····· 58
4.13 计算相似度关键代码 ····· 59
4.14 计算偏好程度预测值关键代码 ····· 60
4.15 可视化模块核心类图 ····· 61
4.16 绘制窗口跳转模型关键代码 ····· 62
4.17 众包需求列表与详情关键代码 ····· 63
4.18 设备运行情况与测试信息关键代码 ····· 64
4.19 众测数据收集与展示关键代码 ····· 65
4.20 提交任务界面截图 ····· 66
4.21 测试基本情况 ····· 66
4.22 应用窗口层级结构 ····· 67
4.23 众包需求列表 ····· 67
4.24 设备信息截图 ····· 68
4.25 设备详情部分截图 ····· 68
4.26 异常信息截图 ····· 69
4.27 引导信息部分截图 ····· 69
4.28 众测进展界面截图 ····· 70
4.29 众测反馈示例图 ····· 70

5.1 众包需求报告页面前端性能分析图 ····· 76
5.2 接口响应时间图 ····· 77
5.3 问卷调查关键评价统计图 ····· 80
5.4 各应用有效 Bug 数统计图 ····· 80
5.5 各应用代码覆盖率统计图 ····· 81

第一章　引言

1.1　背景与研究意义

随着移动互联网产业的繁荣，移动应用的数量与日俱增。据工信部统计，目前国内移动应用总数到达为 367 万，下载数量已超过千亿次 [1]，移动应用正以井喷的形式增长。其中，由 Google 公司开源的 Android 操作系统，以高于 80% 的市场份额[1]，占领着移动操作系统市场的主导地位，远超苹果的 iOS 和其它移动操作系统。基于 Android 系统的移动应用随着移动互联网逐渐渗透至社会服务的方方面面，改变了人们的生活方式。

与此同时，人们对于移动应用软件的质量以及用户使用体验也提出了更高的要求。与传统 Web 应用相比，Android 应用程序架构更加复杂。随着国内外 Android 移动设备厂商的井喷式增长以及移动应用版本的快速更新，传统的手工测试已经难堪重负。在移动应用更新迭代和质量保障的双重压力下，Android 自动化测试技术应运而出。测试人员首先根据 Android 移动应用的图形界面，设计相关用户交互事件；然后，编写测试脚本，借助自动化测试框架在真实的设备上执行测试用例，并记录过程中设备的运行情况与控制台堆栈信息；最后，通过对测试数据进行分析来提取移动应用的运行时缺陷。Baek 等人对上述工作进行了深入的研究，证明 Android 自动化测试是对传统测试手段的一种有效补充 [2]。不仅如此，自动化测试的脚本可以同时在多台设备上反复执行，极大地提高了对移动应用进行兼容性测试和性能测试的效率，降低了测试成本，也减少了来自于测试人员的执行错误 [3]。然而，自动化测试只能执行由测试人员提前编写好的测试脚本，具有一定的局限性。Chen 等人调研了业界主流的自动化测试技术，发现这些技术的覆盖率通常只能到达 50% 至 60% [4]。自动化测试暂时无法对应用中所有的窗口、控件以及对象状态进行覆盖，且对直观的界面样式错误也无法进行捕获。由此可见，自动化测试与真实用户相比仍存在着不足之处。

众包测试的提出，让移动应用快速获得大量真实用户的测试反馈，并以较低成本高效地完成测试任务的想法成为了现实 [5]。众包是一种分布式问题求解模式，它把过去只能由专人来完成的工作，发布到公开的第三方平台，其它群体在自愿的前提下通过平台领取并完成工作，以期达到类外包模式的效果 [6]。然而，开展众包测试需要专业测试人员根据自动化测试的结果，提取和设计众包需求，吸引众包工人领取并执行测试任务，完成测试报告。同时，众包工人应具

[1]www.idc.com/cn

有相应的应用操作和测试技能，并对待测应用有一定的了解。通常来说，不具备测试技能的普通用户才是众包工人的主要组成部分 [7]。因此，上述要求与快速召集大量众包工人并获取测试反馈等目标存在着一定程度的冲突。

上述情况表明，运用自动化测试和众包测试，可以在一定程度上对部分的的传统手工测试进行替代，降低 Android 移动应用的测试成本。自动化测试和众包测试并不是两种相对的测试手段，如何针对两者的优点对其进行有效地结合，是当前移动应用测试团队面临的挑战之一。在此背景下，本文设计并实现了安卓自动化测试驱动的众包需求生成系统。系统根据自动化测试结果提取需要众包工人进行验证的异常；利用程序静态分析技术，弥补自动化测试覆盖率的不足；通过文本和截图相结合的方式，引导众包工人开展测试。同时，系统为用户提供了生成众包需求至验收众测效果的一站式服务，使众包流程成为一个闭环，力求全面提升安卓移动应用众包测试的整体效率与质量。

1.2　国内外研究现状

1.2.1　Android 自动化测试

Android 移动应用是由 Java 语言进行编写，基于事件驱动的图形界面程序 [8]。因此，基于 Android 的自动化测试主要通过在设备上提供真实的交互指令，触发系统监听事件，达到模拟用户操作的效果。在此过程中，对应用程序的运行状态及堆栈信息进行记录，判断是否触发异常或错误 [9]。本文对主流的 Android 自动化测试技术进行了调研，其大体上可归纳为以下几类：

（1）随机事件驱动的自动化测试。通常来说，执行随机事件即在应用程序上进行无目的的操作，它对完全不具备测试经验的用户进行了模拟。这种测试方式尤其适用于软件开发和测试的初期阶段，它不需要提前对应用程序的图形界面与功能需求进行分析，也可以收集到可观的问题反馈，为测试人员节省了大量的时间 [10]。其中，Monkey [2]是 Android 官方提供的基于随机事件流的测试辅助工具，在业界有着丰富的优秀使用案例。测试人员只需对随机事件的类型与数量进行配置，通过 abd 连接真实设备后即可开展测试。然而，基于随机事件的 Monkey 工具，不支持为特定的移动应用添加测试逻辑，也不支持读取待测界面的信息来执行对应操作，仅能发掘一些浅显的程序错误，具有较大的局限性。

（2）基于测试脚本的自动化测试。在自动化测试优势的驱动下，国内外涌现了

[2]developer.android.com/tools/help/monkey.html

许多优秀的自动化测试框架，例如 Instrumentation[3]、Robotium[4]、UIAutomator[5]以及同时支持 Android 与 iOS 的跨平台测试框架 Appium[6]等等。这些框架都可以通过编程语言进行测试脚本的编写，定位移动应用中的控件，并针对目标控件触发框架提供的操作，例如点击、输入以及滑动等等。测试框架通过模拟用户在设备上进行操作，触发事件处理程序，同时记录执行过程中的日志与程序运行情况，以期达到替代传统的手工测试的效果。这种形式的自动化测试需要提供编写好测试脚本，然而越详尽的测试脚本所需的人工和时间开销也就越大 [11]。因此，目前基于脚本的自动化测试一般用于在不同的设备上进行兼容性测试、性能测试以及压力测试等需要重复执行的测试任务。

（3）基于遍历算法的自动化测试。为了探究更高效和智能的自动化测试方案，开发和测试人员借助自动化测试框架，针对 Android 的开发特性设计了通用的遍历算法，希望能对移动应用进行更全面的覆盖。其中，Amalf 等人针对 Android 移动应用的页面跳转与事件驱动机制，改进深度优先遍历算法，提出一种 Android 应用的智能遍历技术 [12]。在此背景下，国内也出现了许多知名的测试服务提供商：Testin 为减少人工编写测试脚本的时间，提供了智能自动化测试服务，关注产品逻辑业务流程问题，力求覆盖更多的场景 [7]。MoocTest 为了达到更全面和充分的测试效果，基于深度优先遍历算法，借助人工测试脚本与配置信息，对移动应用进行覆盖。在测试过程中记录移动应用的窗口变化状态、设备截图与堆栈信息，最终生成完整的测试报告 [8]。

目前，主流的自动化测试手段能够稳定高效的检查程序异常，并提供测试过程中的测试事件、运行日志以及屏幕截图等信息。因此，分析自动化测过程，提取相关事件序列，使用文本和截图相结合的形式，引导众包工人在不同环境不同设备下对异常进行复现是可行的。针对自动化测试过程中覆盖程度的不足等问题，可以通过程序静态分析技术进一步提取目标应用的众包需求，利用众包工人的群体智慧进行解决。

1.2.2　Android 应用静态分析

静态分析是指不执行相关的软件程序，仅从源码入手对该应用进行分析的手段。区别于传统的 Web 应用程序，Android 应用是事件驱动的 GUI 程序，开

[3]developer.android.com/reference/android/app/Instrumentation

[4]github.com/RobotiumTech/robotium

[5]developer.android.com/training/testing/ui-automator

[6]appium.io

[7]www.testin.cn

[8]service.mooctest.net

发者通过声明窗口、控件以及绑定事件监听处理函数来开发 Android 应用。当用户在设备上进行各种操作时，Android 系统会接收并生成对应的事件序列，广播至所有的移动应用。具体而言，位于栈顶的应用窗口将对这些事件进行拦截和处理，每个事件携带着坐标、事件类型以及涉及的窗口控件等参数。其次，控件会对事件进行捕获，检查对应的控件监听事件，如果存在相应的事件处理函数，则将进入到事件处理环节；否则，事件将一直冒泡至上级控件 [13]。此外，在事件处理函数中可以指定各种控制逻辑，如窗口跳转、发送请求以及更新数据等。Android 移动应用的事件驱动机制使得开发人员可以高效的编写相关代码，完成各种功能需求。因此，从静态分析的角度而言，无需在设备上运行对应的移动应用，只要理解代码中的窗口和控件声明、绑定事件类型、事件处理函数以及控制逻辑，即可分析出目标移动应用的逻辑结构。

Lint[9]是通用的 Android 静态代码检查工具，由 Google 官方提供开源并进行维护。它通过代码扫描来发现开发过程中潜在的问题，提醒开发团队及时修正，提高代码质量。除此之外，开发团队可以提供自定义的配置文件来对开发风格进行规范。Yang 等人提出一种用于 Android 静态分析的框架 GATOR，通过构建窗口跳转图，描述 Android 窗口界面的控制变化信息 [14]。其中，图的节点代表 Android 应用程序的一个界面，边记录着界面之间的跳转信息。Arzt 等人定义了 Android 应用程序中的关键字段、对象以及逻辑，通过应用的 Context 信息，分析 Android 生命周期的以及控件监听事件的 Hook 函数，构造了程序的事件调用模型 [15]。Wang 等人对目前的静态分析技术进行了调研，由于脱离真实环境以及缺乏完善的代码处理技术，静态分析工具在稳定性、易用性和兼容性方面的依然存在着不足之处 [16]。

上述研究表明，通过捕获代码中的窗口跳转控制信息，分析应用整体层级结构，这种方案是可行的。针对自动化测试中未覆盖的应用窗口和没有达到相应状态的控件，提取出相应的众包需求，可有效的提高测试的整体覆盖率。

1.2.3　众包测试

在软件工程领域，Mao 等人对给出了众包软件工程的定义 [17]：“众包软件工程是指由大量潜在的、未定义的在线工人以公开召集的形式, 承担外部软件工程任务的行为”。在此基础上，Chen 等人派生出众包软件测试的定义 [18]：众包软件测试即支持软件测试的所有众包活动，如方法、技术工具和平台，都属于众包软件测试领域。在众包软件测试活动中, 主要参与者包括众包任务请求者、众包工人和众包平台。通常来说，众包测试的工作流程包括准备任务、执行测试以

[9]developer.android.com

及整合报告三个阶段 [19]，其步骤可归纳为：

（1）众包任务请求者设计和发布众包任务；

（2）众包工人领取并解答测试任务，提交测试报告；

（3）众包任务请求者接收和整合测试报告。

在此背景下，国内外涌现了一批优秀的众包测试平台，如 Google 的 Crowdsource [10]，腾讯 QQ 众测 [11]以及阿里云 MQC [12]等，它们为实现上述流程提供了技术支持。众包请求发起者根据应用的需求对众包任务进行设计，并发布到众包测试平台；用户在众包平台上领取自己感兴趣的众包任务，完成后以测试报告的形式对其进行提交。

作为一种有效的新兴测试模式，众包测试已经在多个领域得到了广泛应用，如软件质量测试、兼容性测试、性能测试和 GUI 测试。传统的软件质量测试主要通过人工进行开展，成本较高，因此众包测试最先被应用于该测试领域 [20]。通过召集大量众包工人来开展兼容性测试，可以获取到更多关于该移动应用在不同机型不同网络环境下的真实数据。相比传统的购买设备和人工操作的方式，众包兼容性测试显著降低测试成本 [21]。众包性能测试可以收集各类用户在不同执行环节的情况，有效的了解待测应用的性能表现，如 Chrome 中通过内置埋点来实现性能测试的框架 Chrome telemetry [13]。此外，通过网络将 GUI 测试任务分发放给大量的众包工人，可以获得较多的真实使用反馈 [22]。

然而，大多数众包工人缺乏专业的测试技能，这种情况弱化了众包测试快速召集大量众包工人的优势，使得众包测试产生较多的不准确结果 [23]。因此，如何设计一种良好的众包需求提取和引导机制，从而提高众包需求生成效率和众包测试整体质量，是本文工作的目标和重点。

1.3　本文主要研究工作

上述研究表明，当前的自动化测试技术已经能够替代一部分传统手工测试，稳定高效地开展测试工作。然而自动化测试仍然存在一些技术挑战，例如无法满足某些对象状态、无法达到某种判断条件导致分支没有执行、无法完全覆盖所有 Activity、无法发现 UI 界面样式的错误等问题，这些技术挑战导致了自动化测试覆盖率没有办法到达 100%，也没有办法完全替代传统的手工测试。然而，

[10]www.crowdsource.com

[11]task.qq.com

[12]www.aliyun.com/product/mqc

[13]www.chromium.org/developers/telemetry

这些挑战对于众包工人而言并不困难，因此可以通过将自动化测试与众包测试进行结合来共同优化测试流程。为了充分利用自动化测试与众包测试的特点，使两者相互促进，提高测试的整体效率与质量，本文设计并实现了安卓自动化测试驱动的众包需求生成系统。系统分析自动化测试的数据，在不执行代码的情况下，对应用源码进行静态分析，最终生成有助于引导众包工人在不同环境下复现异常、探索未覆盖窗口以及相关类型控件的众包需求。系统为用户提供了可视化界面，对众包概况、需求详情、设备详情等信息进行了展示。经本系统生成的众包需求，可以发布至众包测试审核平台，组织和开展众包测试。系统根据用户的历史信息，对众包需求进行类 Top N 形式的推荐。此外，系统对众包测试的数据进行收集和汇总，借助可视化界面实时反馈测试进展，使众包协作的流程成为一个闭环。本文的主要工作如下：

（1）处理自动化测试数据：利用 M 平台的自动化测试框架对应用程序进行初步覆盖，发现移动应用的运行时缺陷。接着，提取自动化测试操作、屏幕截图以及触发异常的场景信息。系统把具体的测试操作封装成独立的对象实例，提供通俗易懂的语义化描述，并匹配对应的屏幕截图。通过发生窗口跳转的事件序列，构造窗口跳转邻接表，计算从入口到达各个窗口的最短路径，进而获取从应用入口到达异常场景的所需的复现步骤。最后，基于上述信息并进行后续处理，生成引导众包工人复现异常场景的众包需求。

（2）静态分析安卓应用源码：通过 Jdax 反编译工具获取移动应用对应的源码，并使用正则表达式对源码中的窗口跳转信息以及条件分支语句进行捕获。分析应用整体的窗口跳转情况以及条件分支语句内包含的相关组件类型，引导众包工人对自动化测试未覆盖的窗口和涉及条件判断的控件进行深入探索。

（3）众包需求的冷启动与推荐：系统根据需求类型，窗口深度以及严重程度等信息来确定众包需求的困难程度与优先级。对于具有相关测试知识的用户，系统将按需求难度进行降序排序；其他用户则将优先展示简单的众包需求。当众包工人存在历史记录时，系统通过计算需求的相似度，对用户的偏好程度进行预测，最后将根据预测值降序排列众包需求，通过分页 Top N 的形式进行推荐。

（4）实现承载上述功能的可运行系统：用户只需上传待测应用的 APK 文件，即可生成最终的众包需求。系统实现了可视化模块，从应用窗口跳转模型、需求详情、机型设备、运行日志、引导信息等多维度展示了众包服务的概况。在此基础上，用户可将众包需求通过本系统发布至众包测试审核平台投入使用，进行后续的众包测试组织和开展工作。此外，系统对众包测试的最终数据进行了收集和汇总，为用户提供了生成众包需求到验收众包结果的一站式服务。

（5）系统测试与实验分析：系统部署后进行了相关的功能测试与性能测试，

检验了系统功能的可用性以及系统整体在高并发场景下的表现情况。除此之外，使用 15 台主流的安卓设备以及 10 款开源安卓移动应用对系统进行了对照实验，验证了安卓自动化测试驱动的众包需求生成系统的有效性。

1.4　本文的组织结构

本文总共分为六个章节，各章节的主要内容如下：

第一章，引言。本章介绍了系统的提出背景与研究意义，对 Android 自动化测试技术、静态分析技术以及众包测试技术的国内外研究现状进行了阐述。在此基础上，引出了本文的目标与主要工作，并对本文的组织结构进行简述。

第二章，相关技术综述。本章介绍了搭建系统所涉及的核心技术，包括前端框架 Angular、图表工具 ECharts、打包工具 Webpack、后端框架 Spring Boot、容器引擎 Docker、数据库 MongoDB、版本管理工具 GitLab 以及接口规范 RESTful API 等。最后，对相关的冷启动问题和协同过滤算法进行了简要的介绍。

第三章，系统的需求分析与概要设计。本章首先关注系统的功能性需求与非功能需求，借助用例图与用例表等手段对其进行了描述。其次，参照“4+1”视图模型，从代码组织至物理部署等多方面阐述了系统的整体架构，进而划分出系统的各个功能模块，并就其实现原理及合理性进行了分析。最后，通过实体关系模型展示了数据持久化的设计，并对数据表中的各个字段进行了解释。

第四章，系统的详细设计与实现。本章通过流程图、类图与关键部分代码对系统主要功能模块的具体实现进行了详细描述，包括自动化测试数据处理模块、安卓静态分析模块、众包推荐模块以及可视化模块。最后，通过真实的使用场景和界面截图对本系统的操作流程以及运行状态进行展示和说明。

第五章，系统测试与实验分析。本章对系统的功能测试和性能测试进行了设计，展示相应的测试结果，并通过开展对照实验及数据分析证明了安卓自动化测试驱动的众包需求生成系统的有效性。

第六章，总结与展望。本章总结了论文完成期间的主要工作，对系统与技术的未来方向作了进一步展望。

第二章　相关技术综述

2.1　前端相关技术

2.1.1　Angular

Angular 是一款由 Goolge 开源，用于构建用户界面的轻量级渐进式框架。它因高性能、组件化以及易上手等特点，受到广泛关注，是当前业界最为流行的前端框架之一。Angular 在框架层面为前端开发提供了性能优化。当页面数据更新时，它不会对整个页面进行重排和重绘，而是动态分析出页面需要变化的部分，在保证其它元素不受影响的前提下，对页面进行局部更新 [24]。这一技术的提出，解决了传统前端开发中频繁的数据更新而带来的页面性能问题，用户在进行页面的操作时，卡顿的感觉将明显降低，极大地改善了用户体验。对于开发者而言，Angular 易于上手，其原因在于它只关注视图层，并对应用结构进行了良好的划分和组织，减少了代码的编写与繁琐的配置。此外，Angular 拥有良好的开发生态，有助于快速构建前端项目，丰富页面功能。

Angular 采用 MVC 的架构模式，将视图层，后端数据以及前端业务逻辑进行了解耦。视图层只需关注 HTML 和 CSS 的开发，业务逻辑层只需对访问后端获取的数据进行处理，两层之间不直接进行通信。开发者将数据与前端元素进行绑定后，无需进行繁琐的 DOM 元素操作，只需实现对应的数据更新逻辑。当数据发生变更时，框架将追踪数据的变化与检测绑定的目标，数据的变化会映射至前端页面，使页面同时进行更新。这种双向绑定的设计，使得项目的状态管理变得非常简单直接。不仅如此，Angular 提出了组件化的开发哲学，它把页面视为成多个独立的功能模块的组合，开发者只需实现各个模块的功能与逻辑，并在页面中将其引入即可。组件复用极大地提高了页面开发的效率，同时也为前端单元测试提供了有利的条件。

综上，本文选择 Angular 作为前端开发框架，通过组件化以及双向绑定的开发模式，对项目进行拆分和解耦，避免了直接操作视图层而带来的性能问题以及大量编写耦合代码时出现的错误，极大地提升了开发效率，也提高了代码的可读性和可维护性。

2.1.2　ECharts

ECharts 是一个使用 JavaScript 实现的开源可视化库，它为前端开发者提供了数据可视化的定制功能，使用它可以制作美观且交互丰富的图表，并可以在

各种 PC 和移动设备上良好运行，兼容当前绝大部分浏览器，包括主流的火狐 Firefox、谷歌 Chrome、苹果 Safari、微软 IE 等 [25]。ECharts 功能丰富，覆盖程度广，不仅提供饼图、线图、散点图以及柱状图等常见的图表，还可以通过自定义的方式将数据映射到各种图形以及多图形之间的组合。ECharts 在地图、勘测、数据等方面都有着丰富的使用案例，它以免费、美观、全面以及易上手等特点受到了业界的青睐 [26]。因此，本文选择 ECharts 作为图表库来辅助实现可视化模块，用于在不同的浏览器设备上对窗口跳转模型、众包需求列表以及其它数据图表等进行绘制和展示。

2.1.3　Webpack

Webpack 是一个前端项目构建工具，用于对开发中繁杂的业务代码和资源进行处理，包括转译代码、压缩体积、按需引入等功能，它最终会把整个项目打包成一份精简的静态资源文件。Webpack 会从项目的入口以递归的形式遍历所有被引用的模块，然后将应用程序所依赖的全部资源进行编译和融合等处理。此外，借助 Webpack 能把前端项目中的各种资源，如 JS，样式表，第三方工具库等都作为模块来引入和加载，并通过语法转换生成 HTML 可以直接引入的 bundle.js 文件。Webpack 的主要功能如图 2.1所示。

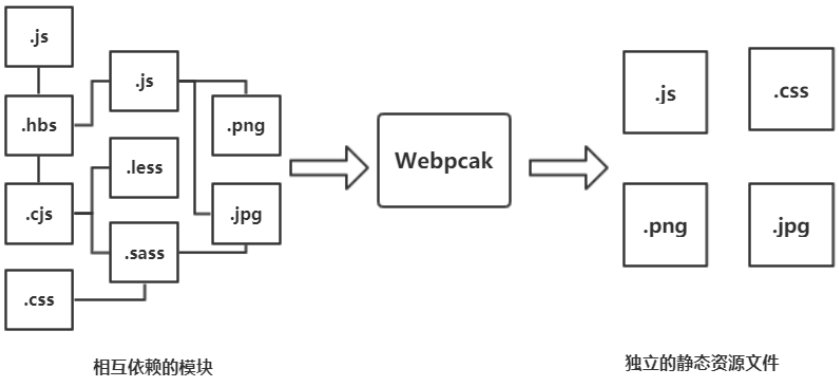


图 2.1: Webpack 功能示意图

作为一个构建工具，Webpack 不仅可以对 JS、CSS 和图片等资源文件进行混淆和压缩，降低部署难度，还可以通过其预留的接口可以针对项目构建流程中的各个生命周期进行相关任务的定制。在此基础上，Webpack 可以将代码打包进不同的模块之中，从而实现按需引用，这种机制可以保证页面只加载需要的 JS 代码文件，减少首次请求的时间。借助 Webpack 还可以搭建本地服务器，在热更新的条件下对项目进行预览。此外，Webpack 还支持 AMD、CommonJS、

ES6 模块方案，可以高效地对第三方库进行集成，无需考虑兼容性问题。因此，本文选择 Webpack 作为前端项目构建工具，对各种业务代码进行打包，生成的静态资源文件托管在 Node.js 服务器上，通过符合 RESTful API 规范的接口与后端进行通信，达到前后端分离和代码解耦的效果。

2.2　后端相关技术

2.2.1　微服务

在早期的软件开发中，为了快速完成业务需求，开发人员往往把项目中所有的代码堆砌在一起，这种形式的代码组织方案被称为单体架构。随着时间的推移，基于单体架构的系统，其维护成本会越来越高 [27]。垂直架构和分层架构是解决这一困境的有效手段。垂直架构通过负载均衡改善前后端通信瓶颈问题，提高了项目的的可扩展性和可靠性；分层架构，尤其是面向服务的架构解决了复杂应用系统之间的集成和通信问题 [28]。在此基础上，微服务的思想被提出，用于对系统进行更细粒度的划分，降低系统整体的复杂程度以及后期的维护成本，提高开发效率。微服务非常适合多功能大型系统的开发与集成，它将系统整体分割为一个个小的功能子组件，每个微服务组件负责实现独立的业务和功能，由不同的开发团队进行开发、部署和维护，使系统具有良好的水平拓展性 [29]。同时，单个组件的故障不会影响系统整体的运行，开发人员可以快速地对故障组件进行重启或替换，使得系统状态变得更加可控。不仅如此，微服务组件之间只需满足统一的接口规范即可相互集成，其具体实现并没有强制要求，开发团队可以结合业务特点自由地选择合适的技术栈。基于上述优点，本系统将以微服务的形式部署系统并提供服务。

2.2.2　Docker

Docker 是一项开源的容器引擎技术，它使用 Go 语言进行开发。Docker 可以让开发者把应用程序和程序所需的运行环境打包成一个标准镜像，并且以容器的方式运行 [30]。软件开发中，环境配置是最为冗余和耗时的工作，只有软件应用需要的各种依赖以及环境变量都设置正确时，应用才能正常运行。如果软件应用的环境需要频繁变更或者升级，将给运维带来极大的压力。在此背景下，Docker 被提出用于解决应用开发的环境与部署问题，开发者可以在 Docker 容器中嵌入所有的运行和开发环境。它提供了一种 Linux 容器打包机制，可以将应用程序与程序所依赖的运行环境打包在同一个文件内，从而实现与系统底层的隔离，这些策略保证了 Docker 容器内应用程序运行环境的稳定性 [31]。用

户只需在目标环境上安装 Docker，并运行对应的镜像文件，Docker 即会为该文件创建一个虚拟容器，保障其正常运行。利用 Docker 容器技术，不管目标环境是 Linux、Mac 还是 Window，开发者都可以高效地部署应用程序。Docker 解决了传统软件开发中的环境兼容难题，为系统的部署和运行提供了一种新的方式，全面提高了系统的可移植性，使得开发团队可以专注于业务功能的研发，无需为繁琐和重复的环境配置问题耗费时间和精力。Docker 因开发简单、部署高效以及体积精简等特点受到了广大开发团队的喜爱，故本系统也采用 Docker 容器技术来对系统和系统运行环境进行一站式打包。

2.2.3　Spring Boot

Spring Boot 是在 2013 年推出用于简化 Web 项目开发、配置、测试和部署的开源框架，它是目前业界最流行的后端开发框架 [32]。Spring Boot 伴随着 Spring4.0 诞生，Boot 意为引导，因此 Spring Boot 致力于帮助开发者快速搭建 Spring 框架以及启动 Web 容器。Spring Boot 的工具库中包含了大量 Web 开发中实用的基础框架，如内嵌容器 Tomcat、JMS 框架、日志框架 Log4j、测试框架 JUnit、NoSQL 数据库 MongoDB 等。Spring Boot 的本质依旧是基于 Web 开发流程而对 Java 组件的生命周期进行了管理，它主要包含 Core、Bean、JDBC、MVC、AOP、Test 等基本模块，Spring Boot 的关键架构如图 2.2所示。

