南京大学

研 究 生 毕 业 论 文

（申请工程硕士学位）

论 文 题 目	面向移动应用众测的人机协同引导技术
作 者 姓 名	朱齐
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	刘嘉 副教授，房春荣 助理研究员

2021 年 05 月 20 日

学　　号：**MF1932272**

论文答辩日期：**2021 年 05 月 20 日**

指 导 教 师：　　**（签字）**

Human-machine Collaborative Guidance Technology for Crowdsourcing Testing of Mobile Applications

By

Qi Zhu

Supervised by

Professor **Jia Liu**

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 2021

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：面向移动应用众测的人机协同引导技术

工程硕士（软件工程领域）专业 2019 级硕士生姓名：朱齐

指导教师（姓名、职称）：刘嘉 副教授，房春荣 助理研究员

摘 要

随着手机操作系统的发展，Android 操作系统在手机市场占据 71.93% 的份额，基于 Android 系统开发的应用也逐年猛增。Android 应用数量庞大、迭代快速的特点也促进了 Android 应用测试的发展，其中主流的测试方法有移动应用自动化测试和众包测试。移动应用自动化测试能够以低成本快速高效的对应用进行测试，但由于 Android 系统碎片化问题严重，使得其测试结果达不到理想要求。众包测试是借助众包工人进行测试，能够使应用在不同机型不同系统版本中进行测试，但由于众包工人专业能力不一，使得其测试结果良莠不齐。

在此背景之下，本文提出了面向移动应用众测的人机协同引导技术，通过兼容移动应用自动化测试和众包测试的优点，以自动化测试结果来引导众包工人进行测试，提高其测试能力，以众包测试对自动化测试未覆盖页面进行再测试，提高窗口覆盖率，最终实现以人机协同的方式更高效的对应用进行测试。本系统包括自动化测试数据处理模块、Android 静态分析模块、Android 引导模块和服务端模块。自动化测试数据处理模块是通过对移动应用自动化测试结果进行解析，提取应用在不同机型上测试的非重复性异常和异常路径信息，作为异常复现的依据。Android 静态分析模块是通过使用静态分析工具对移动应用进行静态分析，获得应用所有窗口跳转路径，并通过与自动化测试结果进行比对，找出自动化测试中未覆盖的窗口，作为未覆盖界面引导的依据。Android 引导模块是以插件形式嵌入到待测应用中，众包工人在测试过程中通过点击插件按钮来请求服务。服务端模块使用 Spring Boot 框架进行开发，通过结合 MyBatis 实现了用户登录、任务推荐、路径引导和测试结果上报等功能的开发，使系统具有更高的可拓展性。

系统在慕测平台进行了应用测试。通过选取 GitHub 中 10 款开源移动应用，设置 4 组对照测试，在 10 台 Android 设备上进行了 20 分钟的系统实验。根据对用户提交的结果进行分析，表明该系统在众包工人测试异常的效率以及发现异常的数量等方面都有显著提高。在窗口覆盖率方面，本系统实现了窗口路径全覆盖，有效弥补了自动化测试的不足。

关键词：自动化测试，众包测试，静态分析，人机协同

南京大学研究生毕业论文英文摘要首页用纸

THESIS: Human-machine Collaborative Guidance Technology for Crowdsourcing Testing of Mobile Applications

SPECIALIZATION: Software Engineering

POSTGRADUATE: Qi Zhu

MENTOR: Professor **Jia Liu**

Research Assistant **Chunrong Fang**

Abstract

With the development of mobile phone operating systems, the Android operating system occupies 71.93% of the mobile phone market and the applications developed on the basis of the Android system have alse soared year by year.The large number of Android mobile applications and rapid iteration of applications have promoted the development of Android mobile application testing. The mainstream testing methods include mobile application automated testing and crowdsourcing testing. mobile application automated testing can test applications quickly and efficiently at low cost, but due to the serious fragmentation of the Android system, its test results cannot meet the ideal requirements. Crowdsourced testing is conducted with the help of crowdsourced workers, which enables applications to be tested in different models and different system versions. However, due to the different professional abilities of crowdsourced workers, the test results are quite different.

In this context, this thesis proposes a human-machine collaborative guidance technology for mobile application crowdsourcing testing, which is compatible with the advantages of automated testing and crowdsourcing testing.It uses automated test results to guide crowdsourcing workers to conduct tests and improve their testing capabilities. At the same time, it can test uncovered windows by leveraging the power of crowdsourced workers to improve window coverage. This system includes automated test data processing module, Android static analysis module, Android boot module and server module.Firstly, the automated test data processing module can analyze the automated test results of mobile applications, and extract the information of non-repetitive

exceptions and path tested on different models as the basis for the recurrence of the exceptions. Secondly,The Android static analysis module conducts static analysis on mobile applications by using static analysis tools to obtain the jump paths of all Windows of applications. By comparing it with the automated test results to find out the windows that are not covered in the automated test, and guide crowdsourcing workers to test them. The Android boot module is embedded in the application as a plug-in. During the test, the crowdsourced workers select the event to be tested by clicking the exception recommend button or uncovered recommend button, and use the path guidance button to quickly locate the location of exception or uncovered windows.The crowdsourcing workers click the buttons to request services.The server uses the Spring Boot framework and Mybatis to realize the functions of user login, task recommendations, path guidance, and test results reporting, which makes the system more scalable.

The system has been tested on the Mooctest platform.This system selected 10 open source mobile applications in GitHub, and conducted a 20-minute integration acceptance test on 10 Android devices by setting up 4 sets of experiments.Analysis of the results submitted by users shows that the system has significantly improved the efficiency of workers' testing of exceptions and the number of exceptions found. In terms of window coverage, the system achieves full coverage of window paths, effectively making up for the lack of automated testing.

Keywords: Automated testing, crowdsourcing testing, static analysis, human-machine collaboration

目录

表 目 录

图 目 录

第一章　引言

1.1　选题的背景和意义

根据 statcounter 的统计，截止至 2021 年 2 月，Android 操作系统占据全球手机操作系统总量的 71.93%[1], 大幅度居于领先地位。由于 Android 系统庞大的市场份额以及人们日常生活和移动手机紧密联系，使得 Android APP 的数量也随之猛增。根据 IT 之家做出的统计，仅在中国地区 Android APP 的数量就高达 449 万个，且其数量仍在高速增长之中，其种类更是覆盖了支付、交通、办公、娱乐的各个方面，从不同的层面影响着我们的生活，带给人们更加新颖更加便捷的生活方式。

数量庞大，类型复杂的 Android APP 也促进了 Android APP 测试的发展，众多软件开发企业开始寻求测试成本低、测试周期短、测试回报高的测试方式，众包测试和移动应用自动化测试逐渐成为主流的测试方式。

移动应用自动化测试是指用已经完整定义的软件测试方法和工具, 来测试移动设备上的本地应用和 Web 应用, 以确保应用的功能、性能以及服务的质量 [2]。移动应用测试具有很多优点。首先，它可以在短时间内进行大量的测试，缩短测试周期。同时它能够完成手工测试难以完成的测试，比如模拟海量用户场景的压测。由于其测试是自动进行的，每次测试执行的内容和结果具有一致性，保障了测试的稳定性 [3]。但是，由于 Android 系统版本繁多，对各类软件的兼容性不一，使得同一款应用在不同版本的手机中进行自动化测试结果并不完全一致。另外，大量手机制造商在开源 Android 操作系统基础之上，进行闭源再开发，使得碎片化问题更加严重，移动应用自动化测试结果更加不稳定。

众包测试是通过利用众包和云平台的优势，将以前由员工执行的测试任务，以自由自愿的形式分派给非定向人员进行测试 [4]。这种由人工进行测试的方式在一定程度上解决了移动应用自动化测试中重复性机械测试和碎片化等问题。此外，众包测试可以通过雇佣大量众包工人来缩短测试周期，同时利用众包工人使用不同品牌不同系统版本的手机收集更详细、更多样的测试结果。但是，由于众包工人多是缺乏专业测试经验的人，测试能力水平良莠不齐，不同人之间的测试结果差别较大。另外，众包工人是根据找到的错误数量来获取报酬，使得他们更热衷于容易被发现但影响较小的错误，忽略复杂操作可能带来的错误，测试结果不全面。