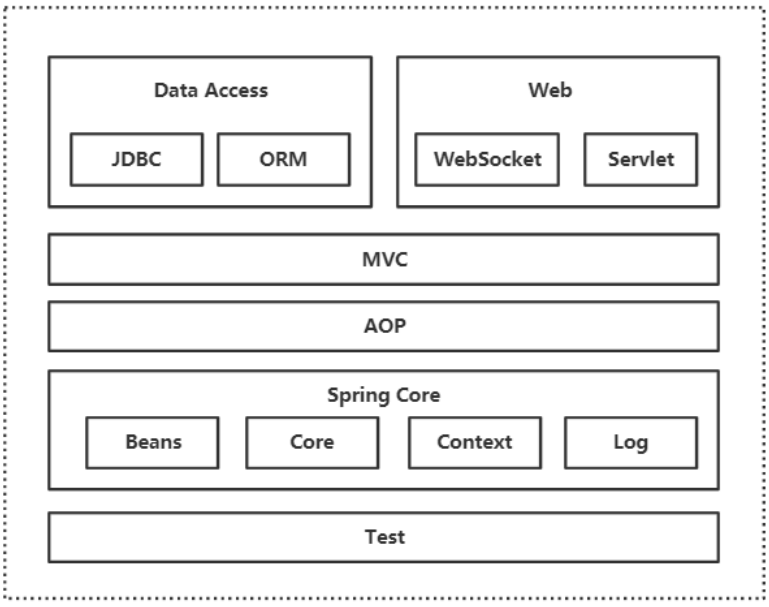


图 2.2: SpringBoot 关键架构图

其中，Core 模块主要包括 Spring 的核心内容，所有模块都将对 Core 进行加载；Bean 模块则负责类的创建和管理以及进行依赖注入和版本控制等功能；

JDBC 模块对底层数据库进行了封装，方便开发者进行连接和调用；MVC 模块分离了控制器、模型对象、过滤器以及处理程序对象等角色，使程序代码更加容易编写和维护；AOP 模块提供了面向切面编程的实现，支持用户自定义拦截器和切点，将切入的代码和其它代码解耦，降低代码的冗余性。

相较于传统 Web 开发工具而言，Spring Boot 极大的简化了 Spring 的复杂的样本形式配置 [33]，它无需部署 WAR 文件，内嵌 Tocamt，可以直接打包成可运行的 jar 文件部署到云服务器上。同时，通过 Maven 进行依赖包的统一管理，用户不需要在各种第三方网站下载和引入依赖包，当需要同时引入多个依赖包时，Maven 会自行选择可配合运行的依赖包，减少第三方库之间的版本冲突。因此，本文使用 Spring Boot 作为系统后端的搭建框架，极大的简化了开发环境和依赖的配置，可以将更多的精力放在业务功能的实现上。

2.2.4 MongoDB

MongoDB 是使用 C++ 进行编写的非关系型数据库，它具有高性能和语法自由的特点 [34]。MongoDB 具有面向代码的特点，它可以方便的对 JSON 的文档集合进行管理，是业界最流行的开源非关系型数据库。它不仅对传统关系型数据库中的关系表结构进行兼容，还支持基于 NoSQL 的文档模型，可以高效的对数据对象进行持久化操作。该模型的提出用于改善传统数据库表模型中耦合度过高的问题，提供了更友好的数据建模方式，可以直接快速地构造复杂的数据类型，而无需担心数据表的复杂度问题。

MongoDB 并不需要预先设置存入到数据库中的数据表的任何结构信息，这使得更新数据库表字段变得更加便捷。传统的关系型数据库进行数据存储时，如果需要给程序中某个实体增加、删除或修改属性，则需要同时对数据库表、数据实体和业务逻辑一起进行更改。尽管可以利用自定义的自动化工具将该过程封装成可复用的任务，但这对于代码整体来说显得非常冗余。特别的，如果对线上的项目进行更新时，将面临更多的风险。使用基于 NoSQL 的 MongoDB 进行数据存储和持久化，只用在业务逻辑层做必要的改动即可完成更新，无需对数据库表进行变更。当业务功能需求与数据实体需要频繁变更时，MongoDB 可以极大地降低开发成本和维护成本，使系统功能和版本得以快速迭代。

不仅如此，MongoDB 还为开发者提供了良好的 API，支持 JSON 查询、范围查询以及正则表达式查询，还可以通过面向对象的方式构造实例进行增删查改，避免了 SQL 语句的设计与维护。MongoDB 的性能表现良好，内存占用少，安装和配置简单高效，适用于各种大型和中型项目开发。因此，本文选择 MongoDB 作为系统数据库，基于 NoSQL 语言对实现数据库业务逻辑的相关代码进行编写。

2.2.5　GitLab

GitLab 是一个遵循 MIT 开源协议的项目生命周期管理平台，它使用 Git 对代码仓库进行管理，还提供了基于 Web 界面的 wiki 编写以及跟踪 issue 等功能 [35]。图 2.3展示了基于 Git 的项目开发流程。其中，clone 命令负责将远程仓库的项目拷贝至本地；开发团队可以根据不同的功能建立不同的代码分支，checkout 命令为此提供了创建和切换分支等功能；通过执行 pull 命令，开发者可将远程分支中的代码拉取至本地的工作区并进行合并，然后在此区域对项目进行开发或修改；当开发任务结束后，可执行 add 命令将代码的改动提交到本地暂存区进行缓存；接着，执行 commit 命令，将暂存区中所有的更改作用于本地仓库，并生成对应的 git 版本号；最后，通过执行 push 命令将本地仓库的代码同步到远程仓库，完成系统的功能更新与版本升级。

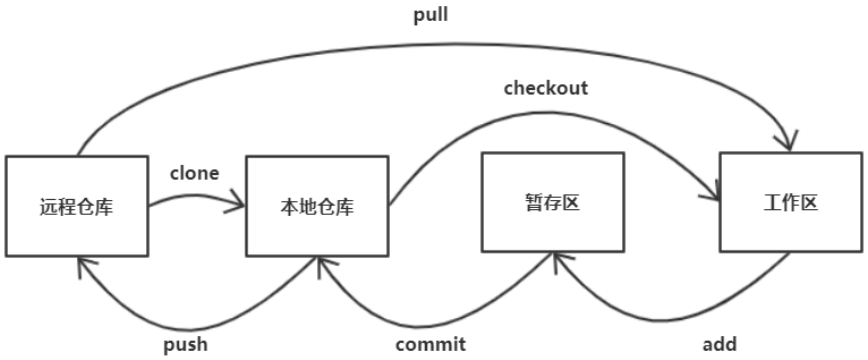


图 2.3: 基于 git 的项目开发流程示意图

GitLab 具有灵活、轻便和高效等特点，它目前已经成为业界主流的企业级代码托管平台，使用者包括阿里巴巴集团，IBM 和 SpaceX 等国际知名大公司。借助 GitLab 以及 Git，开发团队可以方便高效的对项目进行管理，减少因手动删改文件而导致的版本不一致等问题。同时，GitLab 还提供了版本之间的比对和分支开发等功能，不同的开发者之间可以高效地进行协同开发，提升工作效率。基于上述优点，本文选择 GitLab 作为项目的代码托管和版本控制工具，并在此基础上分别搭建 Web 前后端服务，避免了繁琐的文件保存和删除工作，提高了项目开发和运维的整体效率。

2.2.6　RESTful API

客户端设备层出不穷，因此必须设计一套 API 来规范不同设备与服务端之间的通信。在此背景下，RESTful API 被提出，因其直观、简单、结构清晰、易

于拓展和自定义等特性，受到了业界的广泛认可 [36]。其中，HTTP 协议就是 RESTful 风格的一个典型应用。REST 的四大原则如下：

（1）客户端-服务端架构。客户端和服务端彼此间不相互依赖，可以进行独立的替换和开发。客户端记录资源 URI，通过对用户界面与数据存储进行分割，提高了跨平台视图层界面的兼容性。服务端则通过简化服务器的业务层级提升了服务器的拓展性。此外，Web 服务的搭建框架 Sping Boot 对 RESTful API 提供了良好的支持 [37]。

（2）无状态。客户端发送的请求应包含服务端所需的所有内容，服务端内部不会对客户端状态进行存储，每次请求都视为一次独立的事件。如果需要进行状态管理，可以通过 Cookie 等辅助技术实现。

（3）可缓存。响应请求可以直接或间接定义是否可缓存。良好的缓存策略可有效降低客户端和服务端之间的请求数量，减少客户端加载时间，释放服务端处理压力，从而进一步提高系统的性能。此外，设置缓存过期时间可避免客户端利用过期的响应数据进行其它操作，对防止 CSRF 攻击起到了一定的作用。

（4）统一资源接口。统一接口包含了一组受限的预定义的操作。只要访问接口相同，对资源的操作就是相同的，它的实现需要满足四个条件，分别为固定类型操作、资源标识、可自描述的信息和超媒体 [38]。服务端需要预先定义不同的操作类型，通过数据的元操作对资源进行增删改查。使用统一资源定位符标识资源，作为客户端的访问路径。当请求指定了某个资源时，服务根据请求携带的资源标识进行查找并返回。传递的数据应该是可以自我描述的，每个请求响应包含足够多的信息来描述如何处理该数据。客户端通过请求的 body 内容传递参数和数据，通过请求头和请求状态码来提供历史状态，该项技术也称为超媒体。

表 2.1: RESTful API 请求示例表

地址	请求资源	参数	说明
https://service.test.net	/detail	/:userId	通过 GET 方法，获取用户详细信息，参数为用户 ID

表 2.1展示了 RESTful API 接口部分的示例，该请求描述了服务端的地址，需要进行的操作以及需要传递的参数类型。服务端进行处理后，以 JSON 为数据载体对请求进行响应。基于 RESTful API，可以将系统的前后端进行解耦，前端作为客户端向用户提供服务，后端负责复杂功能的实现和资源存取，两端只需关注自身的业务逻辑，当功能发生关联时通过接口进行通信。不仅如此，基于 RESTful API 的服务端接口还具有易于开展测试等特点 [39]。因此，系统将基于该规范对系统前后端之间的通信接口进行设计。

2.3　推荐方式

2.3.1　协同过滤算法

Adoma 等人对主流的推荐算法进行了调研，将它们归类为基于协同过滤的推荐算法、基于内容的推荐算法和混合推荐算法 [40]。这其中以基于协同过滤的推荐算法最为大众常见和熟知，它因简单、高效、易实施等特点得到了业界的广泛使用。协同过滤是利用群体的行为来挖掘事物之间的相关性，通过这种相关性对目标进行推荐。通常来说，协同过滤涉及用户和物品两个群体，一方面可以从物品的相似性入手，向用户推荐与他喜欢的物品的同一类或相似的物品；另一方面，也可以从用户的角度出发，找到与他相似的用户，推荐另一名用户所爱好的物品。可以看出，协同过滤的依据在于用户的行为以及物品之间的相关性 [41]，用户的历史行为越多，物品的相关性计算越准确，则协同过滤的效果就越明显。用户的历史行为包括显式和隐式两种。通过直接打分的形式对物品进行评价，这样的方式为显式记录。然而，例如转发、评论、收藏、点赞、购买等行为则属于隐式记录，它更贴近真实的用户场景。协同过滤可以延伸出基于用户的协同过滤以及基于物品的协同过滤两种算法。根据上面的思想，我们可以向用户推荐与他喜好的物品具有相似性的物品，这就是基于物品的协同过滤。因此，需要先对物品的相似度进行计算。如果存在大量的用户同时对物品 i 以及物品 j 进行过选择，则这两种物品在某种程度上可视为相似。具体来说，计算物品 i 和物品 j 之间的相似度公式如下 [42]：

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

其中，$N(i)$ 和 $N(j)$ 分别代表喜欢物品 i 和 j 的用户集合，当同时喜欢两者的用户越多时，它们的相似度计算值也就更高。接着，根据用户的历史行为，预测用户对物品的喜好程度，具体的计算公式如下 [43]：

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

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。$S(j,k)$ 表示与物品 j 属于同一类的物品集合，其个数为 K。按照公式的定义，r_{ui} 表示用户 u 对物品 i 的直接评分，但在实际的应用中通常根据用户是否存在隐式记录来进行取 1 或 0 值 [44]。最后，对物品喜好程度进行降序排序，取其中的 Top N 个物品 [45] 推荐给用户即可完成基于物品的协同过滤推荐。

类似的，寻找与用户最相似的用户群体，将他们喜好的物品推荐给目标用户，这就是基于用户的协同过滤算法的核心思路。具体的实施流程与上述步骤相近，故不在此进行展开论述。

2.3.2　冷启动

协同过滤算法基于用户的历史数据进行对其他物品的推荐。然而，新用户不存在历史行为，新物品也不会和任何用户进行关联。如何对新用户进行个性化推荐以及如何将一个新物品推荐给合适用户，一直是协同过滤算法中必须解决的问题，这类问题被称为冷启动。冷启动问题常见的解决方案如下：

（1）收集用户信息。根据系统分类算法所需的参数，有偏向的引导用户提供相关信息，将用户划分为不同的群体。例如在注册阶段，设置必填或选填的信息项，收集来自用户的背景信息；也可以在初次进入系统的时候，让用户主动选择自己感兴趣的内容。最后，参照对应类别中的老用户群体的偏好情况，对新用户进行冷启动 [46]。

（2）以非个性化的方式进行推荐。其中最常见的就是直接向新用户推荐热门产品，例如今日头条和京东等知名网站的首页推送。根据 Netflix 等人的调查，大部分用户对当前的系统中热门物品都具较高的期待 [47]，可以通过这种方式满足用户的好奇心，完成冷启动。

（3）主动启动用户兴趣。该模式以部分物品作为测试，记录新用户对这些物品的反馈，对用户历史行为数据进行记录，使用户向老用户进行过渡，完成后续的物品推荐。这种方式要求测试物品的类别广泛且具有明确的特点，以期在有限的用户操作中挖掘出其感兴趣的内容。

2.4　本章小结

本章主要对搭建系统所用到的技术以及支撑相关功能的算法理论进行了阐述。首先介绍了前端开发中的 Angular 框架，绘制图表的流行工具 ECharts 以及对前端项目进行打包的工具 Webpack。然后介绍了后端的微服务思想、容器 Docker、开发使用的主流框架 Spring Boot，进行数据持久化存储的 MongoDB，前后端通信的接口规范 RESTful API 以及进行版本控制与源码托管的 GitLab。最后，针对推荐方式中的协同过滤算法以及冷启动问题进行了介绍。

第三章　　系统需求分析与概要设计

3.1　系统需求分析

安卓自动化测试驱动的众包需求生成系统的用例图如 3.1所示。本系统的服务目标主要为众包任务发起者，其次为领取众包需求的用户，即众包工人。对于众包任务发起者而言，他们希望能够高效方便地提炼出众包测试需求，在短时间内完成各种测试目标，收集大量的真实使用反馈；对于众包工人而言，他们则希望找到自己感兴趣的，具有详细信息的众包需求。

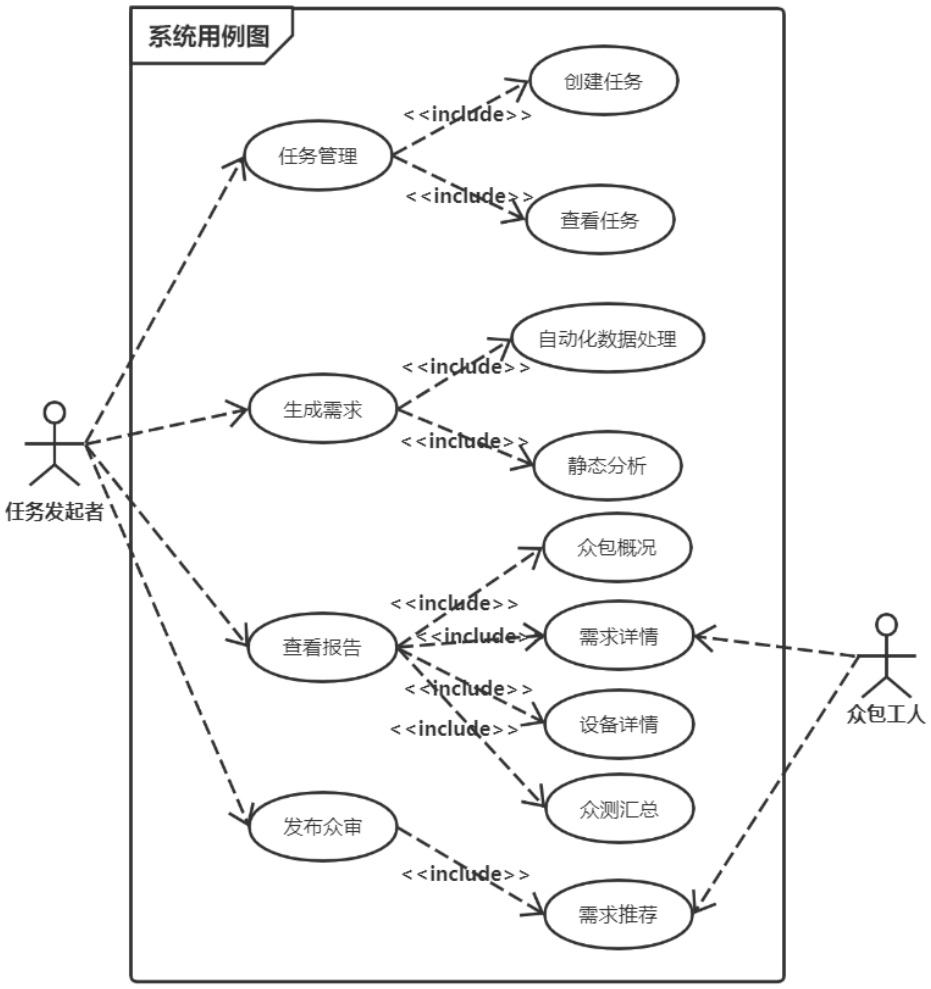


图 3.1: 系统用例图

因此，本系统的业务目标非常明确：在自动化测试基础上，对移动应用进行

进一步分析，提炼出合理的众包需求；同时，众包需求应具有引导信息，降低众包工人的专业门槛，进一步扩大受众范围。系统应从应用窗口层级模型、需求详情、机型设备、运行日志、引导信息等多维度展示众包服务的完成情况。经本系统生成的众包需求可以发布至众包测试审核平台投入使用，组织和开展众包测试。为了提高测试效率，系统根据众包工人历史信息，对其进行类 Top N 形式的众包需求推荐。最后，系统应对众包测试的数据进行收集和汇总，实时反馈测试进展，使众包服务流程成为一个闭环。

3.1.1　功能性需求

功能性需求是设计和实现系统的基础，本章基于上文阐述的业务目标，对系统的功能性需求进行设计，并划分出相应的功能模块。本系统作为 M 平台生态中的一环，用户系统将由平台统一管理，本系统仅沿用其提供的登录态信息，不搭建独立的用户系统，减少如用户注册、登录等重复工作。在此基础上，功能性需求主要由可视化模块、自动化测试数据处理模块、安卓静态分析模块以及众包需求推荐模块这几部分内容组成。可视化模块是系统与用户进行交互的部分，涵盖了系统的主要使用流程，表 3.1展示了可视化模块的功能性需求。

表 3.1: 可视化模块功能性需求表

需求 ID	需求名称	需求描述	优先级
R1	创建任务	用户在系统 Web 端界面创建测试任务，并提供可选的众包协作服务。	高
R2	查看任务	展示用户的任务状态、历史任务和完成任务后的报告	高
R3	查看众包概况	用户查看众包服务完成后的概况，包括页面层级结构和众包需求列表。	高
R4	查看需求详情	系统 Web 端展示需求的详细信息，包括设备信息、异常信息和引导信息。	高
R5	查看设备详情	展示设备的详细信息，包括品牌型号等基本信息、自动化测试中的执行信息、运行截图和日志等	中
R6	发布众包需求	用户将生成的众包需求发布至众包测试审核平台，进行后续操作和开展众包测试。	高
R7	展示众测数据	对众包测试的信息进行汇总，展示具体的评审情况。	高

该模块是本系统直接与用户进行交互的部分，用户的所有操作都在此进行，如最基础的创建任务和查看任务，用户在系统上按照指引提交任务，勾选“众包协作服务”选项，即开启了系统的流程；用户可以浏览任务的状态，待任务完成后可以查看相应的任务报告。任务报告涵盖了应用整体概况、设备列表、Bug 列表和众包概况，众包概况中包含应用的页面层级结构以及众包需求列表。用户点击某个具体的需求，将展示触发异常的设备信息、需求详情和引导信息，用

户可以在不同设备不同环境下对其开展测试。系统提供了设备详细信息的展示，如品牌、型号、分辨率，以及自动化测试过程中的屏幕截图和运行日志。在查看完众包概况后，用户可选择将众包需求发布到众包测试审核平台开展众包测试。此外，系统会收集众包测试中用户的真实反馈信息，对其进行汇总和展示。

表 3.2: 自动化测试数据处理模块功能性需求

需求 ID	需求名称	需求描述	优先级
R8	提取测试操作	从自动化测试日志中提取出不能分解的独立的测试操作，如点击和滑动等，补全所有相关信息并封装成类。	高
R9	提取异常场景	从自动化测试日志中提取出发生异常的场景，补全相关异常信息。	高
R10	提取已覆盖窗口跳转信息	从自动化测试日志与测试操作中提取出所有已覆盖的窗口，构造能描述图信息的邻接表。	高
R11	匹配屏幕快照	根据时间戳信息，将自动化测试中的截图和测试操作以及异常场景进行匹配。	高
R12	搜索最短路径	在已覆盖窗口中，根据单源最短路径算法，基于已有的测试操作，计算出从入口窗口到达各个窗口的最短路径。	高
R13	生成引导复现异常场景的众包需求	根据异常场景的所在窗口，通过最短路径补全复现步骤，引导用户到达异常场景，同时对异常的出错信息进行描述，用户可以自行提供相关的配置文件。	高

表 3.3: 静态分析模块功能性需求表

需求 ID	需求名称	需求描述	优先级
R14	反编译 APK	如果用户上传了应用的 APK 文件，则需要对其进行反编译，获取移动应用源码，为后续扫描源码提供基础；如果用户上传了应用源码则直接进入后续步骤。	高
R15	获取整体窗口跳转信息	在应用源码中对窗口跳转信息进行扫描，获取整体的窗口跳转信息，构造能描述图信息的邻接表。	高
R16	获取待测组件类型	在应用源码中对条件分支语句进行扫描，获取相关的控件类型信息。	高
R17	生成探索未覆盖窗口的众包需求	通过整体的窗口信息与已覆盖窗口进行比对，获取未覆盖窗口信息，并通过最短路径引导众包工人到达前置窗口进行探索。	高
R18	生成测试指定类型组件的众包需求	在具体的窗口下，提供待测的组件类型，如按钮和输入框等，要求众包工人运用等价类等方式进行测试。	高

任务创建完成后，系统首先将调用 M 平台的自动化测试框架，对应用进行自动化测试。待测试结束，系统的功能将转由自动化测试数据数据处理模块负责，该模块是系统的核心模块，承担了许多复杂的任务，主要包括提取测试操作

和异常场景、添加描述性信息、匹配对应的屏幕快照和生成引导复现异常场景的众包需求，自动化测试数据处理模块的功能性需求如表 3.2所示。

自动化测试需要非常精确的定位才可以编写脚本，并对应用控件进行相关的操作。通用的遍历算法所通常无法满足上述要求，故存在覆盖率不高等问题。然而，通过对测试任务进行拆分，并提供相关的指引，即使信息相对模糊，众包工人也能通过经验和思考将上下文逻辑补全，良好地完成任务。因此，系统将通过对移动应用的源码进行静态分析，扫描应用中的窗口跳转控制语句和条件分支语句，比对出自动化测试没有覆盖的窗口，生成探索未覆盖窗口以及测试相关类型控件的众包需求。静态分析模块具体功能性需求如表 3.3所示。

表 3.4: 众包推荐模块功能性需求表

需求 ID	需求名称	需求描述	优先级
R19	众包需求推荐	根据用户的历史习惯和个人信息，采用基于物品的协同过滤方案；对第一次进入系统的用户，采用冷启动模式。	中

此外，系统还需要为众审平台提供需求推荐的接口。根据用户的历史情况预测用户对众包需求的偏好，并对预测值进行降序排序，通过分页和每页固定个数的形式实现类 Top-N 形式推荐，众包推荐模块的功能性需求如表 3.4所示。

3.1.2　非功能性需求

为了保障系统的正常运行、使用与维护，对安卓自动化测试驱动的众包需求生成系统需要满足的非功能性需求进行了设计，如表 3.5所示。

表 3.5: 系统非功能性需求列表

需求 ID	需求名称	需求描述	优先级
R20	可靠性	系统可以在较长时间能维持稳定运行的状态，能够处理高并发下的请求并进行响应，用户不会感觉到明显的卡顿。	高
R21	有效性	系统提供的安卓自动化测试驱动的众包需求生成技术，应能满足众包任务发起者的需求，有效提高众包测试的效率和质量。	高
R22	可扩展性	系统需要设计规范的通信接口，产出的数据应以通用的格式存储，避免与运行环境产生强耦合，无需大的改动即可接入新的功能模块。	中
R23	易用性	系统应功能明确，操作简单，降低用户学习和理解系统使用的成本	高

本系统要求界面简洁，用户能够良好地进行各项操作。考虑到系统可能与

其他模块集成，需使系统能够保持良好的扩展性。系统需支持高并发下运行的场景。同时，系统提供的技术应能弥补自动化测试覆盖率的不足，有效提高众包测试的效率和质量。

3.2　用例描述

本节将从用户角度出发，基于上文的系统用例图 3.1，通过用例表对其进行用例描述。如表 3.6所示，用户对系统进行操作时共涉及 8 个主要功能用例，分别为创建任务、查看任务、查看众包概况、查看需求详情、查看设备详情、查看众测数据汇总、发布众包需求以及众包需求推荐。下文将通过用例表对给出的系统用例进行详细描述。

表 3.6: 系统用例表

用例编号	用例名称	对应功能需求 ID
UC1	创建任务	R1
UC2	查看任务	R2
UC3	查看众包概况	R3
UC4	查看需求详情	R4
UC5	查看设备详情	R5
UC6	发布众包需求	R6
UC7	查看众测数据	R7
UC8	众包需求推荐	R19

表 3.7: 创建任务用例表

用例编号	UC1
用例名称	创建任务
参与者	众包请求发起者
用例描述	用户向系统提交应用安装文件，创建众包协同测试任务
前置条件	用户在 M 平台完成登录
优先级	高
正常流程	1. 用户进入 M 平台； 2. 用户点击导航栏的“发布任务”选项； 3. 选择“App 自动化任务”，并上传应用安装包； 4. 勾选“转入众包协同测试”选项； 5. 填写任务名称和任务描述等信息； 6. 成功创建任务，等待任务执行。
扩展流程	任务提交成功后，可以在“我的任务”中查看任务执行状态

表 3.7是创建任务的用例描述表。创建任务是众包请求发起者进入系统流程的入口，用户可以通过系统 Web 端界面手动上传应用安装包，勾选“转入众包

协同测试”选项，填写相关任务信息，最后创建测试任务。测试任务创建后将由 M 平台进行审核，通过后系统将执行该次任务。首先，调用 M 平台提供的自动化测试框架进行对目标应用进行初步覆盖，获取其运行过程中的异常和缺陷；然后，系统进入自动化测试数据处理以及静态分析等流程；最后，基于上述步骤生成目标应用的众包需求并绘制相应的测试报告。

表 3.8: 查看任务用例表

用例编号	UC2
用例名称	查看任务
参与者	众包请求发起者
用例描述	查看历史任务与正在执行的任务状态
前置条件	用户在 M 平台完成登录
优先级	高
正常流程	1. 用户进入 M 平台； 2. 用户点击导航栏的“我的任务”选项； 3. 选择“App 自动化任务”； 4. 展示历史任务以及正在执行中的任务状态。
扩展流程	如果任务已完成，用户可以执行“查看报告”或“发布众审”等操作。
异常流程	如果用户没有已经提交的任务，则页面会显示相应的提示信息。

表 3.9: 查看众包概况用例表

用例编号	UC3
用例名称	查看众包概况
参与者	众包请求发起者
用例描述	查看众包服务的基本情况
前置条件	任务执行完毕
优先级	高
正常流程	1. 用户进入 M 平台； 2. 用户点击导航栏的“我的任务”选项； 3. 选择“App 自动化任务”； 4. 展示历史任务以及正在执行中的任务状态； 5. 用户点击一个具体任务的“查看报告”选项； 6. 用户点击报告中的“众包概况”标签页； 7. 跳转至众包概况页面，展示应用层级结构图和众包需求列表。
异常流程	如果没有勾选“转入众包协同测试”选项，则报告中不会对众包概况进行展示

表 3.8是查看任务的用例描述表。用户在使用系统的过程中，常常需要查看任务列表，查找已完成的历史任务和确认已提交任务的状态，这是任务管理的基本功能之一。查看任务功能负责展示用户的任务列表，用户可以对正在执行的任务以及历史任务进行查看。如果任务已经完成，则向用户提供对应的“查看报告”以及“发布众审”等功能。

表 3.9是查看众包概况的用例描述表。众包概况页面展示了系统进行众包服务后的基本信息。测试任务完成后，用户可以从查看任务界面进入测试报告，查看众包概况，页面包括应用层级结构模型和众包需求列表，众包需求列表简述了需求的基本信息，如需求类型、异常信息、触发原因、所在窗口等。

表 3.10: 查看需求详情

用例编号	UC4
用例名称	查看需求详情
参与者	众包请求发起者
用例描述	查看众包需求的详细信息
前置条件	任务执行完毕
优先级	高
正常流程	1. 用户进入 M 平台； 2. 点击导航栏的“我的任务”选项； 3. 选择“App 自动化任务”； 4. 展示历史任务以及正在执行中的任务状态； 5. 用户点击一个具体任务的“查看报告”选项； 6. 用户点击报告中的“众包概况”标签页； 7. 点击一个具体众包需求表格中的“查看详情”标签； 8. 跳转至需求详情页面，展示需求详细信息。
扩展流程	需求详情中的设备编号为超链接，点击将进入“设备详情”页面。

表 3.10是查看需求详情的用例描述表。需求详情页面展示了众包需求的详细信息，包括设备简要信息、需求内容和引导步骤。用户在众包概况的需求列表中，点击某一个具体需求的“需求详情”标签，即可打开对应的需求详情页面。

表 3.11: 查看设备详情用例表

用例编号	UC5
用例名称	查看设备详情
参与者	众包请求发起者
用例描述	用户设备的详细信息和测试运行信息
前置条件	用户查看测试报告
优先级	高
正常流程	1. 用户点击报告中的“众包概况”标签页； 2. 展示应用层级结构图和众包需求列表； 3. 点击众包需求表格中的“查看详情”标签； 4. 点击设备信息中的设备编号； 5. 跳转至设备详情页面，展示设备详细信息、测试运行日志和屏幕截图。

表 3.11是查看设备详情的用例描述表。设备详情页面展示了设备的基础信息。用户如果想查看设备详细信息或自动化测试过程中设备的屏幕截图和运行

日志，可以点击需求详情页面中的设备编号，页面将跳转至设备详情界面，该界面展示了设备运行情况、性能结果、测试截图和日志等信息。

用户生成众包需求的最终目标是投入使用，并利用其组织和开展相关的众包测试，为此本系统与 M 公司的众包测试审核平台进行了业务上的集成。通过发布众包需求功能，众包请求发起者可以将此次任务生成的众包需求将推送到 M 公司的众包测试审核平台，借助该平台进行后续操作，对众包测试的内容和形式进行定制。发布众包需求的用例描述如表 3.12 所示。

表 3.12: 发布众包需求用例表

用例编号	UC6
用例名称	发布众包需求
参与者	众包请求发起者
用例描述	将生成的众包需求发布至众审平台
前置条件	任务执行完毕
优先级	高
正常流程	1. 用户点击导航栏的“我的任务”选项； 2. 选择“App 自动化任务”； 3. 展示历史任务以及正在执行中的任务状态； 4. 在一个已完成的任务处点击“发布众审”选项。
异常流程	如果没有勾选“转入众包协同测试”选项，则不提供发布众审功能

表 3.13: 查看众测数据用例表

用例编号	UC7
用例名称	查看众测数据
参与者	众包请求发起者
用例描述	查看使用众包需求开展众包测试后对应的真是数据
前置条件	用户将需求发布到众审平台
优先级	高
正常流程	1. 用户点击导航栏的“我的任务”选项； 2. 选择“App 自动化任务”； 3. 用户点击一个具体任务的“查看报告”选项； 4. 用户点击“众测数据”标签页； 5. 跳转至众测数据页面。
扩展流程	用户点击某场众包测试的“详细信息”标签，可以查看该场测试的各项评审情况。

表 3.13展示了查看众测数据的用例描述。对用户而言，他们希望了解使用这些众包需求进行的众包测试的效果。为此，系统提供了查看众测数据的功能。将众包需求发布至众审平台并开展众包测试后，系统会收集用户对这些需求的真实反馈，更新报告，在测试报告中进行展示，使众包协作成为一个闭环。

众包测试审核平台只负责对众包测试进行组织和开展，为了提高测试的执行效率，系统为该平台提供了众包需求推荐的功能。系统实现了返回需求列表的接口，该接口接收众包测试涉及的应用、场次、用户等信息作为参数，针对具体的用户历史行为数据，进行众包需求的过滤和推荐，最终以 JSON 数组为载体返回按照预测值降序排序的需求列表。众包需求推荐的用例描述如表 3.14 所示。

表 3.14: 众包需求推荐用例表

用例编号	UC8
用例名称	众包需求推荐
参与者	众包测试审核平台
用例描述	众包测试审核平台请求根据众包工人历史记录进行排序的需求列表
前置条件	用户将需求发布到众包测试审核平台
优先级	中
正常流程	1. 众包测试审核平台组织并发起了一次众包测试； 2. 众包测试审核平台将用户信息和此次众测的基本信息发送至本系统； 3. 系统进行冷启动或基于众包需求的协同过滤； 4. 返回降序排序后的众包需求列表。

上文从用户角度出发，通过用例图和用例表，阐述了系统需要为用户提供的操作，包括创建任务、查看任务、查看众包概况、查看需求详情、查看设备详情、发布众审、查看众测数据以及众包需求推荐。为了实现一个稳定可靠的系统，并提供上述功能，下文将对系统整体和各个模块进行分析与设计。

3.3　系统总体设计

本文设计的安卓自动化驱动的众包需求生成系统，以生成众包需求为核心，对众包发起者提供了创建任务、查看任务、查看众包概况、查看需求详情、查看设备详情、发布众包需求和众包需求推荐等功能。为了降低系统之间的耦合度，本文对系统层次进行了划分，本系统自顶向下可以分为视图层、业务层和数据层。项目整体使用 Docker 容器引擎进行打包和部署，使用 GitLab 进行代码托管和研发协作。系统整体架构设计如图 3.2所示。

视图层：视图层负责用户和系统之间的交互，是系统功能的外在体现，由前端技术栈负责实现，包括前端框架 Angular、模板引擎 Pug、图表库 ECharts 以及打包工具 Webpack。Angular 是整个前端项目的骨架，它负责组织前端的界面结构、业务逻辑和具体样式。界面结构对应了系统的每一个功能页面，使用 HTML 语言进行编写，每个元素在页面中遵循“盒子模型”，以矩形的形式进行呈现；业务逻辑对页面结构中的元素进行事件绑定和监听，当用户在元素上触发事件

时，执行相应的逻辑，该部分代码由 JS 语言进行编写；样式代码则负责对页面和元素的外观进行绘制和渲染。除此之外，Angular 还提供了页面路由和双向绑定功能。基于上述技术实现的前端项目，可以向用户提供任务管理、查看众包概况、查看需求详情、查看设备详情、发布众包需求以及查看众测数据等功能。系统采用前后端分离的架构，两端之间采用符合 RESTful API 规范的接口进行通信，以 JSON 为载体统一数据格式。前后端通信的流程如下：首先，用户在系统 Web 端进行操作，触发相应的监听事件；接着，前端向后端发起对应的请求，后端进行事件的处理；最后，后端服务处理成功，向前端返回处理的结果与数据。

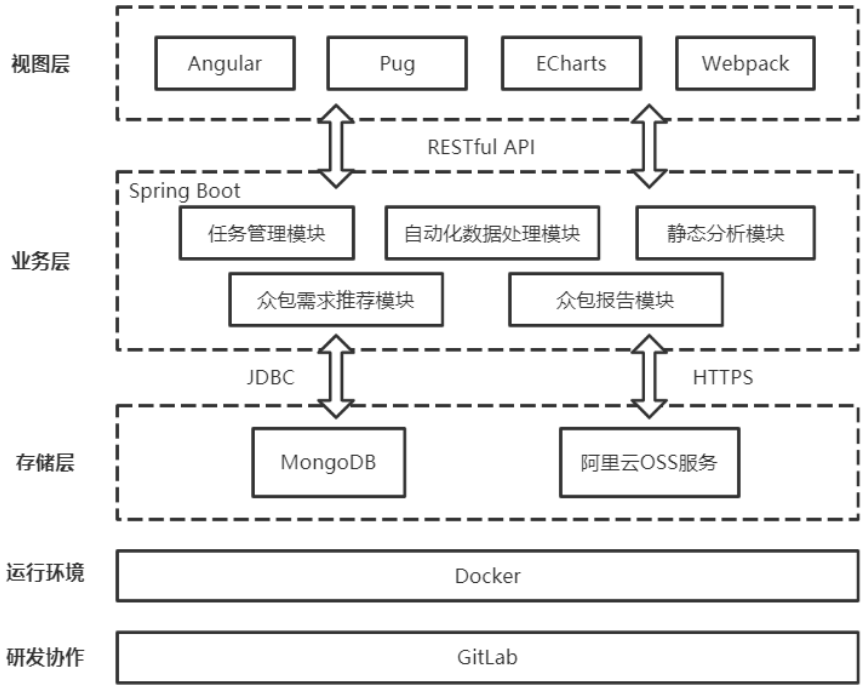


图 3.2: 系统整体架构设计图

业务层：业务层负责提供系统声明的各项功能，由后端技术栈负责实现，其技术栈包括 Web 框架 Spring Boot、数据库 MongoDB 和容器引擎 Docker 等。Spring Boot 负责整个后端项目的搭建，是联结后端各个功能的基础。本系统将后端代码划分为 Controller、Service、Impl 和 Dao 四层。其中，Controller 层负责接收和响应前端发送过来的 RESTful API 请求，参数和响应通过 JSON 实现，针对不同的请求，需要实现不同的 Controller 方法。在响应完前端请求后，需要处理具体的事件，执行系统的业务逻辑。为了降低代码的耦合度，Controller 只负责请求的处理，具体的功能接口封装在 Service 层实现。根据不同的请求，Controller 只需要调用不同的 Service，待事件处理完毕后，以 JSON 为载体，将数据和运

行状态返回给前端。Impl 负责对 Service 提供的功能进行具体的实现，该层封装了系统的主要业务逻辑代码。如果事件处理过程中涉及到数据的存取，则需要把 JDBC 代码封装至 Dao 层。因此，Dao 层中封装着连接和访问数据层的业务代码，包括 MongoDB 的增删改查以及阿里云 OSS 服务的访问。业务层涵盖任务管理模块、自动化测试数据处理模块、静态分析模块、众包需求推荐模块及可视化模块，各个模块的具体设计将在下文进行阐述。

数据层：数据层负责对系统涉及的数据进行持久化处理。常规的数据将直接存储至 MongoDB 中；如果是 zip 包、屏幕截图等大型二进制文件，则借助阿里云 OSS 服务进行存储，将其对应的 url 地址存储至 MongoDB 中。

运行环境：为了实现微服务的效果，使得部署在统一服务器的后端项目不互相影响，系统将利用 Docker 容器引擎进行打包，使其成为一个可移植的 Docker 镜像，然后在目标环境中部署运行。每个 Docker 容器都类似一个真实的服务器，满足项目运行的各种环境。同时，使用 Docker 技术还可以使系统达到平台无关的效果，使其可在 Linux、Mac 和 Window 等平台下良好运行。

研发协作：系统使用 GitLab 进行代码托管，不同的研发人员可以同时进行开发而互不干扰。同时，可以对系统进行方便高效的版本管理，包括克隆至本地仓库、更新远程仓库和版本回退和前进等，有效的避免了直接对文件进行操作。

3.3.1　“4+1”视图模型

“4+1”视图模型是软件工程中最流行的，分析系统架构的一种方法，它从五个不同的角度对系统整体进行描述，分别为场景视图、逻辑视图、开发视图、进程视图和物理视图。场景视图关注系统的最终用户需求，为系统整体提供上下文环境，使其余四个视图可以进行有机关联。因此，上文所提及的系统用例图 3.1可视为本系统的场景视图。

图 3.3是本系统的逻辑视图，该视图从用户角度描述了系统提供的功能和服务，是整个系统的抽象表述。Web 包是经过打包后的前端代码，用于提供平台与用户交互的图形界面，前后端之间通过符合 RESTful API 规范的接口进行通信。为此，系统后端提供了两个 Controller 来处理前端请求，分别为 TaskController 和 ReportController。其中，TaskController 负责处理任务相关的请求，包括创建任务、查看任务和发布众包需求等；ReportController 负责处理报告类请求，包括查看众包概况、需求详情、设备详情和众测数据等。在系统中，Service 负责处理具体的业务逻辑；TaskService 负责创建和查看任务，涉及的数据对象为 Task。DataHandleService 负责对自动化数据进行处理，包括提取测试操作、匹配屏幕截图、构造窗口邻接表、匹配屏幕截图等，涉及的数据对象为 TestAction 和 Exception

等；StaticAnalysisService 负责对源码进行静态分析，包括反编译 APK、源码扫描等，涉及的数据对象为 AppActivity 和 View 等；通过对上述数据进行处理和整合，系统将生成不同类型的众包需求，即为 Requirement。RecommendService 负责对众包需求 Requirement 进行推荐，包括冷启动和协同过滤等步骤；ReportService 负责根据不同的报告类型，从数据库获取和组织数据，对报告进行渲染，包括众包概况、需求详情、设备详情和众测数据等。

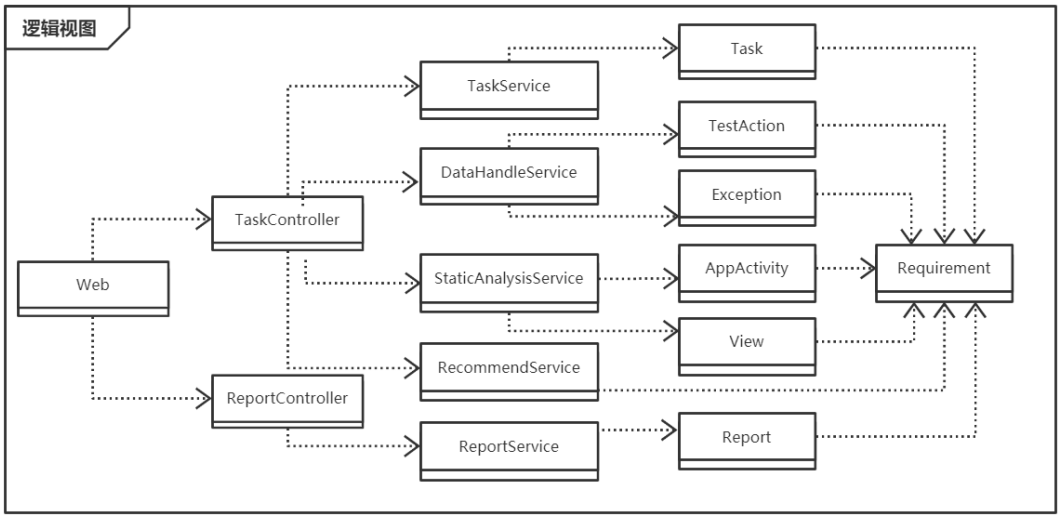


图 3.3: 逻辑视图

图 3.4是系统的进程视图，该视图面向系统集成者视角，展示了系统各个模块间通信的过程。用户提交任务时，系统通过 TaskService 创建对应的测试任务；当任务完成审核后，系统调用 M 平台的自动化测试框架对应用进行自动化测试，并产出测试数据。DataHandleService 负责对测试数据进行处理，包括提取测试操作、匹配屏幕截图、构造窗口邻接表、匹配屏幕截图等，最后生成引导复现异常的众包需求，并存储到数据库中。StaticAnalysisService 负责对应用源码进行静态分析，包括反编译 APK、源码扫描等，最后生成探索未覆盖窗口和相关类型控件的众包需求，并存储到数据库中。当用户查看任务时，系统会从数据库获取用户对应的任务信息，并对信息进行处理，最后返回任务列表。系统提供了查看众包概况、需求详情、设备详情和众测数据的页面，当用户打开某个页面时，将通过 ReportServie 进行处理，返回页面所需的 JSON，然后由前端进行渲染。为了将众包需求投入使用，用户需要将其发布到众包测试审核平台，发布成功后系统会记录此次 Task 对应的 paperId，作为获取众测数据的依据。此外，系统还为平台提供了众包需求推荐服务，RecommendService 接收请求携带的用户信息，查询用户历史记录，返回按照一定规则进行排序的众包需求列表。

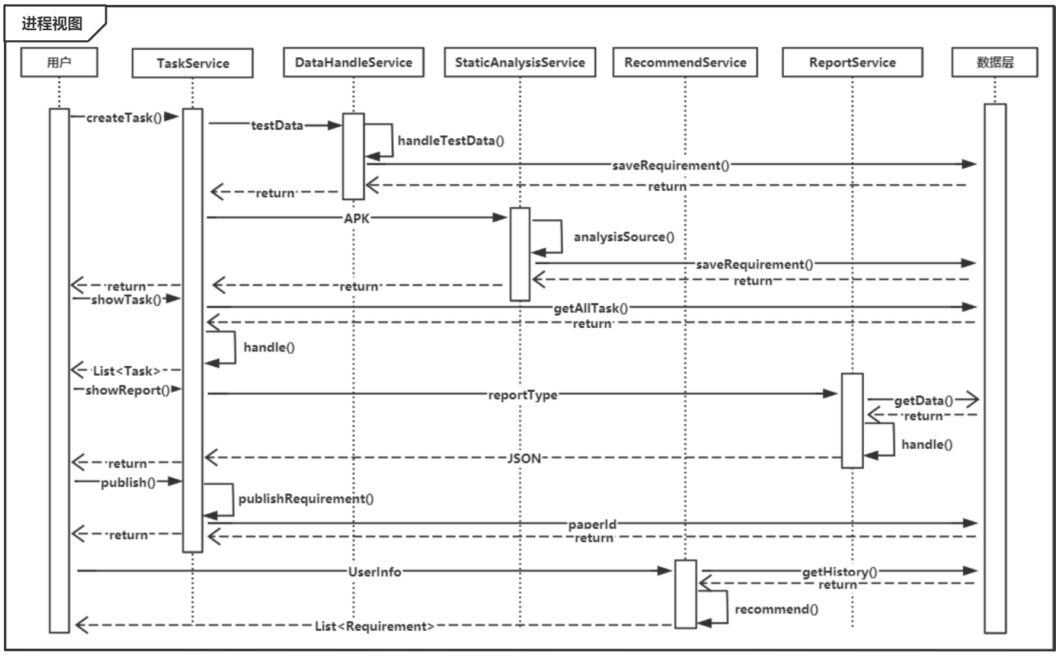


图 3.4: 进程视图

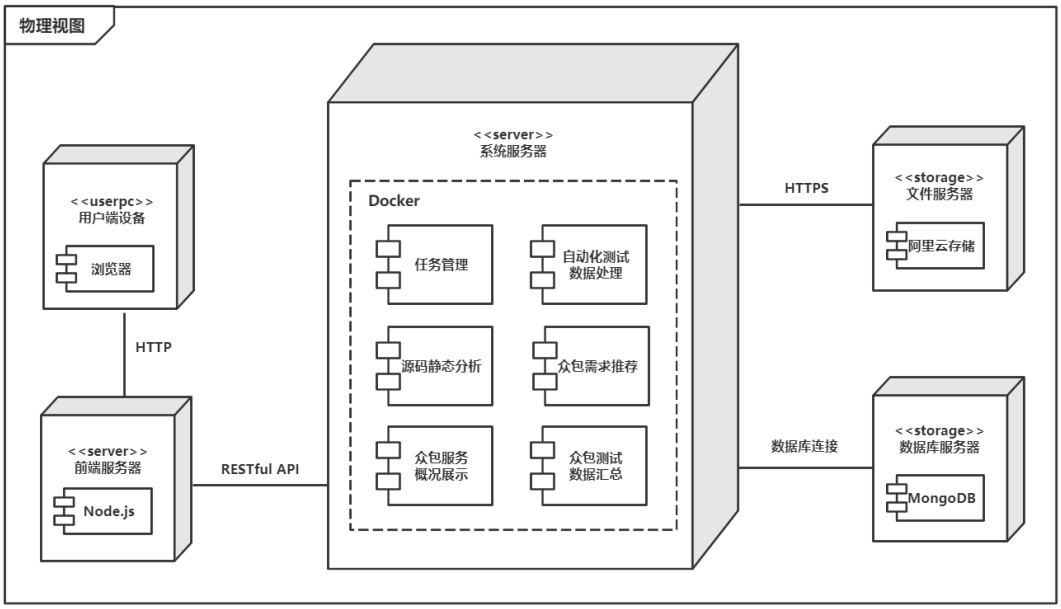


图 3.5: 物理视图

物理视图也称部署视图，它把系统软件映射到物理设备上，用于解决系统拓扑结构、安装和部署等问题，图 3.5是本系统的物理视图。任务发起者通过浏览器访问系统的 Web 界面，系统前端代码作为静态资源由 Node.js 服务器进行托管；系统后端则承担了主要的业务功能，如任务管理、自动化测试数据处理、源

码静态分析、众包需求推荐、众包服务概况展示以及众包测试数据汇总等。在数据处理的过程中，为了降低数据库的压力和优化存储逻辑，程序安装包和屏幕截图等大型文件将上传至阿里云 OSS 服务进行存储，数据库只存储对应的 URL 地址，其它数据则使用 MongoDB 进行持久化处理。

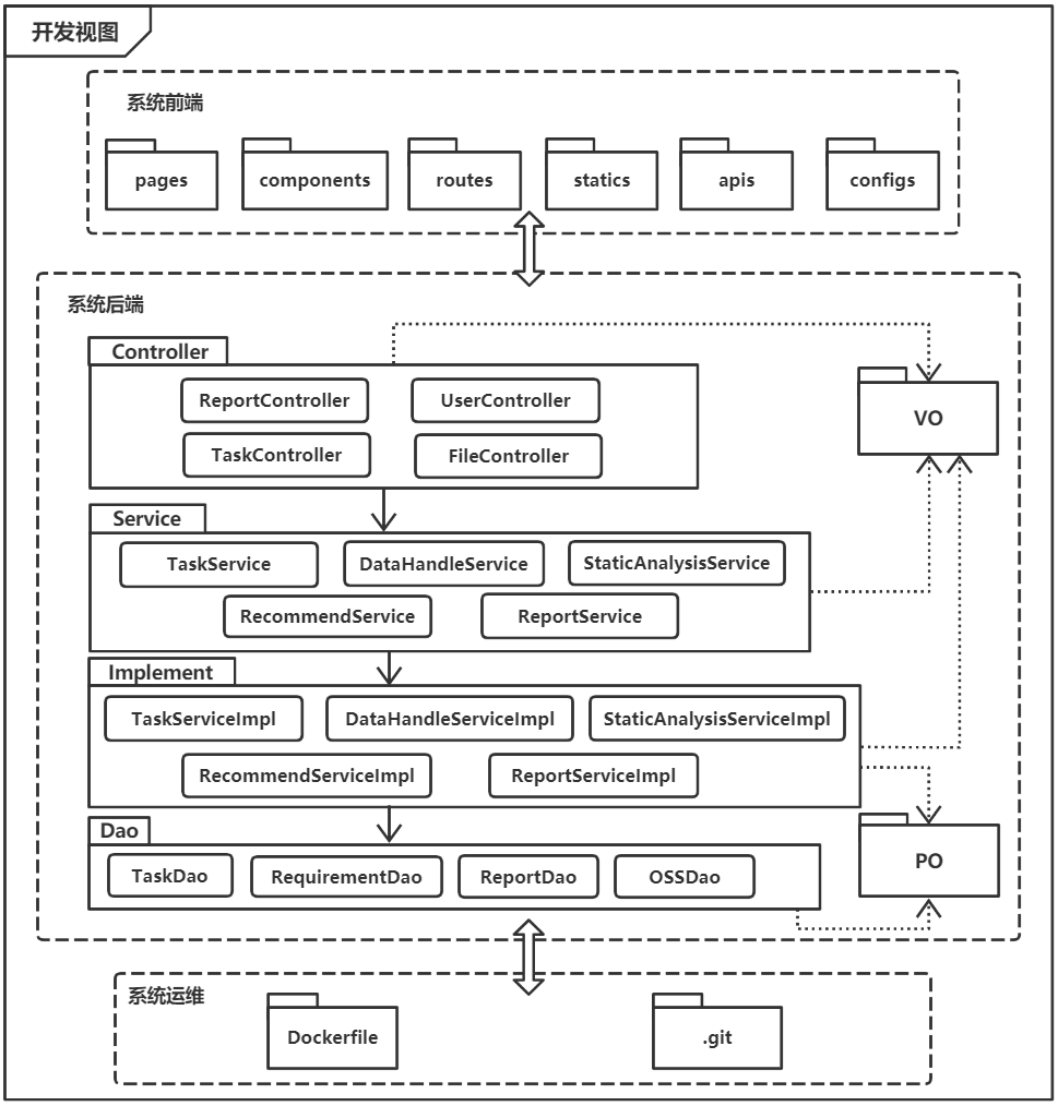


图 3.6: 开发视图

系统的开发视图如 3.6所示，它关注系统中静态层次与代码的组织结构，需要兼顾系统的功能性需求以及项目开发的维护性和可重用性。在系统的前端项目中，pages 包中存放各个前端各个页面的内容；components 是根据页面需求划分出来的各个组件；apis 包记录着系统前后端交互所有的接口；前端页面的跳转与路由通过 routes 包下的代码进行控制；configs 是系统前端项目的各项配置信

息，其涉及的静态资源存放在 statics 包下。后端项目主要采用了后端开发的经典分层架构。其中，Controller 包负责管理路由请求与 Service 调用；Service 包负责提供基础的功能接口，Implement 层是 Service 中复杂业务逻辑的具体实现；Dao 层则负责对连接数据库与进行持久化操作，OssDao 负责对阿里云 OSS 存储服务进行增删查改；VO 和 PO 是数据对象实例在不同业务层次中的不同形态，包括系统功能相关的所有 Java 实例类型。为了提高系统的可靠性与可扩展性，使用 Docker 和 GitLab 对系统进行部署和管理，Dockerfile 文件负责对系统进行容器打包，git 文件夹则记录着系统在 GitLab 上的具体配置和版本信息。