在此背景下，本文提出了一种面向移动应用众测的人机协同引导技术，兼容移动应用自动化测试和众包测试的优点，通过移动应用自动化测试弥补众包工人可重复性测试能力差、测试结果不全面的缺点，通过众包测试弥补移动应用自动化测试界面覆盖率低、灵活性差的缺点，以人机协同的方式实现对 Android APP 更加高效的测试。借助慕测自动化测试平台，该系统能根据平台自动化测试结果抽取不同手机中出现的非重复性异常信息，并利用工具引导众包工人进行异常复现，避免众包工人在已测试界面进行重复测试。对于自动化测试结果未覆盖的界面，系统通过对 Android APP 进行静态分析，对比自动化测试结果找出未覆盖界面，同样通过工具引导众包工人进入未覆盖页面进行测试，弥补了自动化测试的不足。

1.2　国内外研究现状及分析

1.2.1　Android 自动化测试

随着 Android 智能手机的发展，其市场占有率变得越来越高，Android 移动应用的数量也出现几何式增长的局面，而 Android 移动应用更新迭代速度快，对移动应用测试提出越来越高的要求。在此背景之下，各种针对 Android 操作系统的自动化测试工具逐渐发展起来。Android 自动化测试是基于图形用户界面，通过指令来模拟用户操作和系统事件，通过触发事件来检测程序执行过程中是否存在异常 [5]。本文接下来将对几个常用 Android 自动化测试工具进行介绍。

Monkey 作为 Android 原生的命令行工具，可以运行在模拟器或者设备中。它通过 ADB 实现对设备的连接，通过向系统发送伪随机事件，包括触摸事件、滑动事件和导航事件等多种事件来实现对运行的程序的测试 [6]。但 Monkey 测试只能用于验证程序的稳定性和健壮性，伪随机事件的测试方法不允许用户对事件进行自定义，因此只适用于进行黑盒测试而不太适合进行自动化测试 [7]。与 Monkey 基于伪随机事件不同，MonkeyRunner[8] 工具提供了应用程序接口，用户可以使用 Python 语言来编写测试脚本，通过此接口来向设备发送指定命令，截取用户界面图片并存储。同时，MonkeyRunner 还提供录制和回放功能，帮助我们定位并复现异常。

Instrumentation 是 Google 提供的自动化测试工具类，与 Junit 只能进行单元测试不同，它允许用户进行复杂行为测试。Instrumentation 通过在同一个进程中运行主程序与测试程序，使得它可以模拟屏幕点击、滚动等事件，并且能在程序运行期间实现对主程序的实时监控。Robotium 是针对 Android 移动应用进行黑盒测试的框架 [9]，是在 Instrumentation 框架基础之上发展起来的。它通过利用 Java 反射原理，获取应用界面元素，从而实现对界面元素的操作。Instrumentation

和 Robitum 的缺点是都不支持跨 Android APP，且 Robitum 并不能支持所有的视图和对象。

UIAutomator 作为 Android 提供的自动化测试框架 [10]，对所有 Android 事件操作都提供支持，且不需要了解代码实现细节便可实现对界面元素的测试，并且能跨 Android APP 进行测试的特点进一步扩大了其适应范围。但是 UIAutomator 对于 Android 操作系统版本在 4.1 之下的手机以及混合应用、Web 应用不提供支持。Appium 框架是一个开源的自动化测试框架 [11]，与 UIAutomator 不同之处在于，它适用于原生应用、混合应用以及 Web 应用的测试 [12]。除此之外，它对 Android、IOS、FirefoxOS 以及其他可拓展平台都提供支持，具有跨设备、跨语言和跨 APP 等多个优点。

综上所述，自动化测试可以对移动应用进行压力测试，获取测试异常信息和截图并进行异常复现。但是，自动化测试无法解决碎片化带来的问题，因此需要通过人工测试的方式来对自动化测试未能覆盖的界面进行测试。

1.2.2　众包测试

Howe 等人在 2006 年提出众包的概念，他们认为众包是指一个公司或机构把过去应该由内部人员完成的工作任务以自由自愿的形式外包给非特定的大众网络的做法 [13]。Mao 等人在此基础之上，对众包在软件工程中的应用进行了全面总结和概述，他指出利用开放平台来招募全球在线网络工程师从事各种类型软件工程任务，可以通过增加并行性的方式缩短产品上线时间 [14]。众包测试系统组成包括三类主体 [15]：

（1）发包方，任务的发起者，众包测试中从大众输入中获得利益的组织。

（2）接包方，任务的执行者，由个体组成并通过群体方式解决问题，是众包测试中任务的主要负责者。

（3）众包平台，作为中间媒介，将发包方与接包方进行联系。

众包测试流程包括发包方将众包测试任务提交至平台、接包方从平台中获取测试任务，并提供解决方案、众包平台从提交报告中整合提取需要的报告结果。在整个测试过程中，众包平台扮演着重要的角色，平台奖励机制的优劣决定平台能够吸引众包工人数量的多少，平台成果评估能力的好坏决定众包测试的完成度的高低 [16]。如今，越来越多的公司开始寻求第三方测试平台进行软件测试，这极大促进了国内外众包测试平台的发展。国外比较著名的众包测试平台有 Applause、GfK 等，国内的众包测试平台则有 Testin、MOTODO、MOOCTest 和百度众包测试等平台。

众包测试不是对传统测试的取代，而是对现有测试的一种补充，通过不同

地点、不同背景的用户对移动应用进行探索式测试，在获得缺陷报告的同时，能够获得更多元化的用户反馈 [17]。另外，移动应用测试需在不同版本、不同品牌的手机中得到充分测试，若不依赖众包则会花费巨大的人力财力资源，因此众包测试也满足测试成本低的要求。但是，对于应用中存在的简单任务，会出现重复测试的情况，对于应用中的复杂任务，不同测试能力的众包工人测试结果会相差较大。因此，本文会通过自动化测试来减少任务的重复测试，通过引导来增强众包工人的测试能力。

1.2.3　人机协同

人机协同是指将人和计算机置于同一个系统之中，其中计算机通过其强大的算力对大量的数据进行数据计算或者数据推理，人的工作是处理计算机无法通过其算力处理的部分，如决策制定、结果评估等。通过人和机器相互之间的协作，可以高效的处理复杂的问题。

张守刚, 刘海波在人工智能的认识论问题中已经提出，在通过机器对问题进行求解的过程中，不能缺少人的参与，机器求解问题的本质实际上是人和机器求解问题的统一 [18]。J Krueger 将人机协同定义为人与机器协调一致的交互目标，即通过同步协作来实现交互的有效性 [19]。他认为人的任务通常是处理复杂的需要不断适应过程变化的问题，而机器的任务通常是处理重复性且具有较低复杂性的问题。刘杰、韩烨等人提出了一种基于人机协同的软件漏洞模糊测试方法 [20]，通过将人的认知能力与软件服务进行融合，以人机协同的方式实现软件服务和认知服务的灵活切换，使软件漏洞模糊测试方法的测试能力进一步得到了提高。

本文同样采用人机协同的方式对 Android 应用进行测试。通过 Android 自动化测试，对 Android 应用中存在的异常进行捕获，之后以路径引导的方式帮助众包工人快速定位异常并进行异常复现。对于 Android 自动化测试未覆盖界面以及 Android 应用中出现的布局异常等难以通过自动化测试挖掘的问题，通过发挥众包工人的主观能动性，能有效对上述问题进行测试，弥补 Android 自动化测试的不足。

1.3　本文的工作

本文通过对 Android 自动化测试和众包测试进行调研分析，发现 Android 自动化测试能够通过运行脚本命令对 Android 应用进行遍历测试，在触发异常时能够捕获异常信息，同时能够以截图的形式记录异常发生的位置，便于异常复现。但由于碎片化问题，使得 Android 自动化测试的测试窗口覆盖率在不同品牌不同版本的手机中分化严重，且对于布局异常等问题无法进行捕获。众包测试

通过利用人的主观能动性可以解决自动化测试过程中存在的布局异常以及测试窗口覆盖率低等问题，且通过在多种类型的手机进行测试能对应用的兼容性进行更加充分的测试。但是众包工人测试能力不同使得其很难达到满意的测试效果。在此基础之上，本文提出了一种面向移动应用众测的人机协同引导技术，它结合 Android 自动化测试、众包测试、静态分析等多种方式，通过使用 Android 自动化测试和静态分析来实现对异常窗口和未覆盖窗口获取，并对众包工人进行引导，使得能更加高效进行异常复现。同时，利用众包工人测试自动化测试中未覆盖窗口中存在的异常以及布局异常。