3.4　核心模块设计

3.4.1　自动化测试数据处理模块设计

自动化测试数据是由测试脚本和编程框架生成的日志与堆栈信息，数据繁杂，不具备直接阅读的条件，仅靠人力从自动化测试数据里面获取关键信息会消耗其大量的精力与时间。因此，需要通过程序对自动化测试过程中的测试数据进行处理，提取关键的测试操作和缺陷信息。

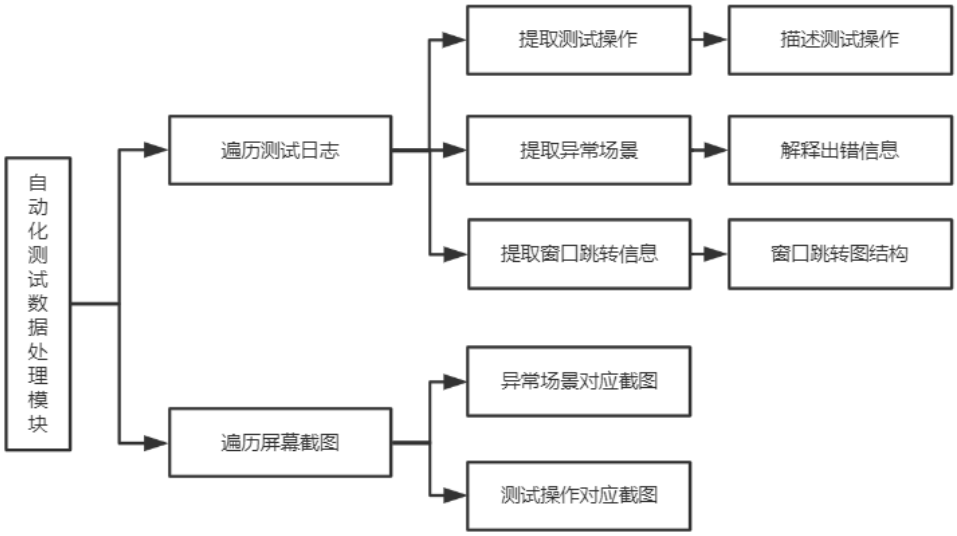


图 3.7: 自动化测试数据处理模块功能示意图

自动化测试数据处理模块的主要功能如图 3.7所示, 该模块负责对自动化测试数据进行处理，从繁杂的日志文件中提取出直观的数据，并将其封装成测试操作实体类型。借助文本和截图等辅助信息，引导众包工人执行相应的测试操作序列，复现自动化测试过程中出现的异常场景。为了能对安卓移动应用进行

初步的覆盖，提取其中的运行缺陷，本文采用了 M 平台提供的自动化测试框架。该框架对应用的窗口进行深度优先形式的遍历，对窗口的控件进行简单的操作。同时，按照一定的时间间隔进行截图，反映设备的运行情况。本模块将以此为前提，对自动化测试的数据进行处理。首先，模块从日志文件中提取出自动化测试脚本在设备上进行的所有操作，如点击和滑动等；补全测试操作中的位置信息，如所在窗口，是否触发窗口跳转等，还需要根据用户配置，对测试操作进行描述；接着，提取所有的窗口跳转信息，构造应用的页面转换连通图，计算从入口窗口到达各个窗口的最短路径；然后，模块将遍历自动化测试中所有的屏幕快照，通过时间优先的匹配规则，为每个测试操作以及异常场景匹配上对应的屏幕截图；最后，根据从数据中提取出的异常场景，补全测试事件和相关辅助信息，生成引导众包工人复现异常场景的众包需求。本模块的主要流程如下：

（1）提取自动化测试操作。系统将从自动化测试的数据中提取并存储每一个测试操作。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 3.15展示了一个测试事件 TestAction 中的关键信息。

表 3.15: 测试事件属性表

属性	说明	举例
time	时间戳	“2020-02-15 21:57:23,667”
type	事件类型	“CLICK”
activityBeforeAction	执行操作之前所在窗口	“ui.FirstActivity”
activityAfterAction	执行操作之后所在窗口	“ui.SecondActivity”
message	操作信息	“Click widget //android.view.View[contains(@bounds, [846,117][906,177]]”

（2）构造窗口跳转模型。因为窗口之间的跳转是相互的，一个窗口跳转到另一个新窗口的同时，也可以通过返回按钮返回；同时，窗口之间的跳转没有权重信息。因此，应用的窗口跳转结构模型是一个无向图，其中节点集合为已经覆盖的所有窗口，边集合为所有的触发窗口跳转的事件序列。当测试操作前后所在的应用窗口不一致时，则可视为发生了窗口跳转事件。遍历事件的同时对窗口信息维护一个邻接表，不断记录和更新窗口之间的跳转情况。图 3.8展示了某真实应用的窗口跳转图与对应邻接表。

（3）匹配屏幕截图。分析屏幕截图与测试操作的时间戳信息，如果测试操作的触发时间大于且最接近某张截图所对应的时间，则该截图可以与测试操作进行映射。设备截图和测试操作之间普遍存在着时延误差，同时也为了反映测试

操作执行前后的应用运行情况，本系统将对应截图的前续和后继截图都记录该测试操作中，即一个测试操作会携带三张屏幕截图。利用运行自动化测试的屏幕快照可以直观地向众包工人描述程序运行状态与操作反馈，有助于引导其对异常场景进行复现。

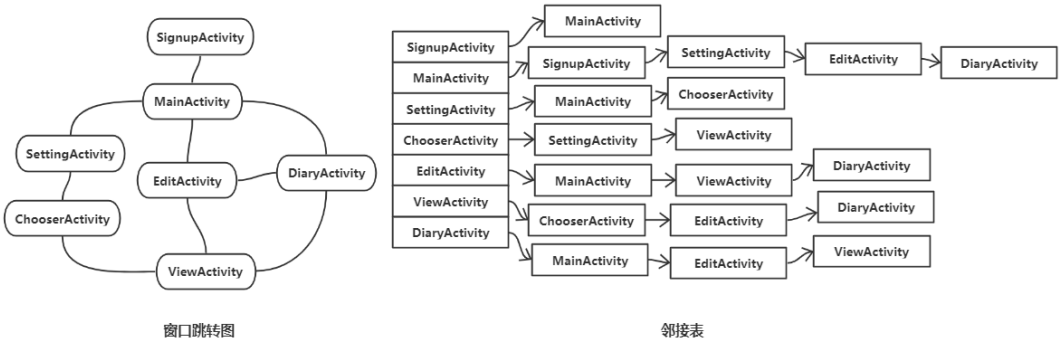


图 3.8: 窗口跳转与邻接表示意图

（4）生成引导复现异常场景的众包需求。基于已覆盖窗口的跳转模型以及测试操作事件序列，可以通过最短路径算法计算出从入口窗口到达各个窗口的最短路径。根据异常场景所在的窗口，提供到达该窗口的测试操作事件序列，并对异常信息进行描述，使用文本与截图相结合的方式引导众包工人进行复现。

3.4.2　安卓静态分析模块设计

为了解决自动化测试中覆盖率不足的问题，需要从应用源码入手进一步发掘众包测试的需求。安卓静态分析模块主要负责对源码进行扫描，找出源码中的跳转控制语句以及分支语句中涉及的控件信息，提取出自动化测试中未覆盖的窗口以及程序中涉及条件判断的控件类型。图 3.9展示了静态分析模块的主要功能，该模块主要的设计思路如下：

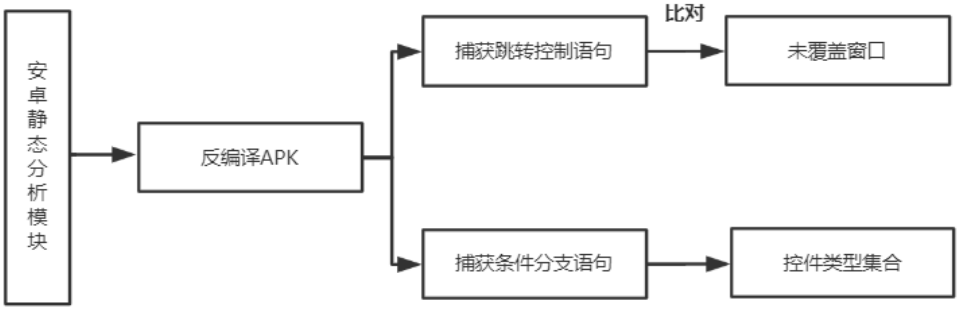


图 3.9: 安卓静态分析模块功能示意图

（1）反编译安卓 APK。与传统的计算机操作系统类似，安卓操作系统也有

其独特的运行环境与程序格式，它仅支持 APK 格式的程序安装包，其它操作系统的可执行文件无法在安卓上运行。APK 涵盖了移动应用在安卓系统上安装和运行的全部信息，包括源码、静态资源文件以及所依赖的第三方库等等。然而，这些数据都经过了编译、压缩和打包等阶段，无法直接获取。为了进行源码的静态分析，需要对用户提交测试时上传的 APK 文件进行反编译，获取应用的源码信息。本文选用开源工具 Jadx 对 APK 进行反编译，获取应用源码，为程序静态分析提供基础，具体的操作步骤将在系统详细实现章节进行阐述。

（2）捕获跳转控制信息。文中所提及的应用窗口，在安卓开发中对应的类为 Activity，这是一种特殊的数据类型。通常来说，每个 Activity 都对应着安卓移动应用中的一个窗口，主要用作与用户进行交互，因此一个安卓应用程序会存在单个或多个 Activity。用户从一个窗口跳转到另一个窗口，在 Android 代码里，是通过原窗口对应的 Activity 类中的代码信息进行控制的，主要的跳转方式有两种，分别为显式跳转和隐式跳转。显式跳转是通过 Intent 类型来直接指定 Activity 的跳转关系；隐式跳转是通过 Intent 指定 action 和 category 信息，符合这两个信息的 Activity，将作为跳转的目标，进而发生窗口跳转。表 3.16展示了安卓的窗口跳转控制语句示例。Intent 是 Android 程序中负责交互的类，通过 Intent 可以声明当前逻辑中所需执行的操作并携带相关的数据，如页面跳转、开启广播以及启动服务等。因此，只需在源码中对各个 Activity 文件进行遍历，匹配 Intent 类型相关的代码，即可对涉及的窗口跳转信息进行提取。

表 3.16: Android 跳转控制语句示例表

跳转方式	说明	举例
显示跳转	从 A 窗口跳转至 B 窗口	Intent intent = new Intent(A.this, B.class); startActivity(intent);
隐式跳转	跳转至 action 为 “A” 且 category 为 “C” 的窗口	Intent intent = new Intent(”A”); intent.addCategory(”C”); startActivity(intent);

（3）提取相关控件的类型。在 Android 代码中，控件信息往往十分模糊，缺少有价值的信息，例如安卓控件的 ID 是非必须的，也是非唯一的；在开发阶段也无需为控件提供相关的定位信息。在自动化测试中，缺少精准的描述信息是无法编写相关测试脚本的。然而，由众包工人去找到这些控件并不困难。每个窗口中的控件信息是有限的，可以通过提供窗口中控件的类型信息，借助众包工人的智慧，根据自身的使用经验对这些控件进行测试。该模块遍历所有窗口文件，捕获代码中的条件分支语句，并记录语句内相关的控件类型信息，即可达到目标。表 3.17列举了安卓移动应用中常见的控件类型。

表 3.17: Android 常见的控件类型

控件类型	说明	控件类型	说明
TextView	用于在界面上显示文本信息	SeekBar	控制窗口滑动
Button	作为按钮与用户进行交互	MenuItem	提供下拉菜单功能
RecycleListView	作为列表容器展示信息	DigitalClock	提供选择时间功能
EditView	输入各类信息的文本框	RadioButton	单选按钮
ImageView	用于展示图片的控件	CheckBox	多选按钮

（4）通过对已覆盖窗口以及整体窗口信息进行比对，可以获取到自动化测试没有覆盖到的窗口，引导众包工人到达未覆盖窗口的前置窗口，对未知窗口进行探索。同时，提供各个窗口下条件分支语句内的控件类型信息，鼓励众包工人发挥群体智慧,对控件进行等价类和边界值等测试,进一步提升测试覆盖率。

3.4.3　众包需求推荐模块设计

为了提高众包测试的执行效率，本文对众包需求的推荐进行了设计。通过分析用户的背景信息与历史选择记录，对众包需求按照一定的规则进行排序，优先展示适合众包工人的需求。众包需求推荐模块的核心主要为冷启动和协同过滤两个部分，图 3.10展示了本模块的主要功能。

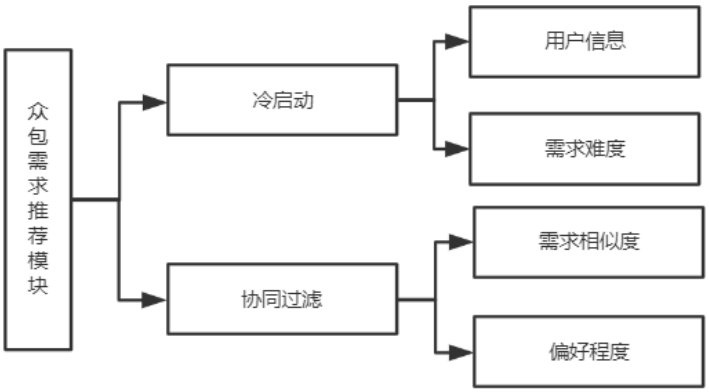


图 3.10: 众包需求推荐模块功能示意图

（1）冷启动。通常来说，众包测试所召集的众包工人缺少固定的组织和管理，每次都会有许多新的众包工人加入到测试任务中。这种类型的用户群体往往缺乏相应的历史行为记录。协同过滤算法根据挖掘用户之间的关联以及物品之间的相似性来完成推荐，因此它不适用于对新加入的众包工人和众包需求进行推荐，冷启动是无法回避的问题。通过冷启动可以主动向用户推荐物品，一方面可以根据用户的个人信息来进行推送，另一方面可以按照平台的策略，有偏向的

向用户推送优先度高的物品。本模块根据众包需求的困难程度以及众包工人的受教育程度进行冷启动。对于普通用户，优先展示简单的众包需求，激发他们的测试热情；对于具有相关测试知识的用户，优先展示困难程度较大，对系统造成较大影响的众包需求，充分发挥他们的背景知识与解决问题的能力。一个众包需求的困难度由严重程度、窗口深度和需求类型三个方面组成：

严重程度：严重程度是针对复现异常场景的众包需求而言，越严重的异常对用户的影响会越大。表3.18对 Android 应用程序的常见异常进行分类，按照异常的严重程度，对不同的异常赋予一定的分数，分数越高越则严重。

表 3.18: 安卓移动应用常见异常类型及严重程度

类型	异常	严重程度
资源类	ClassNotFoundException NoSuchFieldException NoSuchMethodException FileNotFoundException	6
操作类	NumberFormatException NullPointerException IndexOutOfBoundsException UnsupportedOperationException ArrayStoreException IllegalArgumentException	5
网络类	ConnectException SocketException NetworkError	4
响应类	ActivityTimeout ServiceTimeout SystemTimeout	3
线程类	InterruptedException IllegalThreadStateException	2
其它	IOExcetion ArithmeticException	1

窗口深度：根据整体窗口跳转模型，众包需求所在的窗口层级越深，窗口跳转次数越多，众包工人在进行测试时需要进行的前置步骤越多，整体来说该需求测试起来越困难。因此，窗口深度和众包需求的困难度成相关性。

需求类型：基于自动化测试处理模块与安卓静态分析模块，系统生成了三种形式的众包需求，包括引导复现异常场景、探索未覆盖窗口、测试具体窗口下的指定类型控件。对普通用户而言，这三种需求的执行难度是有区别的，测试指定类型的控件自由度较大，只需一些基本的操作，是三者中最易于完成的。复现异常场景只需要在不同设备上，按照引导信息进行操作，即可完成对异常的验

证，比较适合没有测试技能的普通用户。探索未覆盖窗口需要对控件进行更深入更全面的覆盖，需要花费更多的时间。因此，探索未覆盖窗口的需求复杂度最高、其次为复现异常场景的众包需求、最后为测试指定类型控件的众包需求。

众包工人的教育背景是冷启动的关注点之一，对于大专院校以上学历，且为计算机软件相关专业的众包工人，按照众包需求的困难值进行降序排序，优先推荐困难值相对较高的众包需求。这类众包工人具有软件测试的相关知识，结合需求引导信息，可以高效的进行众包测试，解决大部分相对困难的众包需求。对于普通众包工人，则按照困难值升序排列，优先展示较为简单的众包需求。

（2）基于物品的协同过滤推荐。协同过滤技术因其原理简洁、效果良好、实施方便等特性，得到了业界广泛的使用和认可。它根据用户的历史行为记录，挖掘出相似的用户或者物品，按照偏好程度的预测值选出最靠前的选项，向用户进行推荐。在众包测试中，根据用户群体对众包需求的选择情况，计算需求之间的相似度，筛选出最符合用户偏好情况的选项进行推荐是可行的。基于众包需求的协同过滤推算步骤如下：

首先需要计算众包需求之间的相似值。不同的众包需求之间的信息较为模糊，提取精确的特征进行计算比较困难，单纯参照冷启动流程中计算困难值的方式来进行会比较片面。为了计算众包需求的相似度，可以从不同用户对众包需求的选择情况入手。对于众包工人群体而言，如果选择了众包需求 A 的大部分人也选择了众包需求 B，则可认为众包工人对这 A 和 B 两个众包需求的兴趣程度相近，两个需求的相似度高。计算众包需求相似度公式如下：

$$w_{ij} = \frac{|N(i) \cap N(j)|}{\sqrt{|N(i) \cup N(j)|}} \tag{3.1}$$

w_{ij} 代表众包需求 i 和 j 的相似度。其中，$N(i)$ 和 $N(j)$ 分别表示在众包测试中选择了众包需求 i 和 j 的众包工人集合；$N(i) \cap N(j)$ 则代表同时选择了众包需求 i 和 j 的众包工人集合；$N(i) \cup N(j)$ 代表在两个众包集合的总人数。当同时选择两个众包需求的用户越多，则两者的相关性越强。

然后，预测众包工人对众包需求的偏好程度。计算众包工人对一个新众包需求兴趣程度的公式如下：

$$p_{uj} = \sum_{i \in N(u) \cap S(j,k)} w_{ji} r_{ui} \tag{3.2}$$

p_{uj} 代表众包工人 u 对众包需求 j 兴趣程度的预测值。其中 $S(j,k)$ 代表与众包需求 j 最相近的 K 个测试用例的集合；$N(u)$ 代表众包工人选择过的众包需求；i 是同时属于集合 N 和集合 S 中的众包需求。w_{ij} 代表众包需求 i 和 j 的相似度，可通过上一步的公式进行计算；r_{ui} 取值在 0,1 之间，如果众包工人对该需求表示过不感兴趣，则值为 0，其它情况取值为 1。

最后，按照用户对众包需求的兴趣预测值进行降序排序，通过分页和每页固定 10 个众包需求的方式，实现类 Top N 形式的众包需求推荐。处理好的众包需求以 JSON 数组为载体返回至众包测试审核平台。

3.4.4　可视化模块设计

图形界面是用户与系统进行交互的主要接口，后端的业务逻辑是系统功能的基础，但也需要为系统搭建进行功能操作与数据展示的界面。基于上文的需求分析，除了任务管理以外，系统需要对众包概况、需求详情、设备详情和众测数据进行展示。因此，系统对可视化模块进行了设计。在前后端分离的设计思想下，该模块通过前端技术进行数据的渲染与展示，后端负责处理业务逻辑以及组织返回的数据。前端通过符合 RESTful API 规范的接口向后端发送请求，后端逻辑执行完毕后返回相应的 JSON 格式数据，前后端之间的功能实现互不干扰。

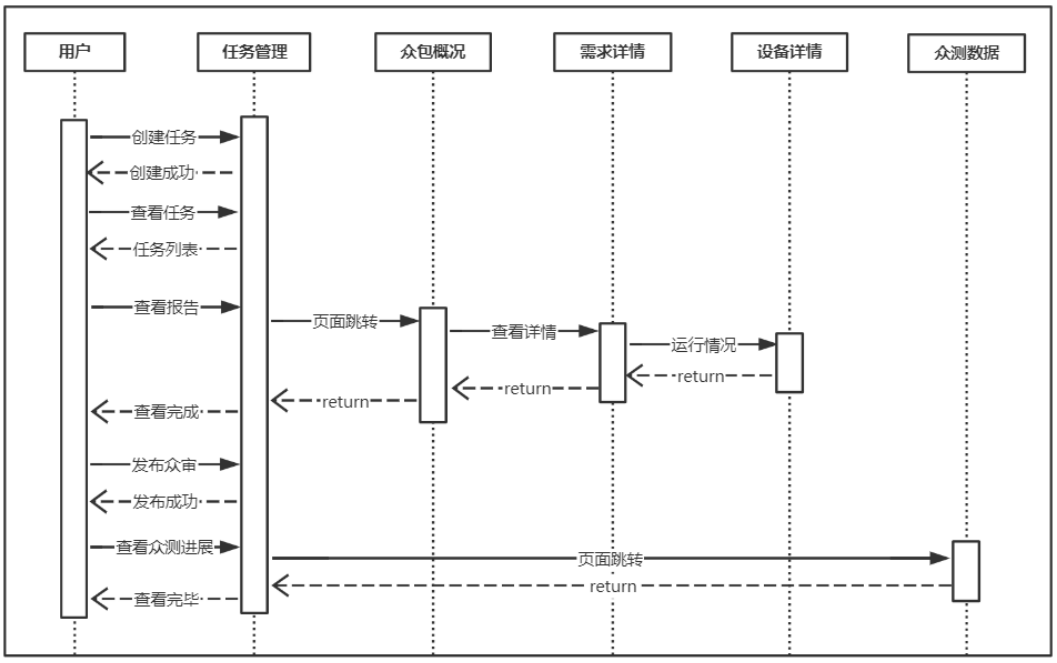


图 3.11: 可视化模块时序图

可视化模块的操作时序图如 3.11所示。由图可得，自动化测试驱动的众包需求生成系统提供的主要操作和界面如下：

（1）创建任务。已登录的用户需要上传待测应用安装包、填写应用名称、应用类型、应用描述等测试信息，然后提交测试任务。前端会发送 RESTful 请求，后端收到请求后，在 TaskSerivce 执行对应操作，创建具体的测试任务。

（2）查看任务。用户可以在任务管理界面查看自己提交的所有任务，展示任务执行状态，并提供查看报告和发布众审等后续操作。用户可以在此界面查看任务对应的测试报告以及将众包需求发布至众包测试审核平台投入使用。

（3）查看众包概况。任务执行完毕后，用户可以在“我的任务”界面查看报告信息，报告模块将发送请求至系统后端获取众包需求列表以及窗口跳转模型数据，对数据进行渲染和绘制。用户可以看到该次测试任务生成的众包需求列表和静态分析后的应用窗口跳转模型。

（4）查看需求详情。当用户点击某个具体的众包需求时，前端页面将通过路由跳转到详情界面，对后端执行 getTaskDetail 方法后返回的 JSON 数据进行渲染，在页面上展示需求的基础信息、异常截图以及具体的引导步骤。

（5）查看设备详情。用户在需求详情界面点击设备的型号信息，即可进入设备详情页面，该页面展示了对应设备的性能情况以及执行自动化测试过程中的运行截图和日志信息。

（6）发布众审。为了使众包需求投入使用，需要将其发布到众包测试审核平台。用户可以在任务查看界面点击“发布众审”选项，系统将通过执行 publish 方法，将众包需求提交至众包测试审核平台，进行后续众包测试操流程。

（7）查看众测数据。众包工人使用系统生成的众包需求开展测试后，系统会收集并汇总众包工人对众包需求进行评审的结果，通过众测数据界面反馈给众包请求发起者，使众包协作达到闭环的效果。

3.5　数据库设计

本系统采用非关系型数据库 MongoDB 实现数据的持久化存储。NoSQL 语言在处理复杂的数据类型时非常高效简洁，MongoDB 在各种环境下也拥有良好的性能表现。然而，数据库的设计需要平衡自由与约束之间的关系，否则也会造性能上的问题，对数据表的理解和维护也会带来影响。因此，需要在 MongoDB 高效自由存储的情况下，参照数据库范式对其进行一定的约束，具体如下：

（1）符合数据库的第一范式。第一范式要求数据表都有对应的实体，且该实体具有独立性，每个属性都不可再分，这是数据库表的初步设计思想。然而，仅符合第一范式设计的数据库表，会存在着数据冗余等问题，并且表的结构也会变得臃肿，对数据库的查询性能也会造成较大的影响。

（2）符合数据库的第二范式。第二范式要求每个数据库表对应的数据实体都有唯一的标识，不同实体之间通过该标识进行关联。实体的标识即为表的主键，不同的表之间通过外键确立关系。第二范式要求开发者对数据库表进行拆分，消除了一张表中数据冗余和内容过多的问题。

第一范式和第二范式是对数据库表进行设计的指导思想，它们要求数据库表和数据实体进行对应，并对表进行一定程度的拆分。如果不进行拆分，一个表中会有非常多的字段，不利于数据查询和存储。对于可以复用并具有独立和普遍意义的数据，可以拆分为一个表，例如应用信息表和设备信息表。同时，对于特别大的数据，如 APP 安装包和屏幕截图等，可以上传至阿里云 OSS 存储服务器，数据库表只存储对应的地址，降低系统的内存压力。然而，如果对数据库表进行无节制的拆分，在程序运行时将会频繁使用级联语句进行查询，严重影响系统性能。为此，对于一些辅助信息类型的数据，应该存放在同一张表中；对于特有的，不具有普遍意义的长数据，可以作为 text 类型进行存储。

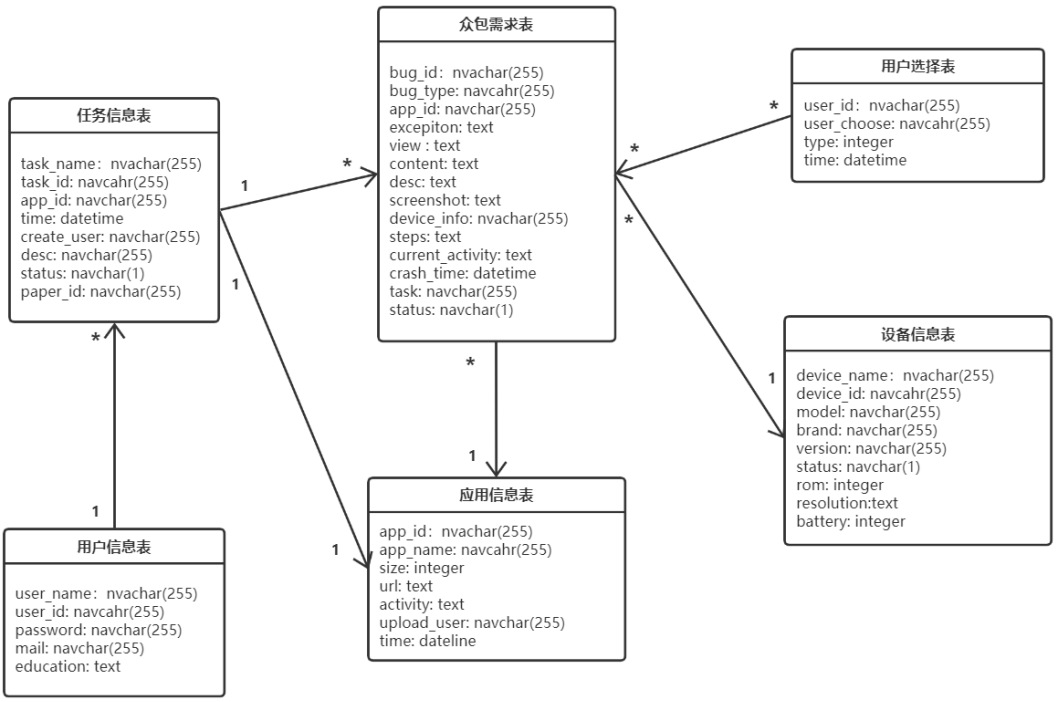


图 3.12: 系统实体关系设计图

系统的实体关系设计图如 3.12所示。系统主要对用户信息、任务信息、众包需求、应用信息、用户历史记录以及设备信息进行了存储。为此，本文将这些需要持久化的实体映射为数据表，构造了实体关系模型，用于描述和展示不同实体的属性与实体之间的关联。在数据库中，实体的属性对应着数据表的字段，不同的表之间通过外键进行联接。数据库的存取功能通过 MongoDB 实现，涉及