本文主要从六个功能模块来进行系统开发，包括自动化测试结果处理模块、Android 静态分析及处理模块、Android 引导模块、推荐模块、路径引导模块和测试结果上报模块。自动化测试结果处理模块是通过对自动化测试结果进行分析，提取移动应用在所有测试机型中的非重复异常信息用于异常复现。同时，对于自动化测试结果中产生的截图信息，该模块会根据测试结果中的点位描述信息对截图进行定点标记，以便更加高效的对众包工人进行引导。Android 静态分析模块是通过静态分析找出移动应用的所有窗口信息以及窗口跳转信息，通过构建窗口转换图并与自动化测试结果进行对比，找出自动化测试未覆盖页面，用于提高测试窗口覆盖率。对于未覆盖窗口，需要通过人工的方式对未覆盖页面进行截图并标记，便于快速引导众包工人到未覆盖窗口进行测试。Android 引导模块包括用户登录、未覆盖界面推荐、异常界面推荐、路径引导和测试结果上报等按钮，它以悬浮按钮的形式嵌入到待测应用中，帮助众包工人进行测试，并监听 Crash 异常。推荐模块包括异常界面推荐和未覆盖界面推荐，帮助众包工人明确异常信息和未覆盖窗口信息，减少其无目的测试行为。路径引导模块是根据众包工人的推荐任务与众包工人当前所处位置，对其下一步操作以信息描述和图片的形式对其进行实时引导，提高其测试效率。测试结果上报模块用于提交测试结果，测试结果包括异常复现测试结果和新异常提交测试结果，通过众包工人对异常信息进行描述，并对异常界面进行截图，将测试结果发送至后台，进行异常信息统计。

1.4　本文的组织结构

本文共分为六个章节，组织结构如下。

第一章为引言，介绍了 Android APP 测试的必要性，并详细说明两种主流测试方法，自动化测试和众包测试的国内外研究现状和其优缺点，并根据当前存在的问题提出了面向移动应用众测的人机协同引导技术。

第二章为技术综述，介绍了本系统中使用的各种开发工具和相关技术，包括 Android 端开发技术、Spring Boot 开发框架、MariaDB 数据库、静态分析工具

Gator、冷启动方式和协同过滤等相关知识。

第三章为系统需求分析与概要设计，首先从总体的角度对系统进行概述，并通过对其进行涉众分析找出系统的功能需求和非功能需求。之后通过系统用例图和表格，借助系统架构、逻辑视图、开发视图、物理视图和进程视图等多个角度，对系统进行深度剖析。最后对系统数据库相关字段进行详细说明。

第四章为系统详细设计与实现，针对各个模块从模块的详细设计与实现、模块类图、模块顺序图和关键代码等方面，详细介绍该模块的实现过程。

第五章为系统测试与结果分析，首先对系统各模块进行功能测试与非功能测试，确保各模块的正确性和可靠性，接着设计四组实验来对系统进行验证。最后展示测试结果并对结果进行分析，验证了系统的可用性和有效性。

第六章为对系统的总结和展望，首先是对系统开发的背景和系统开发的目标进行概述，然后对系统模块的实现进行了简要描述，最后对系统中存在的不足进行分析，并提出了对应的改进方法。

第二章　技术综述

2.1　Android 端相关技术

2.1.1　OkHttp

OkHttp 是移动支付公司 Square 贡献的一个处理网络请求的开源框架，用来替换 HttpUrlConnection 和 Apache HttpClient，是当前 Android 端最火热的轻量级框架 [21]。OkHttp 相比其他的网络框架有很多优势。首先，OkHttp 通过共享 Socket 连接，减少对服务器的请求次数，同时通过连接池策略，大幅度减少请求延迟。对于重复的网络请求，OkHttp 通过使用缓存来进行响应，大幅度减少对流量的消耗，提高了响应速度。

本系统用于提供给众包工人在众包测试过程中使用，众包测试的网络请求具有访问量大且重复性高等特点，采用此框架可以解决上述问题。

2.1.2　CrashHandle

众包测试的过程中，不可避免的会出现 Android APP 系统崩溃的现象。常见的 Android APP 系统崩溃主要有两类，Java Exception 异常和 Native Signal 异常，当出现上述异常时，系统均通过弹出“程序暂无响应”对话框的形式来告知用户，之后便自动退出程序。另外, 系统长时间未响应，屏幕适配失败，Android 版本适配失败等情况都会使众包工人失去对 Android APP 的操纵权而出现系统强制性退出的现象 [22]。上述信息是众包工人进行众包测试过程中的关键信息，但是众包工人不能主动进行上报，因此我们需要通过其他方式来进行处理，这里我们采用定义 CrashHandle 类来进行统一处理。CrashHandle 实现了 Thread 类中的 UncaughtExceptionHandler 接口，其功能是用于捕获 Android APP 崩溃异常。当处在运行中的程序发生无法捕获的异常时，会通过该类进行程序接管，并进行手机设备数据收集，保存异常 log 日志，回传错误日志等操作。

2.1.3　FloatingActionButton

FloatingActionButton 是 Support Design Library 库中引入的一个新的控件 [23]，通过一个圆形的悬浮在 UI 之上的图标，可以灵活方便地指定其实现的动作行为，其主要用于附加功能的开发。

该项目通过继承 FloatingActionButton 来实现按键隐藏效果。通过将实现具体功能的按键进行隐藏，可以减小植入在 Android APP 中的插件给众包工人视

觉及操作上带来的影响，可以使众包工人更专注于众包测试，同时又能方便的使用该插件功能。

2.2　服务端相关技术

2.2.1　微服务

传统的 IT 软件大都是单体结构，是各种独立系统的堆砌，在软件项目规模比较小时，单体结构软件工作情况良好 [24]。随着系统功能逐渐增加，系统规模不断扩大，单体结构拓展性差，维护成本高和可靠性不强等缺陷也随之而来。为了解决其复杂性不断增加，技术债务不断升高，部署速度不断变慢等问题，微服务的思想开始流行。

微服务架构就是将单应用程序作为一套小型服务开发的方法，每个应用程序都运行在自己的进程之中，并通过轻量级设备与 HTTP 型资源的 API 进行通信 [25]。在微服务系统中，微服务以组配的形式来完成各种任务，组配方式的不同，所能解决的任务也就不同，体现了微服务单一性强，灵活性高的特点。该系统同样采用微服务的设计思想，通过微服务云架构和平台将分散的组件进行融合，便于我们对系统进行管理和部署，同时使得功能的交付也变得更加简单。

2.2.2　Git

Git 是 Linus Torvalds 为了更好地管理 Linux 内核而开发的一个开放源码的控制工具，它可以帮助开发人员高效敏捷地开发各种项目 [26]。与 SVN，CVS 等集中式版本控制系统不同，Git 采用的是分布式版本库的方式，使得其在无网络的情况下也能进行工作。另外，对于开发人员来说，每个人都有自己完整的版本库，在一定程度上提高了软件开发的安全性。

通过使用 Git，开发人员可以在本地查看历史版本，能通过简单的命令进行版本回退操作。同时，分布式的工作方式使得开发人员在无网络的情况下也能进行软件开发工作，极大地提高了开发人员的开发效率。Git 支持创建分支，开发人员可在自己的分支中进行功能开发，然后再合并到主分支上，这提高了开发人员的相互协作能力。

2.2.3　Spring Boot

Spring Boot 是由 Pivotal 团队在 2013 年开始研发、2014 年 4 月发布第一个版本的全新开源的轻量级框架 [27]。Spring Boot 的目的为了减小企业级开发的复杂性，其通过采用简化配置的方式使得应用的整体搭建和开发过程都变得更加简单。

首先，Spring Boot 框架采用了开箱即用的策略。开箱即用指系统在开发过

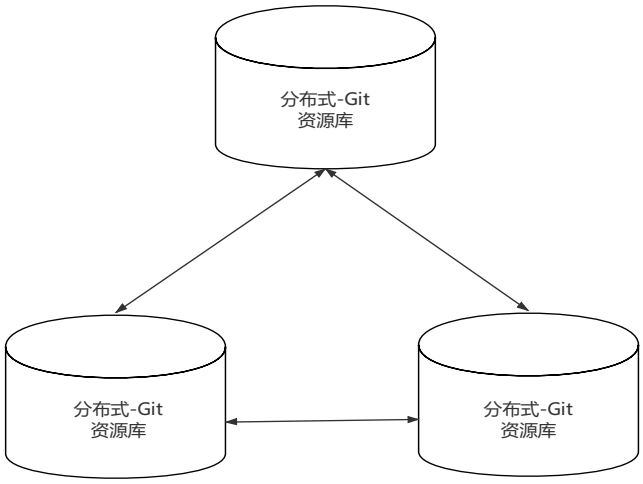

图 2.1: git 分布式结构图

程中，不必通过手动引入 jar 包的形式来添加外部依赖，而是通过在 Maven 项目中的 pom 文件添加对应的依赖，以注解的形式来取代 XML 配置文件配置方式[28]，能更加简单的实现对对象生命周期的管理，同时帮助开发人员免于复杂的配置管理工作，也使得项目更加易于扩展和维护。另外，Spring Boot 框架还采用约定大于配置的策略。约定大于配置是指 Spring Boot 框架本身就设置很多默认配置项，当开发人员在进行软件开发时，可先查看约定配置是否满足需求，如不满足再进行配置。通过采用这种方式，虽然在一定程度上降低了软件开发的灵活性，使得 Bug 定位变得困难，但是它有效的减少了开发人员的工作量，提高了开发人员的工作效率。

本文使用 Spring Boot 作为后端开发框架，在减小了项目开发的复杂程度的同时，可以方便对系统提供的接口进行单元测试。另外，因为其内置了服务器 Tomcat，使得系统的部署变得更加简单。

2.2.4　MyBatis

MyBatis 是 apache 的一个开源项目 iBatis, 2010 年这个项目由 apache software foundation 迁移到了 google code，并且改名为 MyBatis[29]。MyBatis 是一款用来封装 jdbc 的持久层框架，对自定义 SQL 语句、存储过程以及高级映射都提供支持 [30]。MyBatis 通过使用 XML 或者注解的方式来对数据库进行配置，使其能够将原始类型、接口和 Java 中的普通对象映射为数据库对应的字段，免除了开发人员对 jdbc 代码的编写以及参数设置等复杂工作。

下面为 MyBatis 的结构图：

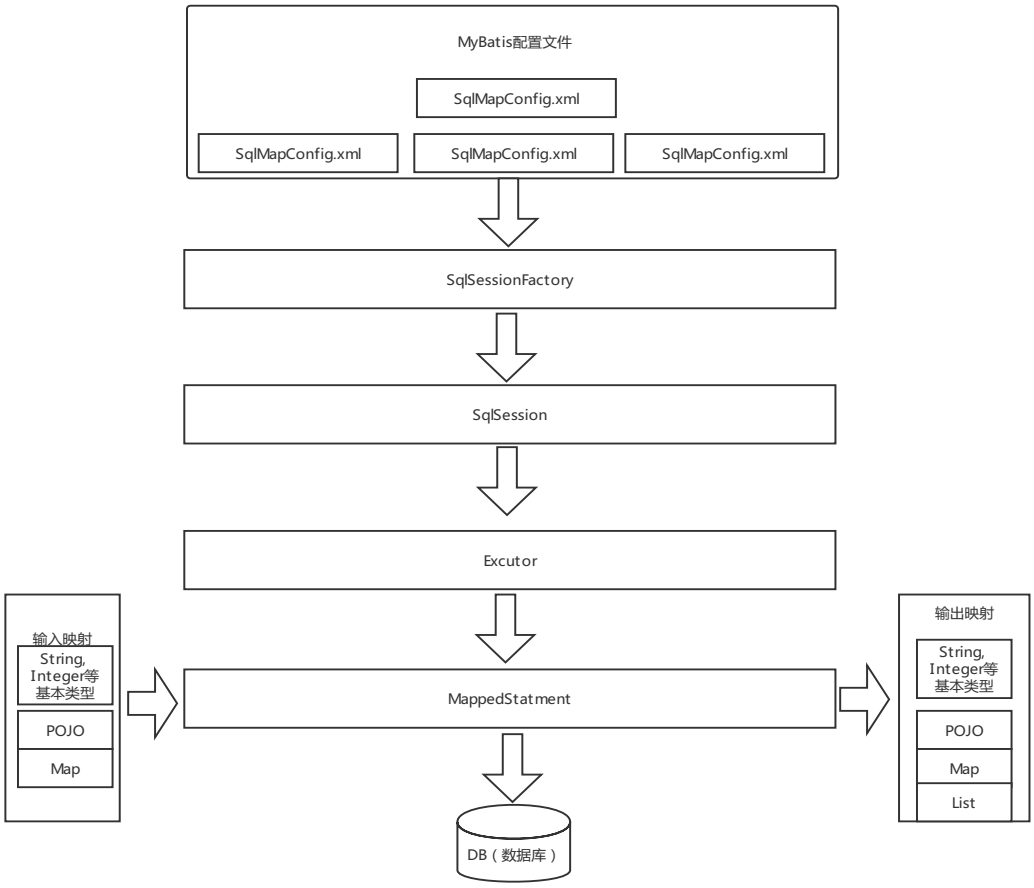

图 2.2: MyBatis 结构图

从图2.2可以看出，SqlMapConfig.xml 是 MyBatis 的核心配置文件，是生成 SqlSessionFactory 对象的基础。SqlSessionFactory 是一个工厂类，用以生成 SqlSession 对象，它是 MyBatis 中至关重要的一个对象，所有的 SQL 语句均通过 SqlSession 对象进行发送并返回 Sql 执行的结果，功能与 jdbc 中的 Connection 对象相似[31]。Excutor 是 SqlSession 中用于执行具体 Sql 语句的底层对象，MapperStatement 也是 SqlSession 的底层对象，它主要用于接收 SQL 语句中的参数信息，即输入映射，并对 SQL 查询语句的执行结果进行映射，即输出映射。

开发人员通过在 Spring Boot 框架中使用 MyBatis 进行数据库操作，则无需编写代码便可以实现对数据库的连接，对于 Statment、ResultSet 等中间对象的复杂操作以及连接异常等问题都可以交由 MyBatis 进行处理。开发人员只要确保 SQL 语句的完整性和正确性，就能方便的实现数据库操作，这极大的提高了开发人员的效率，也保证了系统的稳定性和可维护性。

2.2.5　MariaDB

MariaDB 数据库管理系统由 MySQL 的创始人 Michael Widenius 主导开发，是 MySQL 的一个分支，主要由开源社区在维护，采用 GPL 授权许可 [32]。Michael Widenius 认为 Oracle 可能会将 MySQL 闭源，因此开发 MariaDB 的目的是兼容 MySQL 数据库，包括 API 和命令行，使其能够代替 MySQL。在存储引擎方面，MariaDB 使用 XtraDB 来代替 MySQL 的 InnoDB，另外，它还支持 Maria，PBXT，FederatedX 等多种存储引擎。

MariaDB 通过线程池技术使得其能允许 20 万以上的数据库连接，并且在多连接下数据库性能依旧不受影响，而本系统需要频繁地进行数据库连接和操作，MariaDB 相对于其他数据库能更好地满足需求。

2.3　静态分析技术

Gator

Gator 是由 Atanas Rountev 等人开发的，针对 Android APP 进行静态分析的开源工具。Gator 工具包中主要有两个组件，第一个组件为 Android 软件中的 GUI（图形用户界面）对象开发了静态参考分析，第二个组件是对用户事件驱动的回调的控制流分析。

传统软件是基于框架的事件驱动程序，而 Android 应用程序则是通过图形用户界面驱动，其中 GUI 对象用来对用户的操作进行响应。Atanas Rountev 等人通过对用户事件驱动的组件和从 Android 框架到应用程序代码的生命周期回调和时间处理程序回调进行调研，提出一种捕获该类回调序列的程序表示形式，即通过对上下文兼容的过程控制流路径进行遍历，在此过程中标记可能触发回调的语句来实现图形的可达性 [33]。