的数据表包括用户信息表、任务信息表、众包需求表、应用信息表、用户选择表以及设备信息表，下面将详细阐释各个数据表的功能功能和字段含义。

用户信息表主要用来记录系统功能所需的的基础用户信息，如表 3.19所示。本系统沿用 M 平台的用户系统，故此表仅列出了业务逻辑中涉及的部分用户信息，如用户名，用户邮箱以及用户教育背景等。用户信息表的主键为用户 ID，用于标识和区分不同的用户。

表 3.19: 用户信息表

字段	类型	默认值	说明
user_name	nvchar	NULL	用户名
user_id	nvchar	NULL	主键，用户 ID
password	nvchar	NULL	用户密码
mail	nvchar	NULL	用户邮箱
education	nvchar	NULL	教育情况

任务信息表用于记录用户所创建的任务信息，如表 3.20所示。一个测试任务除了基础的信息之外还包括对应的 App 信息，创建时间、创建人、任务描述以及任务状态等。为了节省系统内存空间，任务状态通过一个字符信息进行存储，0 代表未开始，1 代表执行中，2 代表执行完毕，3 代表执行失败。paper_id 用于记录该任务生成的众包需求发布众审后对应的众测试卷 ID，该属性将作为后续收集和统计众测试数据的参数。

表 3.20: 任务信息表

字段	类型	默认值	说明
task_name	nvchar	NULL	任务名
task_id	nvchar	NULL	主键，任务 ID
app_id	nvchar	NULL	任务对应的 APP
time	datetime	NULL	创建时间
create_user	nvchar	NULL	创建者
desc	nvchar	NULL	任务描述
status	nvchar	NULL	任务状态
paper_id	nvchar	NULL	众测试卷 ID

应用信息表记录了用户上传的 APP 信息，如表 3.21所示。包括名称、大小、时间和上传者等。activity 字段记录了系统静态分析得出的应用窗口跳转模型，通过 TEXT 字段存储。url 字段记录了 APP 的安装包下载地址。由于应用安装包的大小一般在 10M 以上，对于数据库与系统内存来说是个很大的负担。因此，本

系统将用户上传的 APP 安装包通过阿里云的 OSS 存储服务器进行存放，数据库负责存储对应的 URL 地址，在保证功能的同时减轻了本地服务器的负担。

表 3.21: 应用信息表

字段	类型	默认值	说明
app_id	nvchar	NULL	主键，应用 ID
app_name	nvchar	NULL	应用名称
size	integer	0	应用大小，单位为 M
url	text	NULL	应用安装地址
activity	nvchar	NULL	应用窗口跳转模型
upload_user	nvchar	NULL	上传者
time	dateline	NULL	上传时间

众包需求表记录系统生成的众包需求信息，包括对应的 APP，发生异常的设备信息、异常描述、控件信息、所在窗口复现步骤等信息，如表 3.22所示。同上，众包需求的截图信息存放在阿里云的 OSS 存储服务器中，数据库只存储对应的 URL 地址。本文对该表进行了拆分，关键的信息通过外键进行关联，降低了表的大小，提高查询速度。

表 3.22: 众包需求表

字段	类型	默认值	说明
bug_id	nvchar	NULL	主键，需求 ID
bug_type	nvchar	NULL	需求类型
app_id	nvchar	NULL	应用 ID
excepiton	text	NULL	异常信息
view	text	NULL	相关控件
content	text	NULL	内容信息
desc	text	NULL	描述信息
screenshot	text	NULL	截图地址
device_info	nvchar	NULL	设备信息
steps	text	NULL	引导信息
current_activity	text	NULL	当前窗口
crash_time	datetime	NULL	异常发生时间
task	nvchar	NULL	对应任务
status	nvchar	NULL	需求状态

用户选择表记录了用户选择过的众包需求信息，如表 3.23所示，包括用户 ID，用户选择的需求 ID 以及选择时间。type 字段记录了用户对需求的偏好类型，如果用户对需求标记过“不感兴趣”，则该字段为 0。

表 3.23: 用户选择表

字段	类型	默认值	说明
user_id	nvchar	NULL	用户 ID
user_choose	nvchar	NULL	选择的需求 ID
type	integer	1	偏好类型
time	datetime	NULL	选择日期

设备信息表主要用于记录本系统与 M 平台共同维护的自动化测试设备的基础信息，如下表 3.24所示，包括设备型号、设备品牌、设备版本、设备使用状态、内存空间、屏幕分辨率以及设备电量等信息。

表 3.24: 设备信息表

字段	类型	默认值	说明
device_name	nvchar	NULL	设备名
device_id	nvchar	NULL	主键，设备 ID
model	nvchar	NULL	设备型号
brand	nvchar	NULL	设备品牌
version	nvchar	NULL	设备版本
status	nvchar	NULL	设备状态
rom	integer	0	内存大小
resolution	text	NULL	屏幕分辨率
battery	integer	100	设备电量

3.6　本章小结

本章针对系统的业务目标，确认了系统的功能性需求与非功能性需求，采用例图和用例描述的形式对其进行阐述。基于上述分析结果，参照业界流行的前后端分离架构，对系统整体进行了设计。根据“4+1”视图模型，从系统功能、模块通信、代码组织以及物理部署等角度对系统整体架构进行展示。接着，进一步划分出了系统的各个主要功能模块，包括自动化测试数据处理模块、安卓静态分析模块、众包需求推荐模块以及可视化模块，论述了各个模块功能实现的原理、技术要点及合理性。最后，将系统涉及的数据实体映射为数据库表，通过实体关系模型展示了不同表之间的关系，描述了数据表中各个字段的含义。

第四章　　系统详细设计与实现

本章将基于上文的系统需求分析与架构设计，从细节上阐述各个主要模块的详细设计与实现方案，具体包括自动化测试数据处理模块，安卓静态分析模块、众包推荐模块以及可视化模块。最后，通过真实的使用案例和运行截图对系统使用流程以及主要功能进行展示。

4.1　自动化测试数据处理模块的详细设计与实现

自动化测试数据处理模块主要涉及提取自动化测试操作、构造窗口跳转邻接表、匹配屏幕截图以及生成引导复现异常场景的众包需求等过程，图 4.1展示了自动化测试数据处理模块的主要功能及流程：

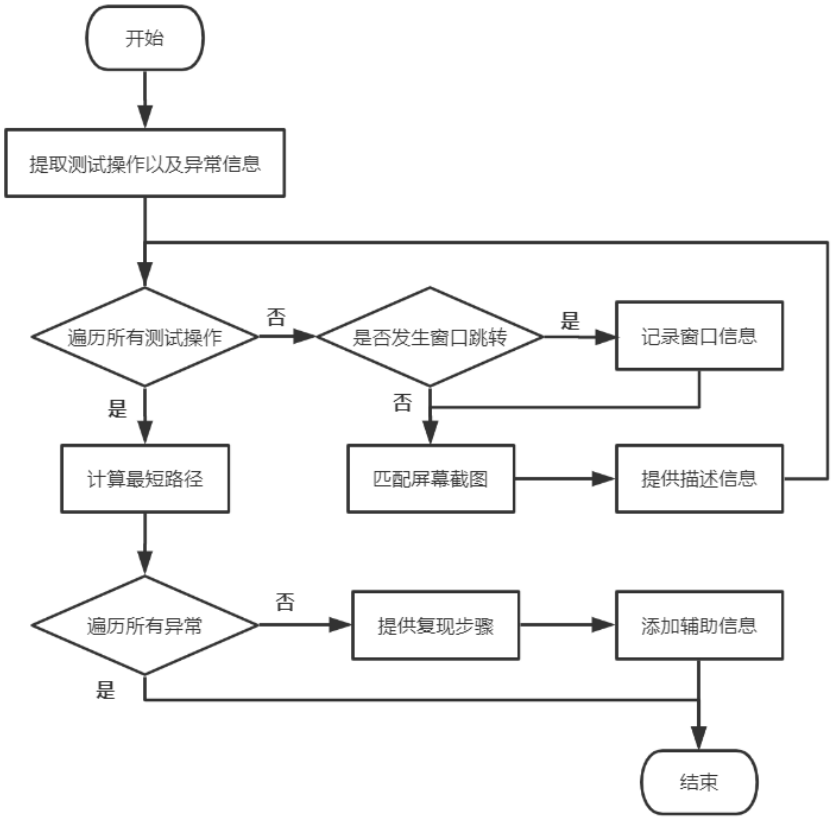


图 4.1: 自动化测试数据模块流程图

本模块需要从测试数据中提取自动化测试的所有操作，并将信息补充完整，

这是引导用户复现的基础之一。然后，根据触发窗口跳转的事件序列，构造出窗口跳转图，并计算出从入口到达各个窗口的最短路径，引导众包工人快速到达异常场景。最后，将自动化测试过程中的屏幕截图与测试操作进行匹配，向众包工人提供直观的测试操作反馈，展示程序运行状态。

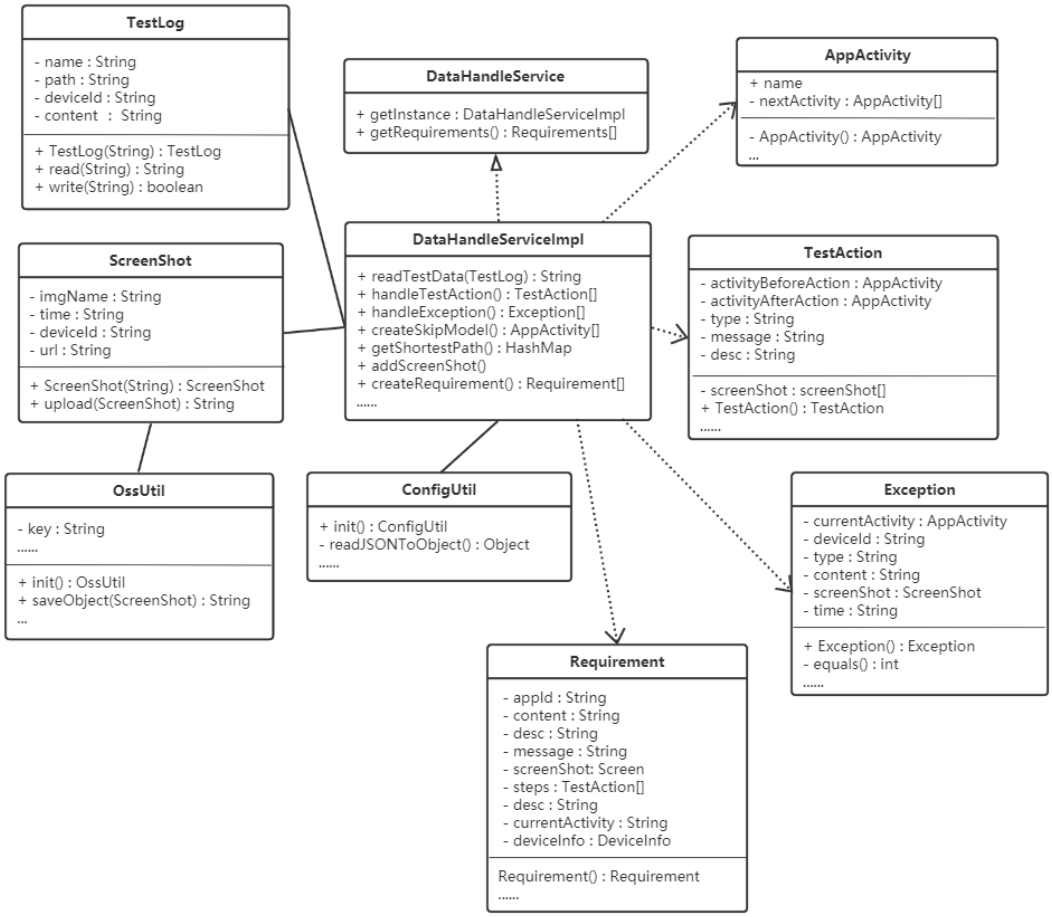


图 4.2: 自动化测试数据模块核心类图

图 4.2为自动化测试数据处理模块的核心类图。DataHandleService 是本模块负责对外提供的服务，其业务逻辑通过 DataHandleServiceImpl 进行实现，具体的接口包括 readTestData、handleTestAction、handleExcepiton、createSkipModel、getShortestPath、addScreenShot 以及 createRequirement。其中，readTestData 负责从原始的日志文件中读取数据日志实例 TestLog；通过 handleTestAction 和 handleExcepiton 提取日志中的测试操作以及异常信息，将 JSON 数据实例化为 TestAction 和 Exception 类型。createSkipModel 遍历触发窗口跳转的事件序列，构造应用窗口跳转模型；通过 getShortestPath 计算出从入口到达各个窗口的最短路径。addScreenShot 为测试操作和异常场景匹配了对应的屏幕截图。OssUtil 实现

了与阿里云 OSS 服务进行增删改查的逻辑，高效进行对象存储并获取返回网络 URL，降低系统内存消耗。同时，通过 ConfigUtil 工具类可以读取用户的各种自定义补充配置，使辅助信息更全面。

4.1.1　提取自动化测试操作

图 4.3展示的是提取自动化测试操作的关键实现代码。为了对自动化过程中产生的脚本日志进行分析，首先需要从测试文件中读取数据，保存为字符串数据流，封装成相关类型实体，作为后续操作的基础。

```
public ArrayList<TestAction> handleTestAction (String path){
        // 读取测试脚本日志
        ArrayList<TestLog> testLog = readTestData(path);
        ArrayList<TestAction> testActions = new  ArrayList<TestAction>();
        // 对原始数据格式进行处理
        String [] actions = testLog.format("}");
        for(int i=0; i<actions.length; i++){
                actions[i]+="}";
                actions[i]=actions[i].replace("\\n","");
                extract("time","type","activity","message" ,actions[i]);
                testActions.add(JSON.parseObject(actions[i], TestAction.class));
        }
        for(int i=0; i<testActions.length; i++){
                TestAction action = testActions.get(i);
                if(action.type != null)
                        // 省略不同操作类型处理代码
                 if(!action.activityAfterAction.equals(action.activityBeforeAction))
                        // 省略窗口跳转事件处理代码
        }
        return testActions;
}
```

图 4.3: 提取自动化测试操作关键代码

readTestData 是本文实现的读取测试日志的工具函数，该函数通过集成 InputStreamReader 等实例的功能，将 TXT 以及 LOG 类型的文件统一按照 UTF-8 编码进行读取和保存，path 参数代表了自动化测试数据的文件夹路径信息。对于原始的字符串类型数据，按照规则进行分割、替代和格式化处理；然后，提取测试上下文中的触发时间、事件类型、前后窗口、事件描述等信息；最后，通过阿里巴巴提供的 JSON 处理库 fastjson 将处理后的原始数据实例化为 TestAction 实例。不同的测试操作类型需要提供不同的语义化描述，用户可以根据测试需求通过外置的 JSON 文件统一配置。当测试操作触发前后所在的窗口不一致时，

视为应用发生窗口跳转，收集此类事件为后续的处理步骤提供基础。

4.1.2　构造窗口跳转模型

应用的窗口变化情况可通过无向图进行表示。其中，邻接表是一种内存空间消耗较低的存储结构。通过构造邻接表来存储应用窗口跳转模型并作为该模型的点集合，窗口跳转事件作为模型的边集合，借助 Dijkstra 算法即可计算出应用窗口跳转模型中的单源最短路径，即从入口窗口到达各个窗口的所需的最短测试操作。图 4.4展示了构造窗口跳转模型的关键代码。

```
public ArrayList<AppActivity> createSkipModel (ArrayList<TestAction> testActions) {
        int length = testActions.size();
        ArrayList<AppActivity> activities = new ArrayList<AppActivity>();
        // 遍历测试操作
        for(int i=0; i<length; i++){
                TestAction action = testActions.get(i);
                AppActivity appActivity = new AppActivity(action.activityBeforeAction);
                if(!activities.contains(appActivity)) {
                        // 如果该 Activity 不在邻接表中，则加入
                        activities.add(appActivity);
                }else{
                        // 如果 Activity 在邻接表中已存在，则获取实例
                        appActivity = activities.get(activities.indexOf(appActivity));
                }
                // 提取窗口发生跳转的事件
                if (!action.activityBeforeAction.equals(action.activityAfterAction)
                        && !action.activityAfterAction.equals(".Launcher")){
                        AppActivity temp = new AppActivity(action.activityAfterAction);
                        // 省略判断检测部分代码
                        // 将新 Activity 加入到当前 Activity 的 nextActivity 中
                        appActivity.nextActivity.add(temp);
                        // 更新邻接表
                        activities.add(temp);
                }
        return activities;
}
```

图 4.4: 构造窗口跳转模型关键代码

邻接表以 ArrayList 的形式进行存储，数组中每个元素为 AppActivity 类型。AppActivity 是类指针结构，其 nextActivity 属性记录着与该窗口相邻的其它窗口。在遍历测试操作的过程中，读取测试操作的 activityBeforeAction 属性，获取其当前所在窗口，如果该窗口尚未加入邻接表数组，则对邻接表数组进行更新；

如果已经加入邻接表，则获取窗口对应的实例。对于触发窗口跳转的测试操作，先对邻接表进行更新，最后将其添加到原窗口的 nextActivity 属性中。

4.1.3　匹配屏幕截图

为了直观的展示设备端的执行状态和程序响应，需要给测试操作与异常场景匹配对应的屏幕截图。根据截图和事件的时间戳信息，选择触发时间上最相近的两者作为配对项。匹配屏幕截图的关键代码如图 4.5所示。

```
for (int i=0; i<testActions.size(); i++) {
   TestAction testAction = testActions.get(i);
   // 处理日期格式
   String time = dateToStamp(testAction.timeBeforeAction,"yyyy-MM-dd HH:mm:ss ");
   for(int j=0; j<screenShots.size(); j++) {
      // 处理截图时间戳
      String temp = screenShots.get(j).split("_")[1].split("\\.")[0];
      // 对用户操作涉及区域进行处理
       dragRectangle(screenShots.get(j));
       if(Long.parseLong(time) <= Long.parseLong(temp)) {
          testAction.screenShot = new ArrayList<String>();
          testAction.screenShot.add(screenShots.get(j));
          //  省略处理前续和后继截图代码
          break; }
   }
}
```

图 4.5: 匹配屏幕截图关键代码

dateToStamp 方法是封装好的日期处理函数，其首先将跳转事件的执行时间与截图文件中的日期信息转化为时间戳。通过转换为 Long 型数据，实现时间戳之间的数值比对。如果屏幕截图的日期大于当前测试操作的执行时间，且与其最接近，则该截图可以视为测试操作对应的屏幕快照。通常来说，测试操作与截图进程存在着时延误差。因此，将对应截图的前继与后续截图都加入了截图列表中，即一个测试操作携带三张屏幕截图。

4.1.4　生成引导复现异常场景的众包需求

根据上述步骤获取的窗口跳转模型与测试操作事件序列，通过提取到达异常场景所在窗口的最短路径，补全辅助信息，使用文本和截图相结合的方式，引导众包工人在不同设备上对应用异常进行复现。生成引导复现异常场景的众包需求关键代码如图 4.6所示：

```
public ArrayList<Requirement> createRequirements(String path,
                        HashMap<String, LinkedList<TestAction>> shortestPath) {
        ArrayList<Exception> exceptions = handleException(path);
        ArrayList<Requirement> requirements = new ArrayList<Requirement>();
        for(int i=0; i<exceptions.size(); i++){
                Exception exception = exceptions.get(i);
                Requirement requirement = new Requirement(exception);
                // 提供复现步骤
                task.steps = shortestPath.get(requirement.currentActivity);
                // 匹配屏幕截图
                addScreenShot(requirement);
                // 通过自定义 JSON 配置辅助信息
                addBugDesc(requirement);
                requirements.add(requirement);
        }
        return requirements;
}
```

图 4.6: 生成引导复现异常场景的众包需求关键代码

handleException 方法从移动应用运行的堆栈文件中搜索以“Caused by”开头的信息，提取自动化测试中触发的异常场景，并通过其构造原始的的众包需求实例，包括异常内容和设备信息等属性。存储最短路径的数据结构 shortPath 是一个 Hash 映射表，传入窗口名称，即可获得从入口窗口到达异常窗口的最短路径，即达到此窗口已知的步骤最少的测试操作。通过 shortPath 可以为每个异常提供到达其所在窗口的测试操作事件序列。最后，为异常场景补充屏幕截图，并通过读取用户自定义的 JSON 配置文件，提供辅助信息。

4.2　安卓静态分析模块的详细设计与实现

为了提高众包测试对安卓移动应用的覆盖率，除了对自动化测试中触发的异常进行验证以外，还需要对安卓移动应用的源码实现进一步分析，提取更多的众包需求。用户在提交测试任务时，可以提交应用的完整安装包信息，即应用源码；也可以提交源码编译打包后的 APK 文件。安卓静态分析模块首先对用户提供的 APK 文件进行反编译，获取应用源码文件。安卓是基于事件驱动的图形界面程序，通过对用户事件进行监听以及绑定对应的处理逻辑来实现各种功能。因此，只需遍历安卓应用程序 GUI 相关的窗口代码文件即可完成静态分析

工作。在遍历文件的过程中，搜索源码中的窗口跳转控制语句和条件分支语句，捕获其对应的代码内容，提取相关信息；最后，引导用户对自动化测试中未覆盖的窗口以及涉及条件判断的控件类型进行探索。安卓静态分析模块的具体功能及流程如图 4.7所示。

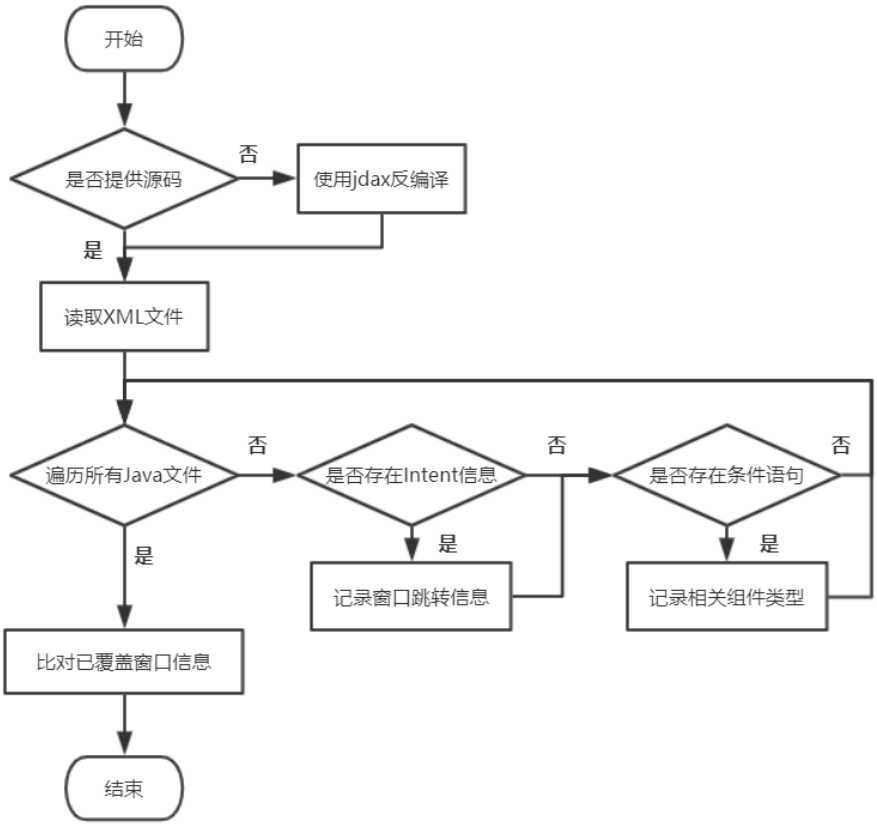


图 4.7: 安卓静态分析模块流程图

图 4.8展示的是安卓静态分析模块的核心设计类图。

其中，StaticAnalysisService 负责处理静态分析请求以及获取负责提供具体实现的实例 StaticAnalysisServiceImpl，最终通过 getRequirement 返回需求列表。StaticAnalysisServiceImpl 提供了 getApk、decompile、scanActivity、scanViews 等接口。具体来说，getApk 负责根据任务的具体信息，确定应用的 ID 与版本号，借助 OssUtil 实例，从阿里云 OSS 存储服务中下载 APK 文件至本地服务器；decompile 方法通过系统命令行的方式调用 Jadx 对 APK 文件进行反编译，提取安卓应用窗口相关的 Java 源代码文件，并实例化为 CodeFile 数组；在遍历分析源码的过程中，通过 scanActivity 和 scanViews 方法捕获窗口跳转控制语句与条件分支语句; 最后，提取出自动化测试中未覆盖的窗口以及涉及条件判断的控件类型，引导用户对其进行探索。

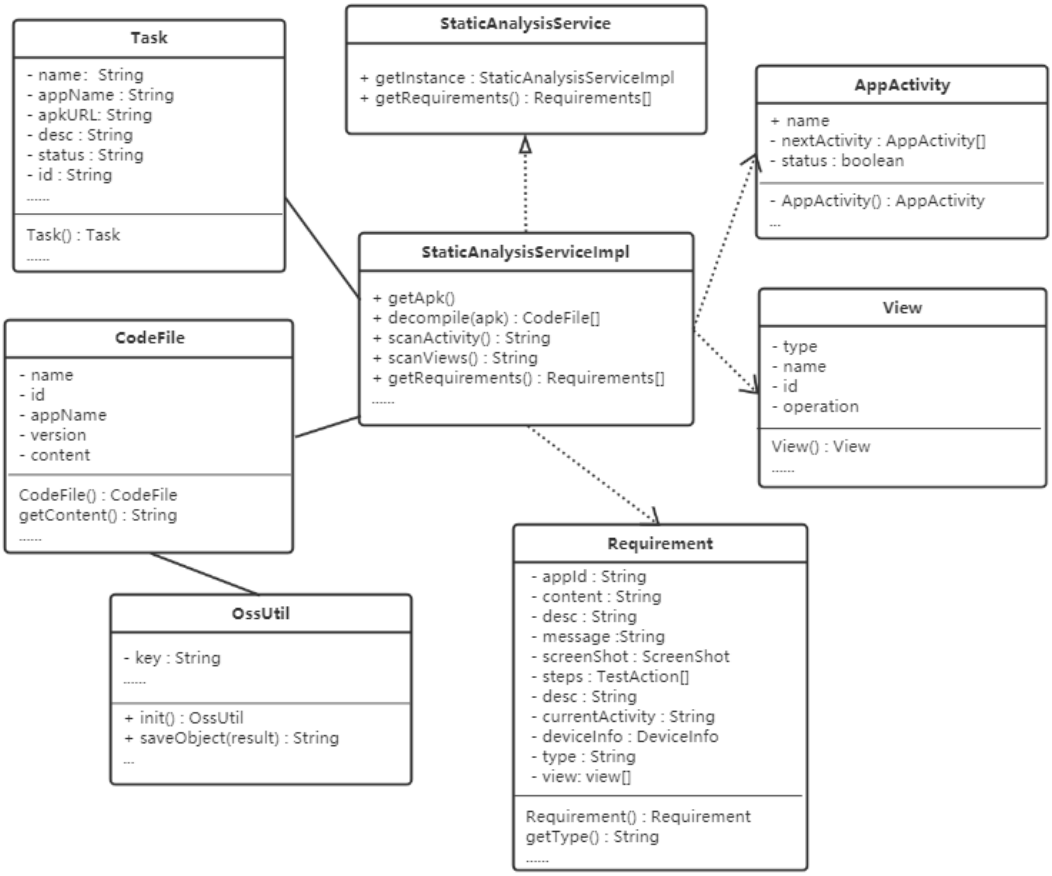


图 4.8: 安卓静态分析模块核心类图

4.2.1　反编译 APK

用户在提交测试任务时，可以提交应用源码，也可以提交编译后的安装文件 APK。如果为后者，为了进行静态分析，需要对 APK 文件进行反编译步骤，获取应用对应的源码。Jadx [1]是目前业界主流的开源安卓逆向调试工具，它同时支持通过命令行和图形界面进行使用，能以最简便的方式完成 APK 文件的反编译操作。借助 Jadx 能够获得安卓应用程序的所有代码以及静态资源文件。本文以开源安卓应用“JianShi”为例，调用 Jadx 图像化界面工具对其 APK 文件进行了反编译，图 4.9展示了该应用进行反编译后的 MainActivity 文件。阅读图中 MainActivity 的代码，可以发现两处关于此窗口的显式跳转，分别为跳转至 EditActivity 以及 DiaryListActivity，恰好验证了使用 Jadx 进行反编译的合理性以及下文将介绍的窗口扫描的可行性。