本系统通过在 Linux 操作系统中使用 Gator 命令对提交的待测 Andorid APP 进行静态分析，将分析结果存入指定文件中。

2.4　推荐方式

2.4.1　冷启动

冷启动主要是为了解决如何给新用户做个性化推荐的问题 [34]。在本系统中，为了提高众包工人在 Android APP 中进行众包测试时的测试速度和测试效率，众包工人可以选择进行异常界面推荐或未覆盖界面推荐。若众包工人是首次使用，系统尚未记录其相关信息，则会以冷启动的方式向其进行异常界面推荐和未覆盖界面的推荐。对于异常推荐，我们主要考虑以下几点：

- 异常界面测试次数：众包测试的整个过程都会记录异常界面被测试的次数，一方面记录测试次数是为了在次数达到规定标准时对该异常界面进行确定性标记，表明该异常已经通过足量众包工人的测试，其异常已经可以根据现阶段的测试结果进行确定，无需再推荐给其他众包工人，避免异常重复性测试，节省众包资源。另一方面，在未达到测试量之前，测试量多的异常用例代表了大多数众包工人的众包测试倾向，具有一定程度的代表性，更可能吸引初次进行众包测试的众包工人，能够在一定程度上提高异常推荐的有效性。

- 异常界面严重等级：异常界面都有其对应的异常类型，为了更加高效地进行异常推荐，我们对异常等级进行了详细的分类。表2.1为异常的类型已经对应异常的等级：

表 2.1: 异常类型及严重程度

异常类型	**代表异常**	**严重程度**
资源加载错误	java.lang.IllegalAccessException	10
	android.system.ErrnoException	10
	android.database.sqlite.SQLiteException	10
	java.lang.FileNotFoundException	9
	NameNotFoundException	8
输入意图错误	java.lang.ArrayIndexOutOfBoundsException	10
	java.lang.NoSuchMethodException	8
	java.lang.NullPointerException	8
	java.lang.ClassCastException	7
	java.lang.ClassNotFoundException	7
	java.lang.IllegalArgumentException	6
网络通信异常	java.net.ConnectException	6
	java.net.SocketException	5
	NetworkError	5
ANR	Service Timeout	6
	ContentProvider Timeout	5
	inputDispatching Timeout	5
	BroadcastQueue Timeout	4
其他异常	java.io.IOException	6
	ExecutionException	4

在初始阶段不知众包工人喜好的情况下，系统若想短时间内最大化众包工人的价值，就需要在冷启动阶段高比例的向众包工人推荐异常等级较高的界面，让严重程度最高的异常得到充分测试。

- 众包工人手机品牌和系统版本：Android 系统的版本繁多，主要集中在 4.0-11.0 之间，碎片化问题严重。另外，依托于 Android 系统，各个手机制造商对 Android 系统进行闭源再开发，再次加重了碎片化问题。其结果是，在 A 品牌手机中测试的异常在 B 品牌手机中却不能复现，不仅严重浪费众包资源，还会造成众包测试结果误差过大。为了更加精确的向众包工人进行异常推荐，系统根据其手机品牌和系统版本，寻找与自动化测试中机型相近或系统版本相近的测试结果，将其选择性推荐给众包工人，一方面减小了结果上报误差，另一方面也提高了众包测试整体效率。

2.4.2　协同过滤

协同过滤算法是依托于用户历史行为数据，挖掘用户的喜好偏向，并预测用户可能喜好的信息进行推荐 [35]，用户通过合作的机制给对推荐的信息给与相当的回应并将之记录下来以便达到过滤的目的，从而更好地对信息进行筛选。回应的不一定都是特别感兴趣的，对于不感兴趣的记录也是非常重要的。协同过滤主要分为两类：基于用户的协同过滤算法 (user-based collaboratIve filtering)，以及基于物品的协同过滤算法 (item-based collaborative filtering)[36]。由于基于用户的协同过滤算法在用户量较大时，用户相似度的计算将变得复杂，在本系统中，采用的是基于物品的协同过滤算法来进行异常界面或未覆盖界面的推荐。

基于物品的协同过滤算法 [37] 包括以下几个步骤：

（1）计算物品的相似度, 在本系统中任务的相似度是通过公式2.1进行计算。其中 N(i) 表示选择任务 i 的用户数量，N(j) 表示选择任务 j 的用户数量，w_{ij} 则表示任务 i 与任务 j 的相似度。

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

（2）通过冷启动阶段收集的用户的测试数据，预测用户对其他任务的感兴趣程度，感兴趣程度通过公式进行计算2.2。其中 N(u) 表示用户选择任务的集合，S(j,k) 表示与任务 j 相似的物品集合，数量为 k，r_{ui} 表示用户对物品 i 的兴趣程度，在本系统中，用户选择任务 i，r_{ui} 为 1，否则为 0。p_{ui} 则表示用户对任务的感兴趣程度。

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

（3）通过使用上述算法，获得用户对当前任务的感兴趣程度列表，对列表从大到小进行排序，选择 TopN 个任务作为推荐任务推荐给用户。

2.5　本章小结

本章主要介绍系统使用的相关技术框架、工具和算法，通过将其分为 Android 端技术、后端技术、静态分析技术以及推荐方式来进行详细描述。Android 端主要包括使用 OKHTTP 框架来进行数据请求，使用 CrashHandle 来捕获应用程序运行崩溃事件，使用 FloatingActionButton 制作悬浮按钮。后端主要包括使用微服务的方式进行服务部署，用 Git 进行版本控制，使用 Gitlab 来保存和管理代码，使用 Spring Boot 框架进行项目开发。数据库端使用 MariaDB 来保存数据。静态分析技术使用开源工具 Gator 对 Android 移动应用进行静态分析。推荐方式采用冷启动和协同过滤两种方式，在用户初次进行众包测试时，使用冷启动进行推荐，之后就根据用户的历史行为，使用协同过滤的方式进行任务推荐。

第三章　　系统需求分析与概要设计

3.1　系统总体概述

随着科学技术的发展，智能手机已经成为我们生活中必不可少的物品。当今市场上，Android 操作系统的市场份额远超其他操作系统，大幅度居于领先地位，因此 Android 应用的数量也急剧增长。水涨船高，Android 应用数量的增加也对 Android 移动应用测试工作提出了更高的要求。

现有的主流的测试方法主要有 Android 自动化测试和众包测试。对于自动化测试来说，通过编写相应的测试脚本，就可在不同的机型上实现对 Android APP 进行自动化的测试，极大地节省了人力资源。但是，Android 系统的版本多种多样，版本之间的兼容性也模糊不清，使得其存在严重的碎片化问题，导致自动化测试也很难灵活地适应于不同的系统，测试结果误差变大。而众包测试通过将众包任务下发给众包工人，由他们进行测试并完成结果上报。虽然众包测试比传统的测试方法省时，省钱，但是因众包工人背景不一，测试专业程度不一，使得他们在众包测试过程中的表现良莠不齐，且会存在重复测试，测试页面不全，上报信息描述不清等多种问题。本系统通过结合 Android 自动化测试和 Android 静态分析，对众包测试的过程加以引导，能有效减小系统碎片化，重复测试等问题，并能弥补众包工人测试能力差异大的不足，提高众包工人的测试速度和效率。

如图3.1本系统首先对 Android APP 在不同的机型上进行自动化测试，收集其异常信息，并对异常信息的类型进行分类，对严重程度进行评估，然后将所有非重复性异常数据存入数据库中作为异常复现的依据。然后，通过 Android 静态分析工具对 Android APP 进行静态分析，找出自动化测试未覆盖的页面，并通过人工辅助的方式对未覆盖页面进行截图标记，将信息进行整合后存入数据库作为未覆盖界面引导的依据。众包工人使用装有引导工具插件的 Android APP 进行测试，他们可选择异常信息推荐和未覆盖界面推荐两种方式去验证异常和发现异常。为了加快众包测试速度，众包工人可选择对路径进行引导提示，更高效的完成验证，并进行结果上报。对于众包工人自己发现的异常，也可直接通过截图加异常描述的形式上报至服务器，经过分析处理后再将该异常信息作为新的异常下发给众包工人，形成回环测试的效果，提高测试准确性。

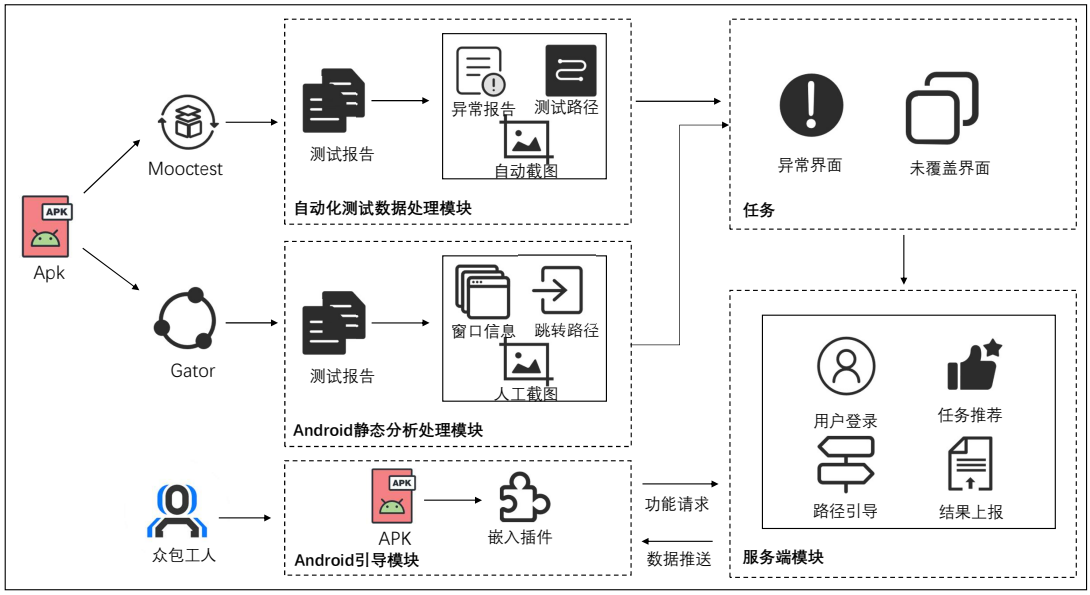

图 3.1: 系统设计图

3.2　系统需求分析

3.2.1　涉众分析

本系统的涉众分析结果如表3.1所示。本系统主要涉及到两个方面的涉众，一方面是测试需求方，另一方面是众包工人。首先，对于测试需求方来说，通过使用该系统要能在 Android 应用测试覆盖率，异常测试结果准确性方面相较传统测试方法要有着显著的提高，能够根据测试结果进行异常的正确排查和解决。其次，对于众包工人来说，众包测试引导工具要具有异常界面推荐，未覆盖界面推荐，路径引导，异常上报等功能，要在不影响众包测试的基础上，有效提高他们的工作效率，弥补其测试专业性不足的缺点。

3.2.2　功能需求

本系统需要对提交的 APK 进行静态分析，对获得的自动化测试数据进行异常提取，结合对系统进行的涉众分析，可对系统进行功能需求分析，本系统的功能需求如表3.3所示：

表 3.1: 涉众分析结果

涉众名称	涉众特征与期望
测试需求方	作为测试需求方，他们希望已提交的 Android APP 能够在多种不同品牌不同版本的机器上进行测试，测试出的异常能够得到众包工人的有效验证，并有相应的测试结果。在同样的测试成本下，期望相较传统的测试方法在测试覆盖率方面能有大幅度的提高。
众包工人	作为众包测试的主要人员，众包工人希望引导工具能够进行个性化的异常推荐和未覆盖界面推荐，能够正确的引导众包工人通过最短路径达到待测页面，且能够简单高效的实现异常信息上报。

表 3.2: 系统功能分析列表

需求 ID	**需求名称**	**需求描述**
R1	自动化测试数据处理	根据在不同机型上测试的自动化测试结果能够适配提取出异常信息，并将图片根据信息描述进行标注
R2	静态分析处理	根据已提交的 APK 数据包，进行静态分析，找出自动化测试未覆盖部分，构建窗口转换图
R3	用户登录	根据众包工人的账号和密码，对其身份进行验证，确保数据来源的可靠性和正确性
R4	异常界面推荐	根据异常的类型和严重程度，计算异常界面的权值，再结合众包工人在众包测试过程中提交的信息，进行个性化 TopN 异常界面的推荐
R5	未覆盖界面推荐	根据静态分析处理构建的窗口转换图，当众包工人进行未覆盖界面推荐时，能根据其当前所处的界面位置，进行单步骤未覆界面的推荐，引导工人进入未覆盖界面中并进行测试
R6	路径推荐	根据众包工人当前所处的界面位置，以逆路径方式寻找进入异常界面最短路径，推荐给众包工人下一步应该点击位置的引导提示
R7	测试结果上报	根据众包工人测试的结果和异常信息进行比对，提交当前异常的测试结果上报到服务器。对于众包工人在测试过程中主动发现的异常信息，以截图加描述的形式上传到服务器

本系统首先根据在不同品牌不同版本 Android 手机中测试出的信息，进行异常数据的模式匹配，提取所有非重复性异常信息，并通过自动化标注的形式，将异常信息存储到数据库中。之后通过静态分析工具 Gator 对 APK 进行静态分析，获得其静态分析结果，结合自动化测试结果，以模式匹配的方式找出未覆盖界面，构建对应 APP 的窗口转换图。众包工人在使用其账号登录系统后，使用装有引导插件的 APP 进行众包测试，可以进行异常界面推荐、未覆盖界面推荐、路径推荐，以提高众包测试的效率。众包工人需要进行测试结果上报，上报的内容需要包括异常界面测试的结果以及主动测试发现的新异常，作为评估测试覆盖率和修改应用中存在异常的依据。

3.2.3　非功能需求

本系统需要提供给众包工人使用，所以在保证功能完善的同时，要对系统的安全性，可靠性，可用性，易用性等非功能性需求提出高要求。结合本系统的特点，经过深入的考虑，本系统的非功能性需求如表3.3所示：

表 3.3: 系统功能分析列表

需求 ID	需求名称	需求描述
NR1	可靠性	本系统应该在 APP 出现崩溃时仍能将错误信息上传至服务端，保证测试数据的完整性。在系统运行过程中，服务端的故障率要小于总运行时间的 1%
NR2	可用性	本系统应该提供高可用服务，在因外界因素造成的服务端进程挂掉时，能够通过守护进程及时启动服务。因众包工人操作造成的错误，能给与提示并修正其的错误
NR2	安全性	本系统应该具备严格的用户身份检测，以防止数据被污染。对于存储在数据库中的数据，不允许随意被进行修改，查看数据时只能以只读的方式进行
NR4	易用性	本系统应该具有良好的引导作用，对于初次使用该系统的众包工人，也能在短时间内熟悉其功能。系统引导的方式要符合众包工人的理解能力，使其能借助系统高效地进行测试工作
NR5	性能	本系统对于所有的用户请求，都必须在 200ms 内完成响应，加载引导图片的数据延迟不能超过 1s
NR6	可维护性	本系统各部分之间的高度解耦，每个部分都有单独的测试，具备良好的可维护性
NR6	可拓展性	本系统现阶段的推荐是基于异常和未覆盖界面的推荐，后期可能需要增加其他方向的推荐，推荐的方式也可能根据不同的用户采用不同的方式

本系统需要提供给背景不同，测试能力不同的众包工人使用，且众包工人在使用过程中需要实时引导，因此要让众包工人能尽快熟悉工具并熟练使用，所以要保证系统的易用性。系统的使用者数量众多，且存在短时间内高并发，高请求情况的存在，需要系统在众包工人活跃的时间内，必须保持高可用、高可靠。众包工人在引导过程中需要实时进行引导请求，因此服务端的响应时间要尽可能的短，引导图的加载时间也要在众包工人可接受范围内，因此需要系统具有良好的性能。众包工人上报到服务端的数据是进行测试覆盖率统计，修改应用存在异常的关键，所以众包工人的身份需要严格验证，避免数据污染。同时，数据统计只能以只读的方式进行，不允许以任何方式对数据进行修改。本系统现阶段只有异常推荐和未覆盖推荐，后期可能需要增加更多特色推荐，因此系统需要具备良好的可维护性和可拓展性，以适应不断变化的需求。

3.3　系统用例分析

3.3.1　系统用例图

系统用例图是指由参与者（Actor）、用例（Use Case），边界以及它们之间的关系构成的用于描述系统功能的视图 [38]。通过对系统涉众进行分析，并对系统的功能需求和非功能需求进行探究，本系统的系统用例图如图3.2所示。对于测试人员，只有数据处理一个用例，数据处理又包括了对 Android 应用的自动化测试数据处理和静态分析数据处理。对于众包工人，共有用户登录、异常界面推荐、未覆盖界面推荐、路径推荐和测试结果上报等五个用例，其中测试结果上报是包括异常验证报告，新异常上报和 APP 崩溃上报。