除图形界面外，在系统中使用 Jadx 提供的命令行工具也可以达到同样的效

[1]https://github.com/skylot/Jadx

果，这为系统在代码中调用 Jadx 对安卓 APK 文件进行反编译提供了基础。具体的代码如图 4.10 所示。

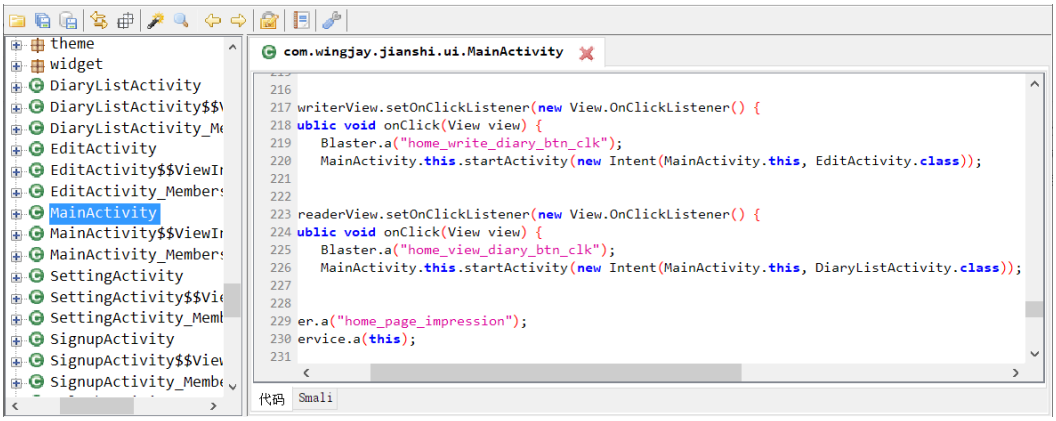


图 4.9: Jadx 图形化界面工具使用效果图

```
BufferedReader br = null;
StringBuilder sb = null;
try {
    // 获取命令行进程实例
    Runtime cmd = Runtime.getRuntime();
    // 执行 jdax 命令
    Process process = cmd.exec("jadx -d " + apkName + " -o " + targetPath);
    // 获取实例输入流
    br = new BufferedReader(process.getInputStream());
    // 获取命令行执行完成后的输出
    String line = null;
    sb = new StringBuilder();
     while ((line = br.readLine()) != null) {
        sb.append(line + "\n");
    }
} catch (Exception e) {
    e.printStackTrace();
    br.close(); }
// 省略后续文件处理代码
```

图 4.10: 反编译 APK 关键代码

Runtime 是 Java 提供的用于启动执行程序的类型，通过此类可以设置启动参数、环境变量和工作目录。Runtime 执行后会返回一个用于管理操作系统进程的 Process 对象。在 Linux 操作系统下，getRuntime 返回的系统进程对象实

例 cmd 可以看作输入命令的终端。cmd 实例通过执行 exec 方法，并传入相关命令行指令与参数，即可完成需求。*–d* 参数指定需要进行反编译的 APK 文件，*–o* 参数指定了反编译数据的输出文件夹路径。上述代码中传入的参数意为对 appName 指定的 APK 进行反编译，任务完成的数据将存储到 targetPath 路径的文件夹下。执行完相关命令行指令后，需要将命令行的显示数据作为该方法的返回数据。BufferedReader 是 Java 中包装和读取字符流的类，通过命令行的输入流创建，即可读取数据。br 将根据命令行的返回逐行读取，读取到的数据暂存在 StringBuilder 的实例中，直到数据读取完毕。对进程实例进行操作可能触发异常，因此无论如何都要在程序返回之前将读取数据的 br 管道关闭，否则将造成内存泄漏。最后，处理 targetPath 下的源代码文件，并实例化为 CodeFile 型数组。

4.2.2　安卓源码扫描

源码扫描负责对安卓应用源码进行分析，获取窗口跳转与条件分支语句相关的代码。如第三章所述，安卓应用的窗口跳转分为显式跳转和隐式跳转两种，显式跳转直接指定了跳转的目的窗口，而隐式跳转则是指定了窗口的 action 和 category 属性，拥有对应属性的窗口将作为目标窗口进行跳转。而本文提及的分支条件语句，即是代码中常见的 if、while 以及 for 循环等代码块。

```
ArrayList<CodeFile> codeFiles = decompile(appName,targetPath);
for(int i=0; i<codeFiles.size(); i++){
        // scanActivity 与 scanViews 内容合并展示
        String rows[] = codeFiles.get(i).getContent("row");
        for(int j=0; j<rows.length; j++){
                String patternA = "new Intent\\((.+?)\\)";
                String patternB = "addCategory\\((.+?)\\)";
                // 判断是否收集到 Intent 的信息
                if(Pattern.matches(patternA, rows[j]) || Pattern.matches(patternB, rows[j]))
                  // 省略后续处理代码
        }
        String patternC = "(if|else|while|for)?\\((.+?)\\)\\{(.+?)\\}";
        Pattern r = Pattern.compile(patternC);
        Matcher m = r.matches(codeFile.get(i).getContent("full"));
        while(m.find())
              // 省略后续处理代码
}
```

图 4.11: 源码扫描关键代码

为了简单高效匹配代码中指定的语句，本系统将使用正则表达式来完成。匹

配的关键部分代码如图 4.11所示。Activity 的跳转语句是在一行内编写的，因此需要将文件逐行读入，同时为了提高效率，读入后的每一行将通过 StringBuilder 汇总起来，汇总后的数据将用于条件分支的匹配，减少文件的遍历次数。对于每一行，使用正则表达式匹配语句中的 new Intent 以及 addCategory 中小括号的内容，如果匹配成功，则对该行后续处理。对于条件分支语句来说，其内容通常是跨行的，因此需要匹配 if、else、while 和 for 关键字后的花括号内容，匹配成功后则将捕获到的数据进行后续处理，包括去除一些多余字符以及格式化操作等。

基于上述步骤，模块提取了应用源码中的窗口层级结构信息后，通过与自动化测试中已覆盖窗口进行比对，可以得到未覆盖窗口的集合，根据上文的最短路径提供到达未覆盖窗口的前置窗口的测试操作序列，使用文本和截图相结合的方式，即可生成探索未覆盖窗口的众包需求；对于在条件分支语句内提取的相关控件类型，若所在窗口已经覆盖，则引导用户到达该窗口，提供需要测试的控件类型，要求众包工人使用等价类或边界值等方式对控件进行覆盖；若未覆盖则参照探索未覆盖窗口的众包需求进行处理。

4.3　众包需求推荐模块的详细设计与实现

众包需求推荐模块负责对上一章节中设计的推荐方案进行实现，将系统生成的众包需求按照一定的规则进行排序，优先展示适合用户的众包需求。该模块提供冷启动和协同过滤两个功能。系统首先对用户进行历史记录的检查，根据数据库中的用户选择表查询该用户是否有进行众包测试的记录。如果用户是首次进入该系统，则对其进行冷启动流程，包括获取用户的教育背景信息以及对众包需求进行困难度计算。对于大专院校学历背景，且专业为计算机软件相关的用户，系统将推荐难度较高的需求；否则优先向用户推荐较为简单的众包需求。如果用户曾经参加过众包测试，则对众包需求进行相似度计算，并根据用户的历史选择情况对其进行偏好预测，最终按照偏好的预测值进行降序排序。系统将排序后的众包需求以每页 N 个的形式进行分页处理，通过 JSON 数组的形式返回，以达到类 Top N 的推荐效果。

4.3.1　冷启动

当新的众包工人加入时，系统并没有其对众包需求的历史行为数据，无法进行需求推荐。因此，系统需要对用户执行推荐冷启动的流程。本系统针对用户进行的冷启动，将根据用户的教育背景信息判断其专业程度，对众包需求的各项属性参照不同的权重进行加权计算，并将其按照从困难到简单的顺序进行排

列。实现冷启动步骤的关键代码如图 4.12所示。

```
for(int i=0; i<requirements.size(); i++) {
        int weight = 0;
        Requirement requirement = requirements.get(i)
        // 根据需求的的类型计算权重
        weight += getTypeWeight(requirement.getType);
        if(requirement.getType().equals(TYPE_GUIDE_EXCEPTION)) {
                // 根据异常严重程度计算权重
                weight += getExceptionWeight(requirement.getContent());
          }
         // 根据页面层级计算权重
         ArrayList<TestAction> testActions = shortestPath.get(requirement.getActivity());
         weight += testActions.size();
         requirement.setWeight(weight);
}
Collections.sort(requirements,compare);

// 自定义 compare 方法
public int compare(Requirement a, Requirement b) {
        if(a.getWeight() > b. getWeight()) {
                return -1;
        }else if(a.getWeight() < b. getWeight() ) {
                return 1;
        }else {
                return 0; }
}
```

图 4.12: 冷启动实现关键代码

代码首先获取需求列表，并对列表进行遍历。在上文的系统需求与设计章节中，表 3.18列出了不同情况下异常对应的权重值。对于每一个需求，使用 getTypeWeight 方法获取类型对应的权重值。对于引导复现异常场景的需求，使用 getExceptionWeight 获取不同异常类型对应的权重值。shortestPath 方法用于计算需求所在窗口对应的最短路径，路径的长度代表了窗口在应用中的层级和深度，因此将直接作为权重值计算。每个需求的权重值为上述三者之和。为了使用权重值 weight 作为排序的依据，需要提供一个用于 Task 类权重比较的 compare 方法，当参数 a 的权重比参数 b 的权重大时，则返回一个小于零的负数。对于 Java 而言，负数代表降序排序，因此 a 排在 b 的前面。调用 Java 的 Collection 类中提供的 sort 方法，并传入自定义的 compare 方法即可对需求进行排序。

4.3.2　基于协同过滤的众包需求推荐

协同过滤推荐负责对已经拥有历史记录的众包工人进行推荐，合理的推荐算法可以在一定程度上提高众包测试的执行效率。为此，本文采用基于物品的协同过滤算法，对众包需求进行推荐，具体实施步骤如下：

（1）计算众包需求之间的相似度。不同的众包需求之间的信息较为模糊，权重难以确定，提取精确的特征进行计算比较困难，不能按照冷启动流程中的方式进行计算。因此通过用户对众包需求的选择情况来进行相似度计算。计算众包需求相似度的关键代码如图 4.13所示。

```
double getSimilarity(requirement a, requirement b) {
        ArrayList<User> usersA = getUsers(a);
        ArrayList<User> usersB = getUsers(b);
        // 计算同时选择需求 a 与需求 b 的众包工人集合
        int a_and_b = 0;
        for(int i=0; i<userA.size(); i++) {
                User user = usersA.get(i);
                for ( int j=0;i<usersB.size();j++){
                        if(user.equals(usersB.get(j))){
                                a_and_b += 1;
                                break; }
                }
        }
        // 计算同时选择需求 a 和 b 的人数占比作为相似度
        double w = a_and_b / Math.sqrt(usersA.size() - 1 + usersB.size() - 1);
        return w;
}
```

图 4.13: 计算相似度关键代码

其中，getUsers 方法用于获取选择对应需求的所有用户，该代码的底层实现是数据表的级联查询，即对上文数据库设计中提及的用户选择表进行查找。对于需求 a 来说，需要计算选择了 a 的用户里面同时选择了需求 b 的用户个数，即需求 a 和需求 b 的用户交集，记为 a_and_b。最后，计算 a_and_b 与用户集合总人数开平方根的比值，比值越大，则代表众包需求 a 和 b 相似程度越高。

（2）预测众包工人对其它众包需求的偏好程度。针对用户选择众包需求的历史记录集合 N，对于需求 A，通过 KNN 算法获取与其最相近的 K 个众包需求集合 S。计算同时在集合 S 与集合 N 中的众包需求，并计算这些需求与需求 A 之

间的相似度，通过对这些相似度进行累加获取最终的预测值。如果众包工人曾对某众包需求标记为不感兴趣，则无论选择与否，该需求权重都设为 0。预测偏好程度的关键部分代码如下文图 4.14所示：

```
for(int i=0;i<requirements.size();i++){
        // 用户历史选择记录
        ArrayList<Requirement> N = findRequirements(userOne);
        // 与当前需求最相似的众包需求
        ArrayList<Requirement> S = findNeighbor(requirements.get(i));
        // 计算 N 与 S 的交集
        ArrayList<Requirement> N_and_S = new ArrayList<Requirement>();
        for(int j=0;j<N.size();j++) {
                for(int k=0;k<S.size();k++) {
                        if (N.get(j).equals(S.get(k))) {
                                N_and_S.add(N.get(j));
                                break; }
                }
        }
        // 对相似度进行累加
        double prediction = 0.0;
        for(int j=0;j<N_and_S.size();j++) {
                if(!isUserDislike(userOne,N_and_S.get(j))) {
                        prediction += getSimilarity(tasks.get(i),N_and_S.get(j));
                }
        }
        // 省略后续处理代码
}
```

图 4.14: 计算偏好程度预测值关键代码

首先分别调用 getTask 以及 getTaskNeighbor 方法，基于 M 平台的统一用户系统，分别用于获取用户所选择过的众包需求集合 N 以及当前需求 i 最邻近的 K 个需求集合 S。获取最邻近集合的算法基于第三方开源算法库 sklearn 进行实现，故不在此进行展开阐述。通过对集合 N 与集合 S 进行双重遍历，获取两个集合中都存在的元素，记为 N_and_S，即是集合 N 与集合 S 的交集，然后计算交集中的每个需求与当前需求的相似度。如果用户对某个众包需求曾标记不感兴趣，则该需求的计算系数为 0，在代码中可映射为忽略该需求，仅对有记录的需求进行计算。最后，通过调用上文阐述的 getSimilarity 方法获取相似度并进行累计，计算最终的偏好程度预测值 prediceiton。在此基础上，调用系统提供的数组排序方法，以 prediceiton 为比较目标对众包需求进行降序排序。

4.4　可视化模块的详细设计与实现

基于上一章节设计的系统操作时序图，可视化模块负责与前端项目协作，实现相应的 Web 端图形界面，以供用户在本系统上进行各项操作。Web 端的一个页面分别对应着三个代码文件，包括负责界面结构的 Pug 模板引擎文件、负责页面业务逻辑的 JS 文件以及负责页面样式的 Less 文件。系统通过 Pug、Less 以及 JSON 中的数据负责对界面进行渲染和绘制，让用户在浏览器上看到各种功能页面。用户在界面进行操作时，会触发 JS 文件中对应的业务逻辑；接着，前端通过符合 RESTful API 规范的接口向服务端发送请求；最后，服务端对各项功能进行处理，以 JSON 为载体返回响应数据。

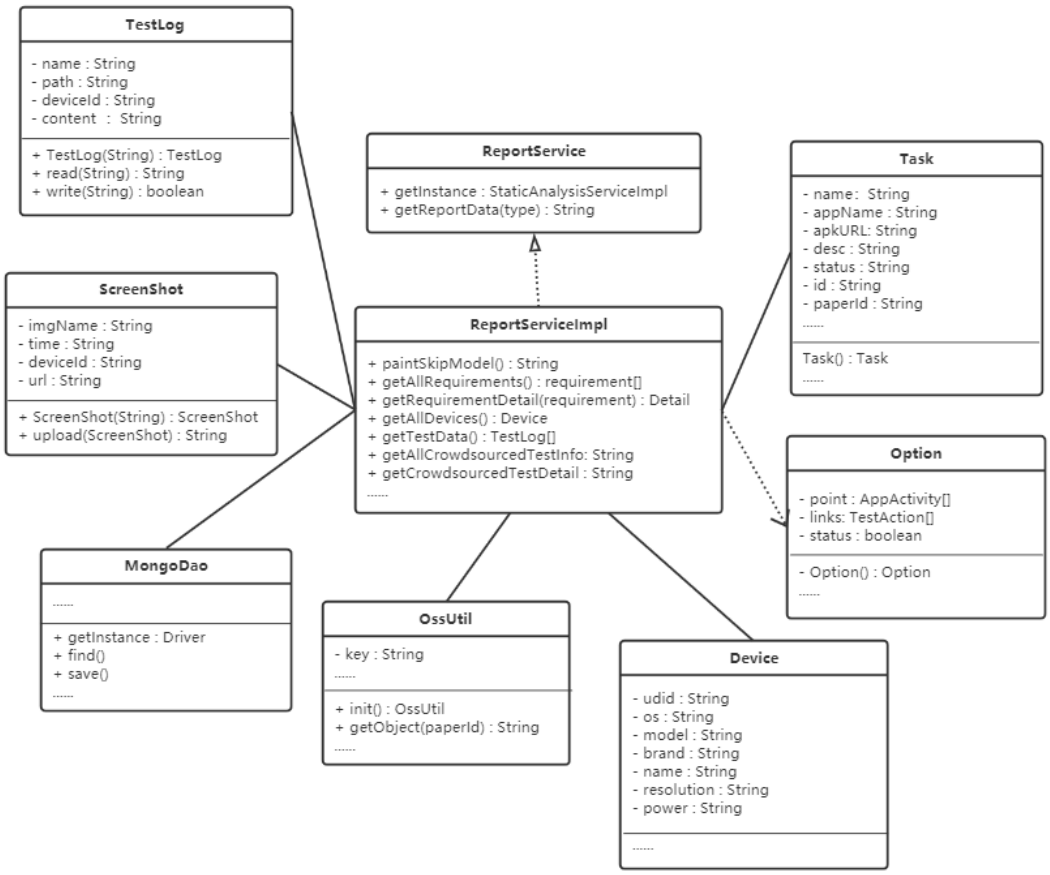


图 4.15: 可视化模块核心类图

图 4.15展示了可视化模块服务端的核心类图。

ReportService 提供处理可视化请求的接口，通过 getReportData 对不同的请求进行处理，并根据请求携带的 type 参数返回相应的 JSON 数据。ReportServiceImpl 是可视化业务功能的具体实现，提供了 paintSkipModel、getAllRequirement、getRequirementDetail、getAllDevices、getTestData、getAllCrowdsourcedTestInfo、

getCrowdsourcedTestDetail 等方法。其中，paintSkipModel 将窗口跳转模型映射为符合 ECharts 的需求数据格式，并将其实例化为 Option 类型；getAllRequirement 用于查询测试任务对应的所有众包需求；系统根据数据库范式进行了表的拆分，通过 getRequirementData 将众包需求与详细信息进行关联；getAllDevices 查询该次测试任务涉及的所有设备信息，并通过 getTestData 补全自动化测试过程中的各项数据，如测试日志、屏幕截图等；getAllCrowdsourcedTestInfo 查询使用生成的众包需求开展的众包测试场基本信息; 最后，通过 getCrowdsourcedTestDetail 方法对众包测试的评审数据进行收集和更新。

4.4.1　绘制窗口跳转模型

为了直观展示目标安卓移动应用的窗口跳转模型，需要手动将存储着该结构的邻接表映射为 ECharts 图表库所需的数据格式。图4.17展示了绘制窗口跳转模型的关键代码。

```
public Option paintSkipModel(ArrayList<AppActivity> activities) {
        // 省略实例声明代码
        for (int i=0; i<activities.size();i++) {
                // 处理模型结点集合
                if(!activityMap.contains(appActivity)) {
                        activityMap.add(appActivity);
                        ActivityPoint point = new ActivityPoint();
                        point.setName(appActivity.getName());
                       // 省略坐标设计代码
                       point.setIndex(x,y);
                       option.activityPoints.add(point);

                       // 处理模型边集合
                       for(int j=0; j<appActivity.nextActivity.size(); j++) {
                               Links link = new Links();
                               link.setSource(appActivity);
                               link.setTarget = (appActivity.nextActivity.get(j));
                               option.links.add(links); }
        // 省略后续处理代码
        return option;
}
```

图 4.16: 绘制窗口跳转模型关键代码

如图所示，paintSkipModel 提供了将邻接表抽像为关系图数据的操作。其中，参数 activities 数组的每一个元素是类指针结构，存储着与之联通的各个窗

口；activityMap 是集合类型的实例，通过集合的互异性可以防止重复设置窗口结点。按照窗口的出现次序设计关系图中的具体坐标，避免结点重叠；然后，遍历窗口的边集合，为关系图中的结点添加连线。ECharts 接收到 option 实例后将对关系图进行动态绘制与渲染。

4.4.2　众包需求列表与详情

getAllRequirement 是获取众包需求列表的接口，它接收任务 ID 作为参数，返回此次众包服务对应的众包需求列表与基本信息，包括需求类别、异常信息、触发原因和所在窗口等。getRequirementDetail 根据众包需求的唯一标识 ID，通过 MongoDB 进行级联查询，获取众包需求的详细信息，包括涉及的设备信息、异常描述和截图以及复现步骤等。数据的存取操作通过 MongoDao 以及 OssUtil 实现。图4.17展示了获取某次任务的众包需求列表与详情关键代码。

```
// 省略实例声明代码
requirements = MongoDao.findRequirements(taskId);
for(int i=0; i<requirements; i++){
        // 获取涉及的测试设备信息
        devices = MongoDao.findDevices(requirements.get(i).id);
        // 获取详请信息
        detail = MongoDao.findDetail(requirements.get(i).id);
        // 下载截图信息
        imgs = OssUtil.findObject(detail.getScreenShots());
        requirements.get(i).setInfo(devices,detail,imgs);
        // 省略后续处理代码
}
// 将 requirements 实例序列化为 JSON 数据
JSON.toJSONString(requirements, SerializerFeature.DisableCircularReferenceDetect);
```

图 4.17: 众包需求列表与详情关键代码

众包需求列表和详情展示内容较多，占用范围较大，因此需要拆分为两个页面，通过前端路由与需求 ID 进行控制跳转，用户点击具体的众包需求可以跳至需求详情页面。为了降低 HTTP 将请求次数与进行缓存策略，从数据库中获取的列表和详情数据将融合进一个 JSON 中，通过一次请求返回。

4.4.3　设备运行情况与测试信息

设备详情用于展示当前测试设备的运行情况，包括运行内存、内存占用情况、电池发热情况、网络流量情况、CPU 占用情况、冷启动时间、测试持续时

间。除此以外，还提供了该设备进行自动化测过程中试的日志信息与屏幕截图。图4.19展示了获取设备运行情况与测试信息关键代码。

```
// 省略实例声明代码
deviceList = MongoDao.findDevices(taskId);
for(int i=0; i<deviceList; i++){
        // 从堆栈文件中提取设备性能情况
        info = performanceResult(stackLog,device.udid);
        // 格式化运行日志
        logs = format(testLogs,device.udid);
        // 获取测试运行截图
        imgs = OssUtil.findObject(device.udid);
        device.setContent(info,logs,imgs);
}// 省略后续处理代码
```

图 4.18: 设备运行情况与测试信息关键代码

findDevices 方法通过传入的 taskId，查询出测试任务对应的设备列表以及基本信息；performanceResult 方法从测试的堆栈信息中提取设备进行自动化测试过程中的性能表现情况；format 方法对运行日志进行格式化处理，逐行输出日志内容，包括触发时间、事件类别、事件标签、事件内容等。最后，将上述内容填充至设备实体中。

4.4.4　众测数据收集与汇总

经本系统生成的众包需求可发布至 M 公司的众包测试审核平台投入使用，组织和开展众包测试。在此基础上，系统对众包测试的数据进行了收集和汇总，为用户提供了查看众测结果的入口，以期达到测试闭环的效果。图4.28展示了众测数据收集与展示关键代码。

getAllCrowdsourcedTestInfo 和 getCrowdsourcedTestDetail 是收集与展示众测数据的 API 接口。与上述功能一致，测试概况与评审详情被拆分为两个页面，通过前端路由动态渲染与控制跳转。当众包需求被发布后，系统会存储众包测试审核平台返回的 paperId。在此基础上，根据 paperId 获取对应众包测试数据，并实例化为 crowdsourcedTestVO 对象。为了兼容多次进行众包测试的情况，crowdsourcedTestVO 被设计为数组类型。在遍历的过程中，通过 handleCrowdsourcedTestInfo 方法提取众包测试的基本信息，包括场次名称、场次 ID、场次描述、起始时间、结束时间、参加人数、审核次数等。job 实例代表进行评审的众包需求，checkItems 为其对应评审信息；由于返回的 JSON 数据 key 值代表评审

问题，对应的 value 值为答案，因此需要通过反射的方式获取具体的信息。

```
// 省略实例声明代码
// 获取众包测试数据并实例化
result = OssUtil.getObject(paperId);
crowdsourcedTestVO = JSON.parseObject(result, CrowdsourcedTestVO.class)
for(int i=0; i<crowdsourcedTestVO; i++) {
        // 提取测试基本信息
        handleCrowdsourcedTestInfo(crowdsourcedTestVO.get(i));
        // 省略中间处理代码
        checkItems = job.reportInfo.getCheckItems();
        for(int j=0; j<checkItems.length; j++) {
        // 处理评审项
        Field[] fields=checkItem.getClass().getDeclaredFields();
        questions = fields.getName();
        answers = fields.getFieldValueByName();
        // 省略后续处理代码 }
}
```

图 4.19: 众测数据收集与展示关键代码

上文对系统主要功能模块的详细设计与实现进行了阐述，下面将通过系统的运行截图对系统操作流程和相关界面进行展示。

4.5　系统案例与运行截图

本小节以京东 App 的开源 Bate 测试版本为例，提供系统运行截图，以更直观展示系统操作流程与运行状态，本系统的核心功能如下：

（1）任务管理。用户登录本系统，点击左侧边栏的“发布任务”选项，选择“App 自动化测试”，进入任务提交界面；上传待测应用安装包，并勾选“众包协同测试”选项，填写相关信息，提交测试任务。图 4.20展示了系统的提交任务界面。提交任务后，点击侧边栏中“我的任务”选项，查看自动化测试任务执行状态。待任务执行完成后，可以点击查看测试报告。用户点击“发布众审”选项后，此次生成的众包需求将发布到众审平台，进行开展众包测试的后续处理。

（2）查看测试报告。用户进入报告页面后，可以看到此次测试任务的基本情况，如图 4.21所示。报告首先展示了应用的各项基本信息，如应用名称、应用版本、测试设备数量、设备覆盖率以及测试时长等，接着通过 ECharts 的饼状图表