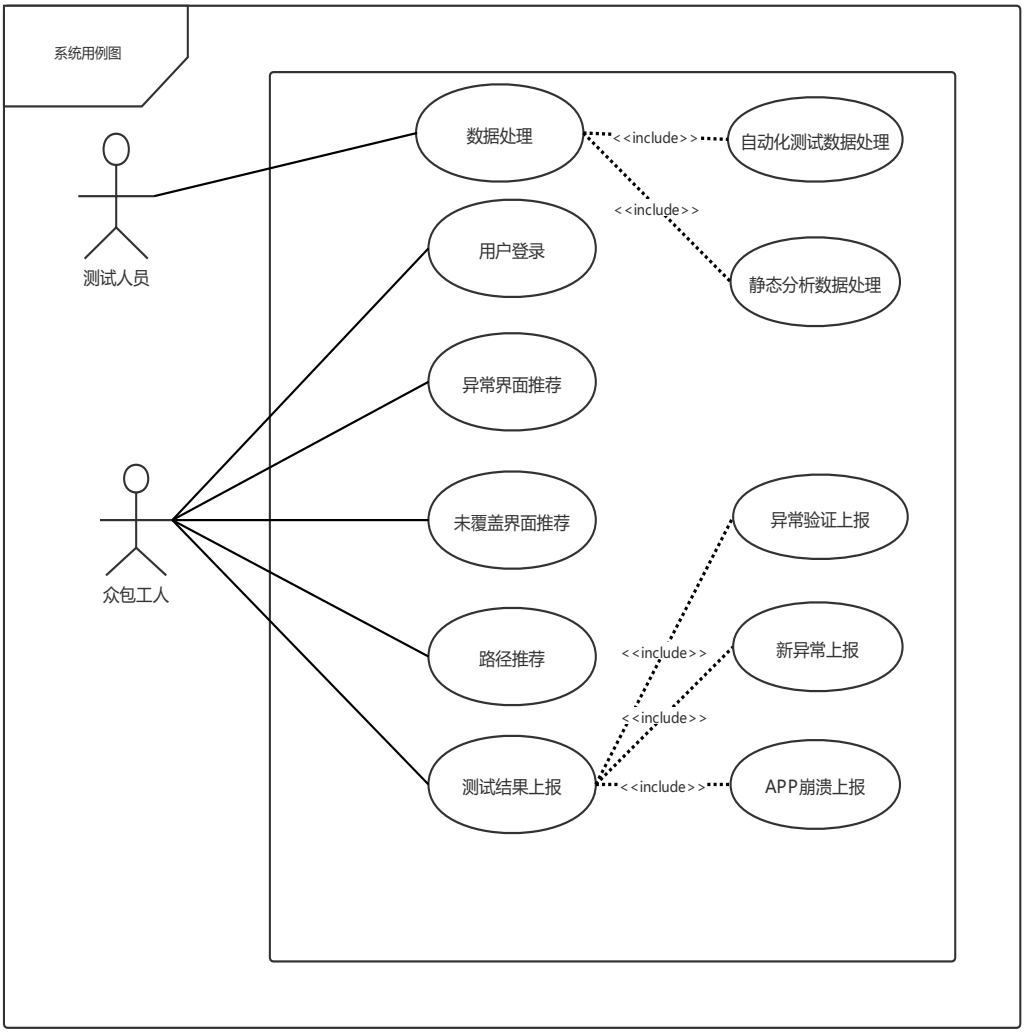

图 3.2: 系统用例图

3.3.2　系统用例分析

自动化测试数据处理是本系统的基础，系统的异常界面推荐和路径推荐都要根据自动化测试处理的数据为基础。测试需求方需要先获取待测 Android APP 在不同机型上的测试信息，通过对该测试信息进行分析，提取异常界面信息和到达异常界面的路径信息，该信息主要用来对众包工人的复现异常过程提供步骤引导。同时，该信息中包括异常发生时的坐标点位信息，系统会通过该坐标位置对自动化测试结果中截取的图片信息进行标记，以更加明确的方式告知众包工人应该点击的位置，提高测试效率。自动化测试数据处理的详细用例描述如表3.4所示。

表 3.4: 自动化测试数据处理用例描述

ID	**UC1**
名称	自动化测试数据处理
参与者	测试人员
触发条件	开启自动化数据处理模块
前置条件	测试人员必须已经获取自动化测试结果
后置条件	自动化测试数据处理结束
优先级	高
正常流程	1. 测试人员获取自动化测试结果 2. 测试人员启动自动化测试数据处理模块 3. 数据处理流程开始对数据进行模式匹配，提取所需信息，并根据提取的信息进行截图标记 4. 将处理过的数据存入数据库
异常流程	1. 自动化测试数据处理模块启动失败 2. 自动化测试数据格式不正确，需要 json 格式 3. 自动化测试数据中点位信息不正确，截图标记有误差

静态分析数据处理是本系统的重要组成部分，通过开源的静态分析工具 Gator 对 Android APP 进行分析，能够提取该应用的所有界面信息和窗口跳转信息，通过窗口跳转信息便可以构建该应用的窗口转换图。再通过与该应用的自动化测试数据处理模块得到的数据进行对比，提取出该应用在自动化测试过程中未覆盖的窗口信息，存入数据库中。对于未覆盖界面的截图，需要通过人工的方式进行截取标记，作为未覆盖界面引导的依据。静态分析数据处理的详细用例描述如表3.5所示。

表 3.5: 静态分析数据处理用例描述

ID	**UC2**
名称	静态分析数据处理
参与者	测试人员
触发条件	开启静态分析处理模块
前置条件	测试人员必须已经获取自动化测试处理结果
后置条件	静态分析处理结束
优先级	高
正常流程	1. 测试人员获取自动化测试处理结果 2. 测试人员通过使用 Gator 对提交的 Android APP 进行静态分析 3. 测试人员启动静态分析处理模块，根据比对自动化测试数据和静态分析数据的内容提取未覆盖界面信息 4. 将处理过的数据存入数据库
异常流程	1.Gator 无法处理已提交的 Android APP 2. 静态分析模块启动失败 3. 静态分析模块提取过多无关数据

Android 引导模块是待测 Android APP 均需要嵌入的插件代码，众包工人通过点击引导按钮来触发各个模块的功能。对于在测试过程中可能发生的系统崩溃而强制退出的场景，该引导模块可以自动将异常信息上报服务端。Android 引导模块的详细用例描述如表3.6所示。

表 3.6: Android 引导模块用例描述

ID	**UC3**
名称	Android 引导插件
参与者	众包工人
触发条件	点击用户引导插件按钮
前置条件	众包工人使用带有引导插件的 Android APP
后置条件	进行众包测试
优先级	高
正常流程	1. 众包工人点击引导插件按钮 2. 众包工人在不同界面下均点击引导插件按钮下的隐藏按钮，进行众包测试 3. 众包测试过程中发生系统崩溃而强制退出
异常流程	1.Android 引导按钮点击无反应 2. 在个别窗口中引导按钮没有出现 3. 发生系统崩溃时，服务端未收到异常报告信息

用户登录是众包工人使用该引导工具的前提，通过身份验证确保其身份的安全性，并保证测试数据来源的可靠性。用户登录的详细用例描述如表3.7所示。

表 3.7: 用户登录用例描述

ID	**UC4**
名称	用户登录
参与者	众包工人
触发条件	点击用户登录按钮
前置条件	众包工人的使用带有引导插件的 Android APP
后置条件	用户登录
优先级	中
正常流程	1. 众包工人点击登录按钮 2. 输入账号密码后，点击确认 3. 用户登录成功
异常流程	1. 用户账号密码错误 2. 用户忘记密码 3. 用户非法登录

异常界面推荐是系统的主要功能之一，主要是根据众包工人的测试过程，个性化的向众包工人推荐不同的异常界面。众包工人点击推荐按钮，选择异常用例，然后根据异常信息提示去验证异常是否存在。异常界面推荐的详细用例描述如表3.8所示。