展示了测试中发现的 Bug 的各种类型与等级占比情况。用户可以对此次测试任务的结果以及应用存在的问题有大致的了解。

图 4.20: 提交任务界面截图

图 4.21: 测试基本情况

（3）查看众包概况。在提交自动化测试任务前，如果用户勾选了“众包协同测试”选项，则最后的移动测试报告中会新增额外的众包报告页面，该页面对众包需求的生成情况进行展示。用户点击导航栏中的“众包概况”选项，查看众包需求报告。需求报告涵盖了当前应用的窗口层级结构图以及众包需求列表，点击需求的详情按钮，可以查看其具体的信息。

应用窗口层级结构图是上文静态分析的结果，如图 4.22 所示，该部分反映了应用的层级结构以及各个窗口之间的连通情况，应用的入口为 PrivacyActivity，通过入口可以到达的窗口为为 WelcomeActivity，以此类推。窗口之间可以到达即可通过 Return 按键返回，因此此图为无向图。用户可以在首次接触该应用的情况下，对窗口的跳转情况以及应用深度有所了解。

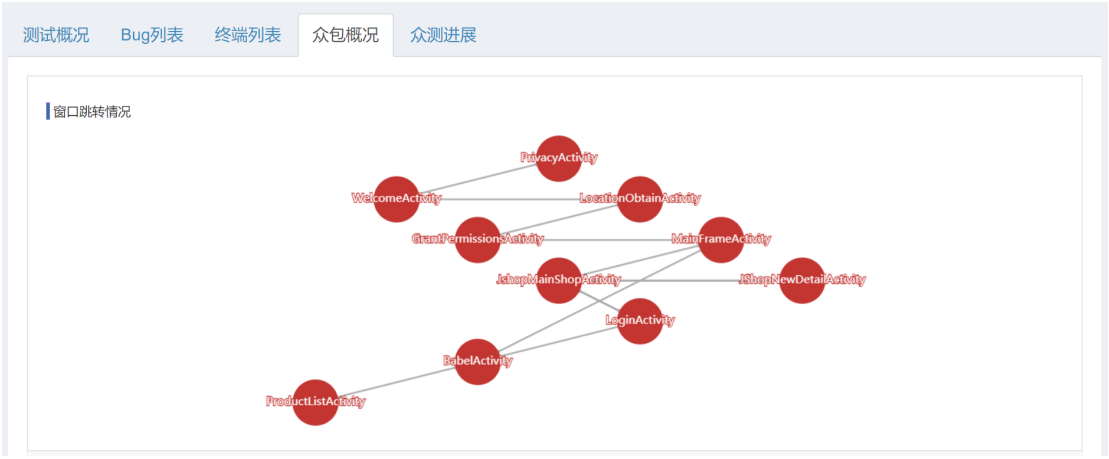
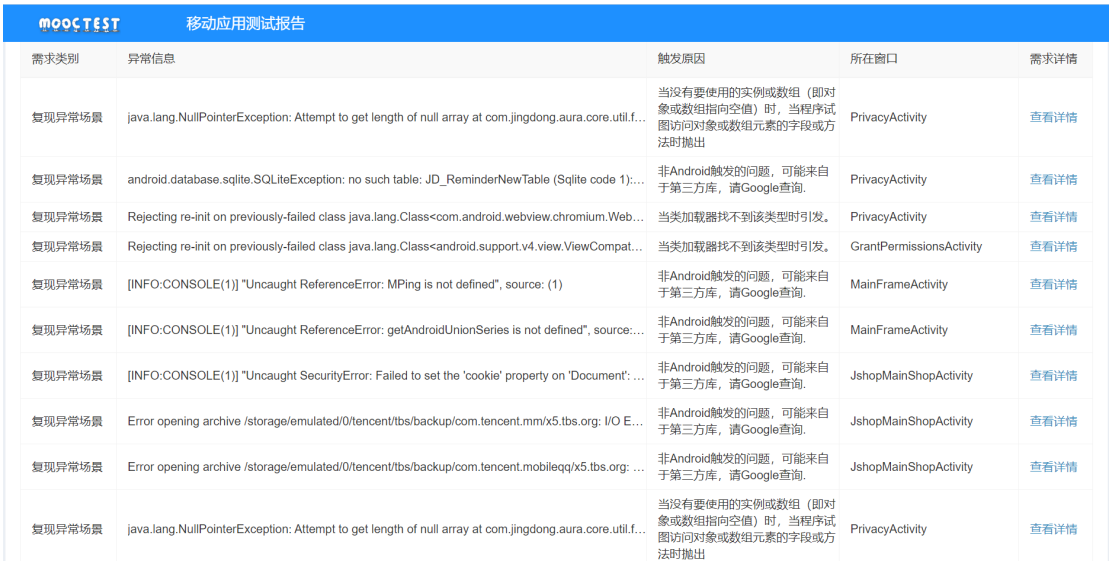


图 4.22: 应用窗口层级结构

众包需求列表是针对该次自动化结果与源码分析后生成的所有众包需求，该表中展示的信息包括需求类别、异常信息、所在窗口等，点击右侧“查看详情”标签会跳转至需求对应的详情界面。



需求类别	异常信息	触发原因	所在窗口	需求详情
复现异常场景	java.lang.NullPointerException: Attempt to get length of null array at com.jingdong.aura.core.util.f…	当没有要使用的实例或数组（即对象或数组指向空值）时，当程序试图访问对象或数组元素的字段或方法时抛出	PrivacyActivity	查看详情
复现异常场景	android.database.sqlite.SQLiteException: no such table: JD_ReminderNewTable (Sqlite code 1):…	非Android触发的问题，可能来自于第三方库，请Google查询.	PrivacyActivity	查看详情
复现异常场景	Rejecting re-init on previously-failed class java.lang.Class<com.android.webview.chromium.Web…	当类加载器找不到该类型时引发。	PrivacyActivity	查看详情
复现异常场景	Rejecting re-init on previously-failed class java.lang.Class<android.support.v4.view.ViewCompat…	当类加载器找不到该类型时引发。	GrantPermissionsActivity	查看详情
复现异常场景	[INFO:CONSOLE(1)] "Uncaught ReferenceError: MPing is not defined", source: (1)	非Android触发的问题，可能来自于第三方库，请Google查询.	MainFrameActivity	查看详情
复现异常场景	[INFO:CONSOLE(1)] "Uncaught ReferenceError: getAndroidUnionSeries is not defined", source:…	非Android触发的问题，可能来自于第三方库，请Google查询.	MainFrameActivity	查看详情
复现异常场景	[INFO:CONSOLE(1)] "Uncaught SecurityError: Failed to set the 'cookie' property on 'Document': …	非Android触发的问题，可能来自于第三方库，请Google查询.	JshopMainShopActivity	查看详情
复现异常场景	Error opening archive /storage/emulated/0/tencent/tbs/backup/com.tencent.mm/x5.tbs.org: I/O E…	非Android触发的问题，可能来自于第三方库，请Google查询.	JshopMainShopActivity	查看详情
复现异常场景	Error opening archive /storage/emulated/0/tencent/tbs/backup/com.tencent.mobileqq/x5.tbs.org: …	非Android触发的问题，可能来自于第三方库，请Google查询.	JshopMainShopActivity	查看详情
复现异常场景	java.lang.NullPointerException: Attempt to get length of null array at com.jingdong.aura.core.util.f…	当没有要使用的实例或数组（即对象或数组指向空值）时，当程序试图访问对象或数组元素的字段或方法时抛出	PrivacyActivity	查看详情

图 4.23: 众包需求列表

（4）查看需求详情与设备详情。用户在需求列表中点击某个需求的需求详情后，系统会打开新的页面，对需求进行更详细的展示。

页面顶部是触发异常的设备信息，包括设备编号、品牌和分辨率等。图 4.24展示了某台测试设备的基本信息。用户可以使用同样的设备进行测试，检查该异常是否真实存在；也可以在不同设备不同环境上执行同样的操作，检查该异常是否具有普遍性。

图 4.24: 设备信息截图

为了进一步展示测试信息，用户点击设备编号，可以查看设备在该次测试任务中的详细信息，包括运行内存、内存占用情况、网络流量情况、电池发热情况、CPU 占用情况、冷启动时间、测试持续时间的信息。此外，系统还提供了在该应用在设备上进行自动化测试的运行日志与屏幕截图，如图 4.25 所示。

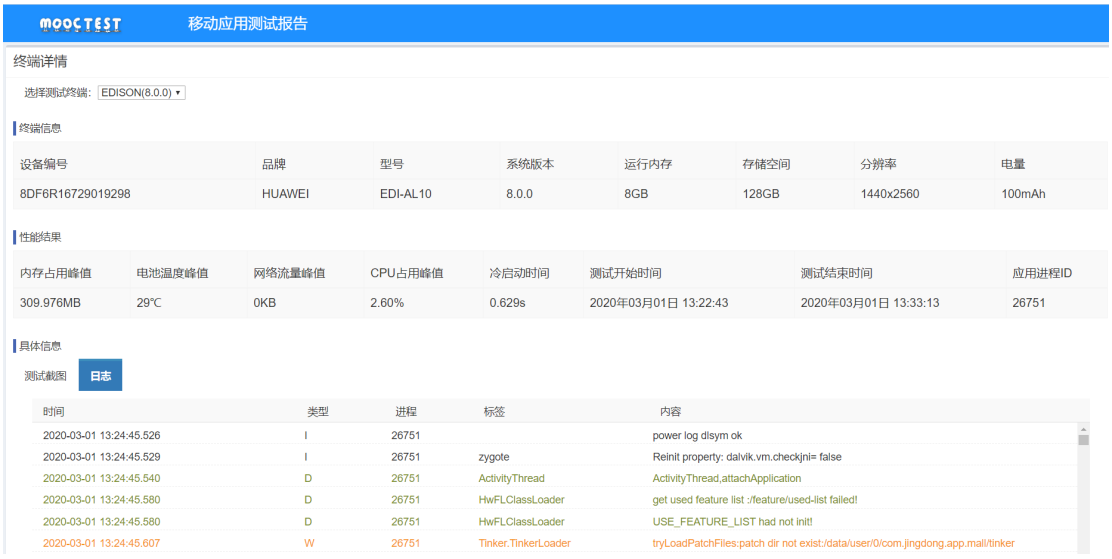


图 4.25: 设备详情部分截图

异常的具体信息包括异常触发时的屏幕截图、异常内容、异常描述以及异常所在窗口。图 4.26展示了某个众包需求中的异常具体信息，结合上文提及的设备信息可得：通过 HUWEI EDL-AL10 设备使用该移动应用时，在进入到 Jshop-MainShopActivity 窗口后，应用触发了一个异常。该异常不属于安卓原生异常，

根据异常内容进行查询后发现“Uncaught ReferenceError”来源于 JS 代码，该异常的触发原因是 H5 页面的整合过程中使用了未定义的变量。用户可以收集遇到的未知异常信息，通过 JSON 文件进行配置，当再次遇到同样的异常时可以进行补充说明，减少查询操作。

图 4.26: 异常信息截图

系统为众包工人提供了相应的引导信息，图 4.27展示了上述异常的某个复现步骤，包括操作描述、窗口跳转以及操作前后的屏幕截图等，用户可依照复现步骤在不同设备不同环境下开展测试和复现异常。

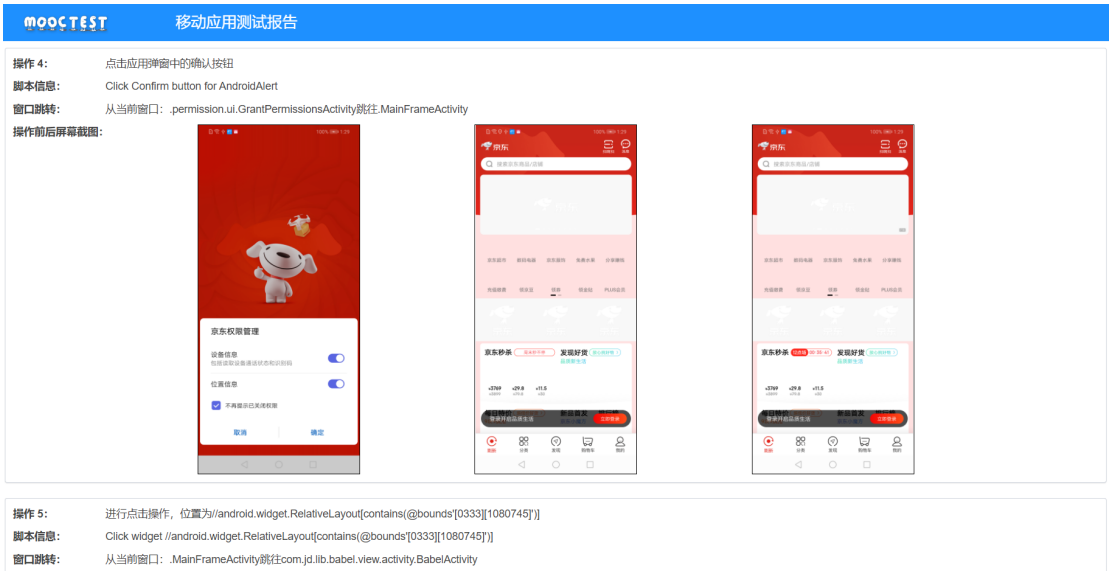


图 4.27: 引导信息部分截图

（5）查看众包测试情况。众测进展页面展示了使用本系统生成的众包需求，发布到 M 公司的众包测试审核平台后，组织开展的众包测试的基本信息，包括场次名称、场次 ID、场次描述、起始结束时间、参加人数以及审核次数。图 4.28展示了使用同一个任务生成的众包需求开展了三次众包测试的概况。

场次名称	场次ID	场次描述	起始时间	结束时间	参加人数	审核次数	查看详情
app众包审核	5e6e1e7cc9e77c000c409fde	jingdong app审核	2020-3-15 20:00:00	2020-3-15 22:00:00	67	1005	查看详情
app众包审核	j16kldnr3dcwyebzmnye8ojg	jingdong app审核	2020-3-16 08:00:00	2020-3-16 10:00:00	82	1148	查看详情
app众包审核	dz9l2nt229fl86lwnrgbsfec	jingdong app审核	2020-3-16 20:00:00	2020-3-16 22:00:00	75	1050	查看详情

图 4.28: 众测进展界面截图

图 4.29展示了某个众包需求的评审情况。该众包需求指定了一个评审问题，题目为“是否 Bug”，在一场众包审核中 67 名众包工人选择了“是”，因此通过众包审核验证了该众包需求中涉及的异常确实属于该应用的一个 Bug。

图 4.29: 众测反馈示例图

4.6　本章小结

本章对系统主要的功能模块进行了详细设计，包括自动化测试数据处理模块、安卓静态分析模块、众包需求推荐模块以及可视化模块，每个模块主要通过类图以及关键代码来阐述核心业务逻辑的实现细节。最后，通过真实案例对系统界面进行截图，包括任务管理、窗口跳转模型、众包需求列表与详情、设备详情与运行信息以及众包测试数据等页面，展示了系统的使用流程和实际效果。

第五章　系统测试与实验分析

本章将基于系统的功能性需求与非功能需求，论述其功能测试与性能测试的详细设计与实施过程，展示测试的具体结果。此外，将通过组织和开展对照实验，对安卓自动化测试驱动的众包需求生成技术的有效性进行验证。

5.1　测试环境

针对系统的功能需求与具体实现，表 5.1列举了系统部署和运行的所需的关键软件工具与硬件设备。在此基础上，对其开展功能测试、性能测试以及对照实验。系统的后端使用 Spring Boot 进行构建，为此需要提供 Java 运行环境；系统的前端项目通过 Webpack 进行打包，作为静态资源托管至 Node.js 服务中。软件和工具使用官方提供的稳定版本，其它选项则参照实际开发环境来进行配置。

表 5.1: 测试环境配置表

设备或软件	配置或版本信息
Angular	4.1
Webpack	4.16.1
Node.js	v10.12.0
Spring Boot	2.0
JDK	1.8.0
MongoDB	V4.2.5
Docker	18.03
浏览器	Chrome60、Firefox71、IE9
硬件设备	操作系统 Linux Redhat 7.6
系统内存	32G
网络带宽	100M

5.2　功能测试

安卓自动化测试驱动的众包需求生成系统的功能测试主要从自动化测试数据处理，安卓静态分析、众包需求推荐模块以及可视化模块四个方面展开。其中自动化测试数据处理模块、安卓静态分析模块以及推荐模块是系统中一个连续的过程，用户无法从外界感知。可视化模块则主要涉及众包需求的展示与用户界面交互。因此，本文中自动化测试数据处理模块，安卓静态分析模块以及众包

需求推荐模块主要通过手工调用相关函数完成；可视化模块的功能测试则可从通过系统的图形界面完成，以下是各个模块的具体测试用例。

（1）自动化测试数据处理模块。表 5.2是自动化测试数据处理流程的测试用例，该测试用例主要用于测试应用的上传、自动化数据分析以及生成引导复现的众包需求的整个流程，主要包括提取自动化测试步骤、构造窗口邻接表以及截图匹配等。实际测试结果符合预期，测试用例全部通过。

表 5.2: 自动化测试数据处理模块测试用例表

测试项	操作/输入	预期结果	测试结果
读取日志文件与堆栈信息	调用 readTestData 函数	读入自动化测试数据，并保存到内存，无格式错误	通过
提取自动化测试操作	调用 handleTestAction 函数	提取自动化测试操作，获取相应测试信息，封装成 TestACtion 类	通过
构造窗口跳转模型	调用 createSkipModel 函数	提取测试操作中的窗口跳转事件，构造出表示无环图的邻接表 ArrayList	通过
匹配屏幕截图	调用 addScreenShots 函数	对于每个测试操作，匹配到最接近的三张截图	通过
生成引导复现的众包需求	调用 createRequirements 函数	生成不重复的具体的众包需求，具有引导信息，并符合 JSON 格式规范	通过

（2）静态分析模块。表 5.3是静态分析模块的测试用例，该测试用例主要用于源码扫描并产出未覆盖窗口以及相关组件类型的众包需求，主要包括反编译 APK 以及匹配源码中的关键信息。实际测试结果符合预期，测试用例全部通过。

表 5.3: 静态分析模块测试用例表

测试项	操作/输入	预期结果	测试结果
反编译 APK	调用 Jadx 图形界面工具, 调用 decompile 函数	生成反编译后的文件夹，文件夹下包含应用源码信息，源码可读	通过
匹配窗口跳转信息	调用 scanActivity 函数	窗口的跳转情况和对应的邻接表 ArrayList	通过
匹配条件分支语句	调用 scanViews 函数	条件分支语句内的组件类型，以及所在窗口	通过
生成探索未覆盖窗口的众包需求	调用 createRequirements 函数	生成去重后的需求，并提供到达前置窗口的测试操作，格式符合 JSON 规范	通过
生成探索分支内组件的众包需求	调用 createRequirements 函数	对于已覆盖窗口，提供组件类型以及所在窗口，如果窗口未覆盖则忽略。	通过

（3）众包需求推荐模块。表 5.4是众包需求推荐模块的测试用例，该用例用

于测试推荐模块中的冷启动以及协同过滤功能。对于众包需求推荐模块，测试用例仅检查其是否有正常的输出，具体的推荐效果可通过问卷调查的形式进行评估。众包需求推荐模块实际测试结果符合预期，测试用例全部通过。

表 5.4: 众包需求推荐模块测试用例表

测试项	操作/输入	预期结果	测试结果
冷启动	调用 RecommendService，设置为冷启动模式。	按照困难程度排序的需求列表，大专院校以上学历用户按降序排列；否则按升序排列。	通过
协同过滤	调用 RecommendService，设置为协同过滤模式。	按偏好程度预测值降序排列的需求列表	通过

（4）可视化模块。可视化模块为用户提供了对系统进行操作的图形界面，包括任务管理、发布众审、查看众包需求概况、查看众包需求详情、查看设备详情和查看众测数据统计。该模块的测试用例与测试结果如表 5.5所示。

表 5.5: 可视化模块测试用例表

测试项	操作/输入	预期结果	测试结果
提交任务	上传待测应用文件，勾选“众包测试协同”选项，提交任务	任务成功提交，等待执行	通过
查看任务	点击“查看任务”选项	展示历史任务及其完成状态	通过
查看众包概况	点击“众包概况”标签页	页面展示窗口跳转图以及众包需求列表	通过
查看需求详情	点击具体的众包需求	页面展示众包需求的详细信息	通过
查看设备详情	点击需求详情中的设备编号	展示设备详细信息、性能信息以及自动化测试过程中的运行信息	通过
发布众审	点击“发布众审”选项	将众包需求发布到 M 公司的众包测试审核平台	通过
查看众测数据统计	点击“查看众测结果”按钮	展示众包测试的基本信息及各项需求的评审情况等	通过

5.3　性能测试

5.3.1　页面性能测试

Web 界面是系统和用户进行交互的部分，前端的性能好坏直接影响到系统用户的使用体验甚至会影响用户的正常操作。因此需要对前端页面，使用 Chrome Devtools 进行性能测试。

图 5.1为用户打开众包需求报告页面时的性能数据，在整个页面加载过程中，JavaScript 的执行时间为 819 毫秒，渲染时间为 92 毫秒，绘制时间为 7ms。页面从打开到加载完毕并进行展示，整个流程耗时 1291 毫秒，在正常的预期之内。页面变化的过程中，每帧绘制得比较流畅，用户不会感觉到明显的卡顿。

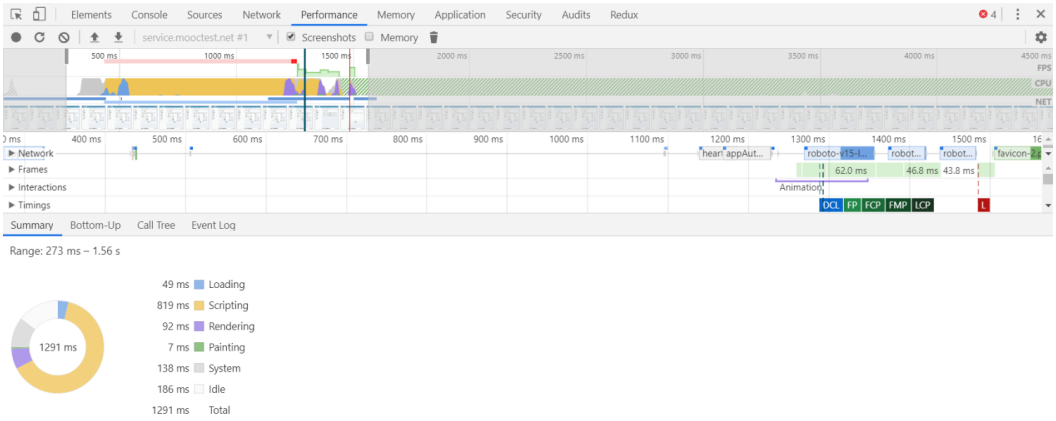


图 5.1: 众包需求报告页面前端性能分析图

表 5.6所示为本系统所涉及的页面加载平均时间，从表格可以看出，本系统的核心功能，包括任务管理、众包概况、需求详情、设备详情、众测数据，其对应的页面加载时间均在 1.5 秒以内。用户的正常操作，并不会引起页面的重新绘制，加载数据时，页面变动的部分还将提供 Loding 动画，减缓用户等待的焦虑。因此，此次前端性能测试的结果为通过。

表 5.6: 页面加载平均时间

页面	测试次数	平均加载时间
任务管理	20	1022ms
众包概况	20	1246ms
需求详情	20	1135ms
设备详情	20	1367ms
众测数据	20	1176ms

5.3.2　接口性能测试

系统的性能的另一个指标是接口响应速度。JMeter 是 Apache 组织旗下的性能测试工具，它为用户提供了图形界面，可在不同的平台运行，是进行性能测试的首选工具。通过 JMeter 可以同时启动多个线程向指定的系统接口发送请求，携带相关参数，模拟多用户和高并发的真实场景。使用 JMeter 对本系统进行性能测试的具体实施步骤如下：

（1）创建线程组。打开 JMeter 的图形化操作界面，在 Test Plan 下新建接口测试的线程组 Thread Group。线程组是进行性能测试的独立单位，它的每一个线程都可以视为使用对系统发起请求的一名真实用户；

（2）填写性能配置。设置 Thread Group 中的虚拟用户数 Number of Threads 为 1000，线程启动时间 Ramp-up period 为 2，循环次数 LoopCount 为 20。该配置模拟 1000 名真实用户，在 2 秒内向系统同时发送请求，为了减少单次测试带来的误差，重复 20 次同样配置的接口测试；

（3）添加测试目标。对创建任务、查看任务、获取众包需求，查看众包概况、需求详情、设备详情以及众测数据相关的 RESTful API 型接口进行配置，创建 HTTP 请求，提供对应的参数，对系统开展性能测试。

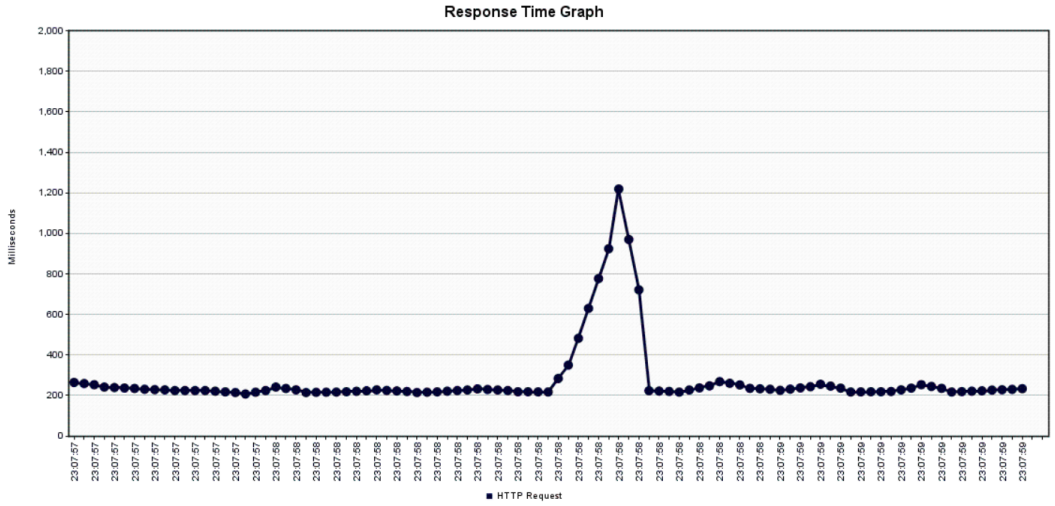


图 5.2: 接口响应时间图

图 5.2展示了模拟 1000 名真实用户同时使用系统的情况。JMeter 在 2 秒内对系统各个接口同时发起了 1000 次请求，图中纵轴表示接口响应时间，横轴表示请求的发起时间，时间间隔为 50 毫秒。由图可得，响应时间曲线大体上是平稳的状态；在请求峰值的最坏情况下，响应时间也在 1.5 秒以内。对数据的进一步分析显示，测试的平均响应时间为 243 毫秒，响应时间中位数为 237 毫秒; 绝大部分请求能在 300 毫秒以内得到响应; 所有请求均得到正常响应，全程无错误发生。因此，本次接口性能测试的结果为通过。

综上所述，系统的前端页面与后端接口均在性能测试中获得了较为优秀的表现。在模拟高并发的场景下，接口响应时间均在合理范围内，系统依然能够正常运行，维持正常的业务流程；系统的前端页面能及时地对响应数据进行渲染，没有明显的卡顿情况发生，为用户提供了良好的使用体验。

5.4　实验分析

5.4.1　实验目标

为了进一步验证安卓自动化测试驱动的众包需求生成系统在实际应用场景下的有效性，本节分别面向众包任务发起者及众包工人开展了实验分析。通过实验，本文将对以下两个问题进行探索：

问题一：本系统能否有效提升众包任务发起者的工作效率?

问题二：本系统能否有效提升众包测试的效率和质量?

5.4.2　实验设置

本实验的实验对象分别为众包任务发起者以及众包工人。其中，众包任务发起者选定为 M 公司的 16 名众审运营人员，他们拥有开展众包测试的相关经验；众包工人则为随机选取的 40 名学生，他们的测试报告具有较高的准确性。此外，本文选取了 10 个提供测试版本的安卓移动应用，分别为 JianShi、TesterHome、Leafpic、YouBo、GuDong、SeeWeather、QQplayer、CloudReader、AnkiDroid、IT-house，这些应用的 APK 文件可在开源平台 Github [1]中进行下载。提供当前国内市场占有率较高的 15 种安卓手机，作为实验的测试设备，如表5.7所示。

表 5.7: 实验设备表

品牌	型号	品牌	型号	品牌	型号
华为	Mate 30	华为	P30	华为	nova5 Pro
华为荣耀	V30	vivo	X30	vivo	NEX 3S
vivo	Z6	OPPO	Remo 3	OPPO	Find X2
小米	10 Pro	小米	9 Pro	小米	MIX 2S
三星	Galaxy20	三星	Note10	一加	7T

为了探索实验提出的问题，本文进行了对照实验，具体的实验设置如下：

（1）众包任务发起者分别通过传统方式和本系统来设计众包需求，并汇总最终的测试数据。在传统方式中，需手动分析自动化测试的原始数据，根据移动应用的使用情况和运行异常，设计相关众包需求；待测试完成后，再手动提取汇总所有测试报告的数据。使用本系统，众包任务发起者需上传待测应用的 APK 文件，选择相关测试设备，并在测试结束后在“众测进展”页面记录最终的数据。

[1]https://github.com/

最后，分别记录两种方式的平均完成时间，并通过问卷调查的形式收集众包任务发起者对不同工作方式的感受与反馈。

（2）众包工人分别参照手动方式设计的众包需求和本系统生成的众包需求完成测试任务，时间限定为 2 小时。最后，记录实验过程中发现的有效 Bug，并检查应用的代码的覆盖情况。

5.4.3　实验结果

为了便于描述，我们把与传统流程相关的人员和数据称为对照组，与本系统相关的人员和数据称为实验组。表 5.8展示了对照组和实验组的实验结果。在面向众包发起者的实验中，对照组设计众包需求的平均时间为 16 分钟，汇总测试结果的平均时间为 12 分钟；实验组完成同样流程的平均时间分别为 3 分钟和 2 分钟。在面向众包工人的实验中，对照组提交了 258 个 Bug，有效 Bug 为 127 个，有效 Bug 占比为 49.2%，平均代码覆盖率为 81.2%；实验组提交了 214 个 Bug，有效 Bug 为 169 个，有效 Bug 占比为 78.9%，平均代码覆盖率为 94.4%。两组均完成了对应用所有窗口的测试，其中对照组所需时间为 112 分钟，实验组所需时间为 84 分钟。

表 5.8: 实验结果统计表

	对照组	实验组
设计需求平均时间	16 分钟	3 分钟
汇总数据平均时间	12 分钟	2 分钟
Bug 总提交数	258	214
有效 Bug 数	127	169
有效比	49.2%	78.9%
平均代码覆盖率	81.2%	94.4%
覆盖全部窗口所需时间	112 分钟	84 分钟

根据实验结果可得，实验组在开展测试任务上的时间花销远远低于对照组。另外，随着众包任务发起者人数的减少，实验组的优势将更加明显。因为实验组通过系统自动生成众包需求及汇总测试数据，替代了传统流程中重复和繁琐的操作，降低了在人力及时间方面的开销。此外，使用本系统还能对测试数据进行去重，进一步提高准确度。在实验结束后，我们对众包任务发起者进行了问卷调查，他们中的大部分人都对本系统表示了肯定的态度。其中，68.75% 的众包任务发起者表示本系统明显提升了他们的工作效率，81.25% 的人员表示希望继续使用本系统来优化众包工作流程。由此可见，本系统为生成众包需求和验收众测数据提供了一站式的服务，有效地提高了众包任务发起者的工作效率。

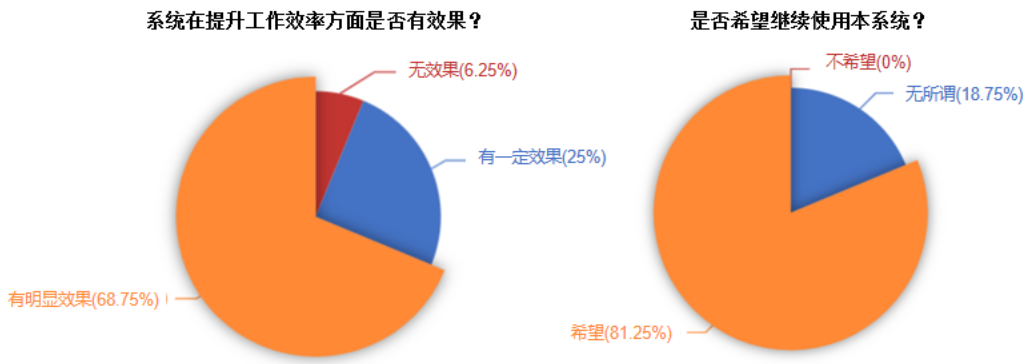


图 5.3: 问卷调查关键评价统计图

在本次实验中，由系统生成的众包需求涵盖了更多的机型设备，也更具有针对性。同时，实验组在复现异常的基础上，运用人的经验与智慧对应用进行了深入探索，因此实验提交的有效 Bug 更多，占比更高。系统通过对应用的静态分析，提供了目标应用的窗口跳转模型，实验组的用户可以直接了解每个应用所涉及的窗口层级和跳转关系，在此基础上可以更快地对所有窗口进行覆盖。实验结束后由众包任务发起者对用户提交的 Bug 进行了核查和分析，对照组和实验组针对每个应用发现的有效 Bug 数如图 5.4所示。在所有的实验应用中，实验组提交的有效 Bug 数量均多于对照组。

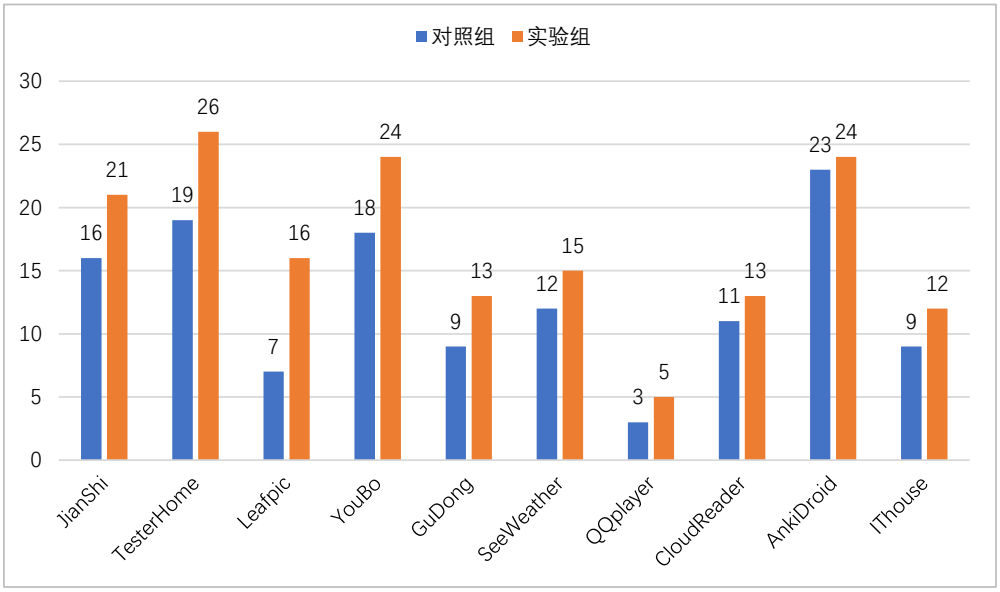


图 5.4: 各应用有效 Bug 数统计图

图 5.5展示了对照组和实验组在各个应用中进行测试的代码覆盖率情况。通

常来说，众包工人不具备阅读源码的条件，也没有专业的测试知识，一些隐藏的窗口和多状态的控件常常没有得到关注和探索，这也是对照组的平均代码覆盖率较低的主要原因。本系统生成的众包需求包括自动化测试中触发但没有得到验证的异常场景、自动化测试未覆盖的窗口以及没有满足某些条件而未触发的事件。通过静态分析技术，实验组用户可以了解到应用所有的窗口以及窗口中涉及条件分支的控件，会针对这些部分进行更细致的探索。

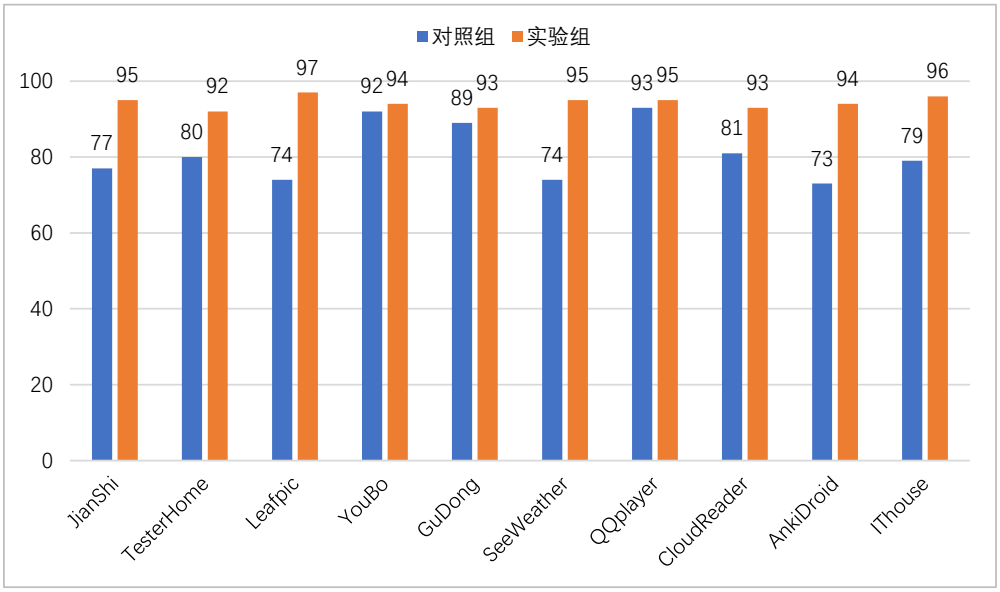


图 5.5: 各应用代码覆盖率统计图

综上所述，众包工人参照本系统生成的众包需求开展测试任务，在窗口覆盖完成时间、有效 Bug 数量以及代码覆盖率方面均优于对照组。由此可得，安卓自动化测试驱动的众包需求生成系统有效提升了众包测试的效率与质量。

5.5　本章小结

本章在真实的环境下，根据功能性需求设计了相关测试用例集，对系统各项功能进行了测试，保证了系统的可用性；对系统的页面和接口进行了性能测试，确保系统能够支持高并发的场景，使用户能够流畅使用，不会感受到明显的卡顿，保障了系统的可靠性。在实验分析中，针对众包任务发起者及众包工人开展了对照实验，并对实验结果进行展示和分析。实验结果表明，安卓自动化测试驱动的众包需求生成系统可以有效地提高安卓移动应用众包测试的效率与质量，并提升众包任务发起者的工作效率，具有积极的使用意义。

第六章　总结与展望

6.1　总结

为了结合自动化测试与众包测试的优点，提高移动应用测试整体的效率与质量，本文设计并实现了安卓自动化测试驱动的众包需求生成系统。本文首先从学术研究和行业应用两个方面，对现有的自动化测试技术、安卓静态分析技术以及众包测试技术进行了调查与研究。接着，介绍了实现系统所涉及的概念与技术：系统前端通过 Angular 进行搭建，利用 ECharts 对数据图表进行绘制，使用 Webpack 将其打包为静态资源文件进行托管；采用 Spring Boot 作为系统后端框架，结合 MongoDB 与阿里云 OSS 存储服务对数据进行持久化管理；系统整体使用 GitLab 进行代码托管与版本控制；通过 Docker 容器引擎将系统及其运行环境打包为一个整体，以微服务的形式进行集成并向外提供服务。

本文对系统的功能性需求与非功能性需求进行了分析，通过“4+1”视图模型全方位展示了系统的整体架构，并划分出具体的功能模块，包括自动化测试数据处理模块、安卓静态分析模块、众包需求推荐模块以及可视化模块。其中，自动化测试数据处理模块负责将自动化测试过程中触发的异常与缺陷转化为众包需求；静态分析模块对移动应用源码进行扫描，提取自动化测试中未覆盖的窗口以及涉及条件判断的控件类型，引导众包工人探索新异常；众包推荐模块根据用户的历史行为数据，对众包需求按照一定的规则进行排序，以期提高众包测试的执行效率；可视化模块为用户提供了图形操作界面，并对各项数据进行了详尽的展示。本文对上述功能模块的技术原理与实施步骤进行了设计，并通过类图和关键代码详细阐述了各个模块的具体实现。最后，通过真实的使用案例和界面截图展示了系统的主要功能与运行情况。

在此基础上，本文对系统进行了功能测试与性能测试，测试结果表明系统具有较优的可靠性，能在高并发的场景下正常提供服务，用户体验良好。同时，通过开展对照实验，证明了安卓自动化测试驱动的众包需求生成技术的有效性。

6.2　展望

目前，安卓自动化测试驱动的众包需求生成系统已正式投入使用。结合真实的使用效果和用户反馈来看，系统基本满足了用户的主要需求，但未来还可以朝着以下方向进行努力：

（1）本文基于安卓移动应用实现了自动化测试驱动的众包需求生成系统，为提高移动应用测试的效率与质量提供了一种可行的思路。然而，因为自动化测试技术、开源条约与平台运营等方面的限制，系统仅对安卓应用提供了支持。为了后续的进一步拓展，系统的实现遵循“高内聚、低耦合”的原则，每个功能模块相互独立，各端之间数据格式统一。当研究和技术条件成熟时，只需提供对应的 iOS 功能模块，设计符合相应规范的数据格式，即可完成 iOS 系统的接入。

（2）本文的众包需求推荐主要是静态方面的，即根据众包工人的历史行为进行推荐，暂时没有实时关注众包任务的执行过程。在未来可以根据众包工人最终实际接受的任务以及他们实际完成任务的效率和质量，动态调整对每个众包工人的任务推荐策略，以期达到降低测试成本的效果。

（3）本文实现了对众测需求的评审情况展示，但因开发时间与技术方案上的限制，暂未对多次众包测试的整体数据进行融合、分析与评价。但是，本系统收集了众包工人对每个众包需求的选取情况以及评审报告，为下一步的数据分析与测试报告融合提供了数据基础。未来系统将借助众包测试报告融合技术，对测试结果进行整合，力求进一步提高用户体验。

参考文献

[1] 中华人民共和国工业和信息化部, 2019 年互联网和相关服务业运行情况, http://www.miit.gov.cn/n1146312/n1146904/n1648355/c7676707/content.html.

[2] Y. M. Baek, D. 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, ASE 2016, Singapore, September 3-7, 2016, 2016, pp. 238–249.

[3] A. M. Memon, I. Banerjee, A. Nagarajan, GUI ripping: Reverse engineering of graphical user interfaces for testing, in: 10th Working Conference on Reverse Engineering, WCRE 2003, Victoria, Canada, November 13-16, 2003, 2003, pp. 260–269.

[4] N. Chen, S. Kim, Puzzle-based automatic testing: bringing humans into the loop by solving puzzles, in: IEEE/ACM International Conference on Automated Software Engineering, ASE'12, Essen, Germany, September 3-7, 2012, 2012, pp. 140–149.

[5] 张志强, 逄居升, 谢晓芹, 周永, 众包质量控制策略及评估算法研究, 计算机学报 (8) (2013) 1636–1649.

[6] H. J., The rise of crowdsourcing, in: Wired Magazine, 2006, pp. 14(6):1–4.

[7] E. Dolstra, R. Vliegendhart, J. A. Pouwelse, Crowdsourcing GUI tests, in: Sixth IEEE International Conference on Software Testing, Verification and Validation, ICST 2013, Luxembourg, Luxembourg, March 18-22, 2013, 2013, pp. 332–341.

[8] C. Hu, I. Neamtiu, Automating GUI testing for android applications, in: Proceedings of the 6th International Workshop on Automation of Software Test, AST 2011, Waikiki, Honolulu, HI, USA, May 23-24, 2011, 2011, pp. 77–83.

[9] M. Usman, M. Z. Iqbal, M. U. Khan, An automated model-based approach for unit-level performance test generation of mobile applications, J. Softw. Evol. Process. 32 (1).

[10] P. Patel, G. Srinivasan, S. Rahaman, I. Neamtiu, On the effectiveness of random testing for android: or how i learned to stop worrying and love the monkey, in: Proceedings of the 13th International Workshop on Automation of Software Test, AST@ICSE 2018, Gothenburg, Sweden, May 28-29, 2018, 2018, pp. 34–37.

[11] 王焱, 张征, 基于持续集成的 android 自动化测试, 计算机系统应用 (5) 263–268.

[12] D. Amalfitano, A. R. Fasolino, P. Tramontana, S. D. Carmine, A. M. Memon, Using GUI ripping for automated testing of android applications, in: IEEE/ACM International Conference on Automated Software Engineering, ASE'12, Essen, Germany, September 3-7, 2012, 2012, pp. 258–261.

[13] 杨博, 唐祝寿, 朱浩谨, 沈备军, 林九川, 基于静态数据流分析的 android 应用权限检测方法, 计算机科学 39 (s3) (2012) 16–18.

[14] S. Yang, H. Zhang, H. Wu, Y. Wang, D. Yan, A. Rountev, Static window transition graphs for android (T), in: 30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015, Lincoln, NE, USA, November 9-13, 2015, 2015, pp. 658–668.

[15] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. L. Traon, D. Octeau, P. D. McDaniel, Flowdroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps, in: ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '14, Edinburgh, United Kingdom - June 09 - 11, 2014, 2014, pp. 259–269.

[16] 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, SOAP@PLDI 2016, Santa Barbara, CA, USA, June 14, 2016, 2016, pp. 18–23.

[17] K. Mao, L. Capra, M. Harman, Y. Jia, A survey of the use of crowdsourcing in software engineering, J. Syst. Softw. 126 (2017) 57–84.

[18] 章晓芳, 冯洋, 刘迪, 陈振宇, 徐宝文, 众包软件测试技术研究进展, 软件学报 29 (1) (2018) 69–88.

[19] 冯剑红, 李国良, 冯建华, 众包技术研究综述, 计算机学报 (9) (2015) 1713–1726.

[20] K. Chen, C. Wu, Y. Chang, C. Lei, A crowdsourceable qoe evaluation framework for multimedia content, in: Proceedings of the 17th International Conference on Multimedia 2009, Vancouver, British Columbia, Canada, October 19-24, 2009, 2009, pp. 491–500.

[21] D. Liu, R. G. Bias, M. Lease, R. Kuipers, Crowdsourcing for usability testing, in: Information, Interaction, Innovation: Celebrating the Past, Constructing the Present and Creating the Future - Proceedings of the 75th ASIS&T Annual Meeting, ASIST 2012, Baltimore, MD, USA, October 26-30, 2012, 2012, pp. 1–10.

[22] R. Vliegendhart, E. Dolstra, J. A. Pouwelse, Crowdsourced user interface testing for multimedia applications, in: Proceedings of the ACM multimedia 2012 workshop on Crowdsourcing for multimedia, CrowdMM@ACM Multimedia 2012, Nara, Japan, October 29, 2012, 2012, pp. 21–22.

[23] A. Kittur, E. H. Chi, B. Suh, Crowdsourcing user studies with mechanical turk, in: Proceedings of the 2008 Conference on Human Factors in Computing Systems, CHI 2008, 2008, Florence, Italy, April 5-10, 2008, 2008, pp. 453–456.

[24] N.Murray, F.Coury, A.Lerner, C. Taborda, Ng-book:The Complete Guide to Angular, CreateSpace Independent Publishing Platform, 2018.

[25] D. Li, H. Mei, Y. Shen, S. Su, W. Zhang, J. Wang, M. Zu, W. Chen, Echarts: A declarative framework for rapid construction of web-based visualization, Vis. Informatics 2 (2) (2018) 136–146.

[26] 冀潇, 李杨, 采用 echarts 可视化技术实现的数据体系监控系统, 计算机系统应用 (6) 72–76.

[27] 孙昌爱, 金茂忠, 刘超, 软件体系结构研究综述, 软件学报 13 (7) (2002) 1228–1237.

[28] 张广胜, 蒋昌俊, 汤宪飞, 徐岩, 面向服务的企业应用集成系统描述与验证, 软件学报 (12) 3015–3030.

[29] H. Kang, M. Le, S. Tao, Container and microservice driven design for cloud infrastructure devops, in: 2016 IEEE International Conference on Cloud Engineering, IC2E 2016, Berlin, Germany, April 4-8, 2016, 2016, pp. 202–211.

[30] S. Wang, L. Zhu, M. Cheng, Docker-based web server instructional system, in: 18th IEEE/ACIS International Conference on Computer and Information Science, ICIS 2019, Beijing, China, June 17-19, 2019, 2019, pp. 285–289.

[31] C. Boettiger, An introduction to docker for reproducible research, Operating Systems Review 49 (1) (2015) 71–79.

[32] 胡启敏, 薛锦云, 钟林辉, 基于 spring 框架的轻量级 j2ee 架构与应用, 计算机工程与应用 044 (5) 115–118.

[33] F. Gutierrez, Spring Boot Messaging, Messaging APIs for Enterprise and Integration Solutions, Springer, 2017.

[34] M. R. Naveira, R. Sánchez-de-Madariaga, J. B. Castro, L. C. García, G. V. González, S. Pérez, M. P. Carrasco, F. Martín-Sánchez, A. M. Carrero, An archetype query language interpreter into mongodb: Managing nosql standardized electronic health record extracts systems, J. Biomed. Informatics 101 (2020) 103339.

[35] J. C. C. Ríos, K. Kopec-Harding, S. Eraslan, C. Page, R. Haines, C. Jay, S. M. Embury, A methodology for using gitlab for software engineering learning analytics, in: Proceedings of the 12th International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE@ICSE 2019, Montréal, QC, Canada, 27 May 2019, 2019, pp. 3–6.

[36] 王非, 蔡勇, 贺志军, Restful web services 在信息系统中的应用, 计算机系统应用 (2) 223–227.

[37] K. Guntupally, R. Devarakonda, K. Kehoe, Spring boot based REST API to improve data quality report generation for big scientific data: ARM data center example, in: IEEE International Conference on Big Data, Big Data 2018, Seattle, WA, USA, December 10-13, 2018, 2018, pp. 5328–5329.

[38] A. Martin-Lopez, S. Segura, A. Ruiz-Cortés, A catalogue of inter-parameter dependencies in restful web apis, in: Service-Oriented Computing - 17th Interna-

tional Conference, ICSOC 2019, Toulouse, France, October 28-31, 2019, Proceedings, 2019, pp. 399–414.

[39] A. Arcuri, Restful API automated test case generation with evomaster, ACM Trans. Softw. Eng. Methodol. 28 (1) (2019) 3:1–3:37.

[40] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions, IEEE Trans. Knowl. Data Eng. 17 (6) (2005) 734–749.

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

[42] B. M. Sarwar, G. Karypis, J. A. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001, 2001, pp. 285–295.

[43] A. Laishram, V. Padmanabhan, Discovery of user-item subgroups via genetic algorithm for effective prediction of ratings in collaborative filtering, Appl. Intell. 49 (11) (2019) 3990–4006.

[44] B. McFee, L. Barrington, G. R. G. Lanckriet, Learning similarity from collaborative filters, in: Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010, Utrecht, Netherlands, August 9-13, 2010, 2010, pp. 345–350.

[45] G. Karypis, Evaluation of item-based top-n recommendation algorithms, in: Proceedings of the 2001 ACM CIKM International Conference on Information and Knowledge Management, Atlanta, Georgia, USA, November 5-10, 2001, 2001, pp. 247–254.

[46] B. Lika, K. Kolomvatsos, S. Hadjiefthymiades, Facing the cold start problem in recommender systems, Expert Syst. Appl. 41 (4) (2014) 2065–2073.

[47] P. Cremonesi, Y. Koren, R. Turrin, Performance of recommender algorithms on top-n recommendation tasks, in: Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010, 2010, pp. 39–46.

简历与科研成果

基本情况 韦志宾，男，汉族，1996 年 2 月出生，广东省茂名市人。

教育背景

2018.9 ~ 2020.6 南京大学软件学院 硕士

2014.9 ~ 2018.6 电子科技大学软件学院 本科

读研期间的成果

1. Haoyu Li, Chunrong Fang, **Zhibin Wei**, Zhenyu Chen: CoCoTest: collaborative crowdsourced testing for Android applications. ISSTA 2019: 390-393
2. 陈振宇，**韦志宾**，房春荣，李玉莹，“一种基于控件融合的群体智能测试方法”，申请号：2019107240511，已受理。
3. 房春荣，徐悠然，田元汉，**韦志宾**，陈振宇，李玉莹，“一种结合群体智能与机器智能增强移动应用测试的方法”，申请号：201910952056X，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

致　谢

转眼间，美好的研究生生活即将画上句号。在这两年里，我努力学习软件工程专业知识，积极参与相关项目实践，并得到了老师、同学和家人的关怀和帮助。在毕业论文完成之际，我要向他们致以最诚挚的感谢。

首先，我要感谢我的两位导师，陈振宇教授和房春荣老师。我的成长离不开他们的辛勤指导。陈老师学识渊博、经验丰富，在论文选题期间给我提供了很大的帮助。在论文写作过程中更是悉心指导，多次对论文提出了宝贵的修改建议。房老师也对我的提出了许多实质性的建议，让我能够及时发现并改正论文中存在的问题。感谢他们的悉心指导，我才能够顺利完成项目的开发与论文的编写。

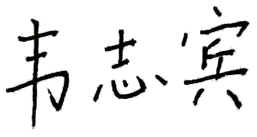
其次，我要感谢 iSE 实验室的同学们。两年的时光里我们一同学习和成长。我们经常在学习方面互相帮助，在实际项目方面共同探讨，相互促进。在 iSE 实验室里，我不仅学到了许多项目开发技术和经验，也学到了同学们身上那些努力、乐观、积极的生活态度。

最后，感谢我的父母。感谢他们坚定不移的在人生道路上支持着我，他们的关怀、鼓励与陪伴，是我学习和生活的重要动力来源。他们总是在精神上不断支持我，尊重我做的每一个决定，父母永远是我最坚强的后盾，感谢他们。

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： 2020 年 5 月 28 日

《学位论文出版授权书》

本人完全同意《中国优秀博硕士学位论文全文数据库出版章程》（以下简称“章程”），愿意将本人的学位论文提交“中国学术期刊（光盘版）电子杂志社”在《中国博士学位论文全文数据库》、《中国优秀硕士学位论文全文数据库》中全文发表。《中国博士学位论文全文数据库》、《中国优秀硕士学位论文全文数据库》可以以电子、网络及其他数字媒体形式公开出版，并同意编入《中国知识资源总库》，在《中国博硕士学位论文评价数据库》中使用和在互联网上传播，同意按“章程”规定享受相关权益。

作者签名：韦志宾

2020 年 5 月 28 日

论文题名	安卓自动化测试驱动的众包需求生成系统的设计与实现				
研究生学号	MF1832173	所在院系	软件学院	学位年度	2020
论文级别	☐学术学位硕士 ☑专业学位硕士 ☐学术学位博士 ☐专业学位博士 (请在方框内画钩)				
作者 Email	wzb19960208@outlook.com				
导师姓名	陈振宇，房春荣				

论文涉密情况：

☑ 不保密

☐ 保密，保密期（_____年_____月_____日 至 _____年_____月_____日）