表 3.8: 异常界面推荐用例描述

ID	**UC5**
名称	异常界面推荐
参与者	众包工人
触发条件	用户点击异常界面推荐按钮
前置条件	众包工人成功登录
后置条件	系统弹出异常界面推荐列表
优先级	高
正常流程	1. 众包工人点击异常界面推荐按钮 2. 选择一个测试用例，进行异常验证 3. 若发现有异常就提交异常信息
异常流程	1. 异常界面重复推荐 2. 异常界面不存在

未覆盖界面推荐是系统的另一主要功能，主要是根据众包工人当前所处的位置，就近向众包工人推荐未覆盖的界面。众包工人点击未覆盖页面推荐按钮，选择未覆盖用例，根据提示进入未覆盖界面进行测试，主动发现异常。未覆盖界面推荐的详细用例描述如表3.9所示。

表 3.9: 未覆盖界面推荐用例描述

ID	**UC6**
名称	未覆盖界面推荐
参与者	众包工人
触发条件	用户点击未覆盖界面推荐按钮
前置条件	众包工人成功登录
后置条件	系统弹出未覆盖界面推荐列表
优先级	高
正常流程	1. 众包工人点击未覆盖推荐按钮 2. 选择一个测试用例，进行未覆盖界面进行测试 3. 若发现有异常就提交异常信息
异常流程	1. 未覆盖界面重复推荐 2. 未覆盖界面无法进入 3. 系统崩溃, 强制退出

路径推荐是系统的重要组成部分，其主要功能是对众包工人的测试流程加以路径引导，以加快众包工人测试的速度，提高测试效率。众包工人在选择完测试用例后，点击路径引导，弹出下一步路径提示。路径推荐的详细用例描述如表3.10所示。

表 3.10: 路径推荐用例描述

ID	**UC7**
名称	路径面推荐
参与者	众包工人
触发条件	用户点击路径推荐按钮
前置条件	众包工人选择测试用例
后置条件	系统弹出路径推荐提示
优先级	高
正常流程	1. 众包工人点击路径按钮 2. 系统弹出路径推荐提示，众包工人点击确定 3. 进入下一界面
异常流程	1. 路径推荐失效 2. 推荐的路径不存在

测试结果上报是系统的主要功能，主要根据众包工人测试过程中发现的异常信息，根据提示上报到服务端。测试结果上报主要有异常信息验证结果上报，和新异常信息结果上报，测试结果上报的详细用例描述如表3.11所示。

表 3.11: 测试结果上报用例描述

ID	UC8
名称	测试结果上报
参与者	众包工人
触发条件	用户点击测试结果上报按钮
前置条件	众包工人成功登录
后置条件	测试结果上报到服务端
优先级	高
正常流程	1. 众包工人点击测试结果上报按钮 2. 选择上报异常信息类型，并进行异常描述 3. 进行测试结果上报
异常流程	1. 测试结果上报失败 2. 截图功能有误差

3.4　系统总体设计

3.4.1　系统架构设计

系统架构图是系统框架的总体展示，主要对系统的组成部分、逻辑结构进行描述，可以作为详细设计的依据 [39]。本系统的系统架构图如图3.3所示，系统主要分为 Android 端引导模块，服务端响应模块，自动化测试数据处理模块，静态分析处理模块等几个不同的部分。

本系统的上游设备主要是众包工人使用的不同品牌，不同版本的 Android 手机，通过这些移动设备，在装有引导模块的 Android APP 中以 OkHttp3 为网络框架向服务端请求众包测试引导，所有请求结果以 JSON-RPC 格式进行响应，以图片加信息描述的形式展示在众包工人手机中。系统服务端使用的是 Spring Boot 开发框架，该框架具有开箱即用，没有代码生成，也无需 XML 配置等特点，可以使我们更加快速地进行项目开发。Spring Boot 通过连接 MariaDB 数据库，以 WebSocket 的形式向众包工人提供用户登录、异常界面推荐、未覆盖界面推荐、路径推荐、结果上报等接口，供众包工人进行请求调用，是系统最重要的模块。

本系统要依赖于慕测自动化测试平台，通过提交待测 Android APP 给慕测平台，可以获得其在不同设备上的自动化测试结果，测试结果以 json 的形式存储在系统文件中。服务端则通过自动化数据处理模块，遍历文件并以模式匹配的方式提取其中异常界面信息，根据异常类型和异常名称评估异常等级，并存储至数据库中。同时，系统还需要使用静态分析工具 Gator 对 Android APP 进行静态分析，将分析结果与自动化测试结果进行比对，构建窗口转换图，并找出自动化测试中未覆盖的路径，将未覆盖信息存储在数据库中。系统的外部存储主要使用 MariaDB 数据库和 Linux 系统文件。MariaDB 数据库可以接受高并发的

请求，更好地满足众包测试的需要。Linux 文件则主要保存在自动化测试和静态分析过程中产生的中间数据。

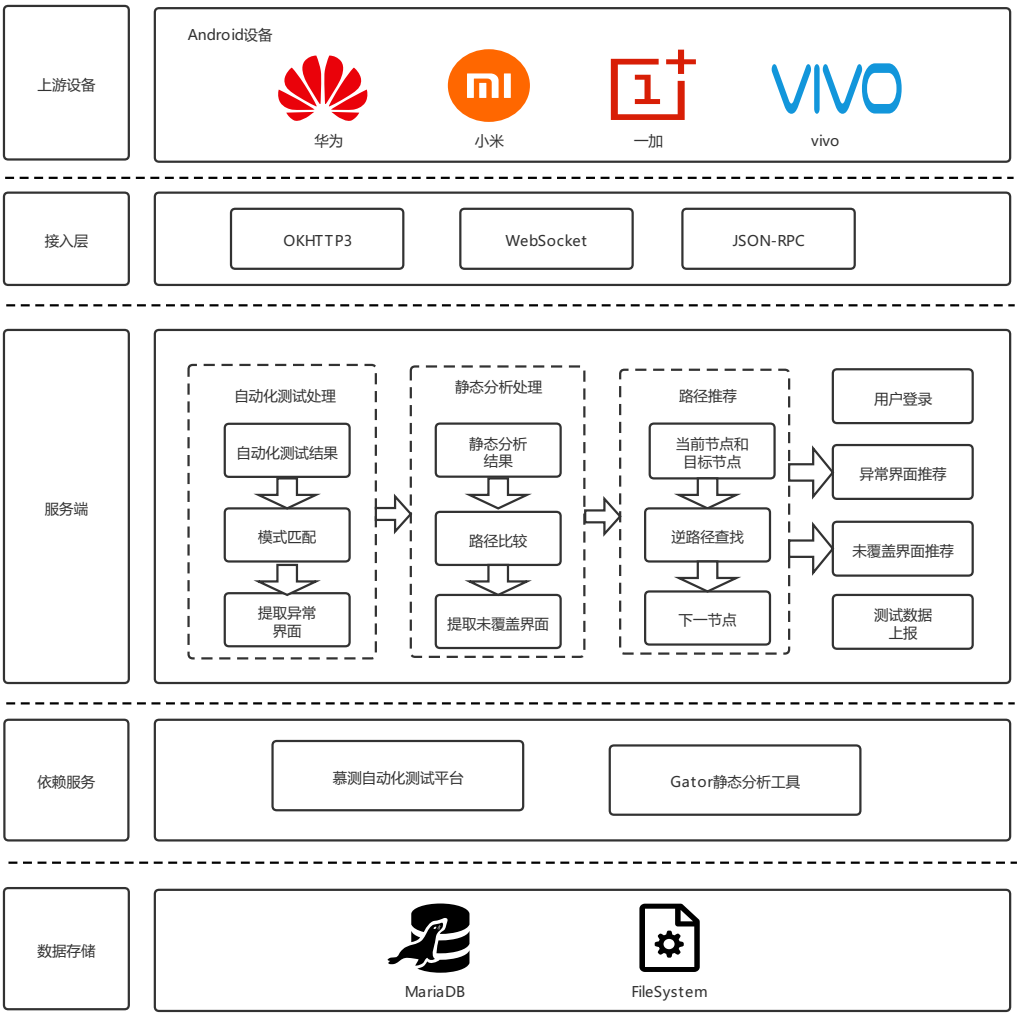

图 3.3: 系统架构图

3.4.2　4+1 视图模型

“4+1” 视图，即逻辑视图，开发视图、进程视图、物理视图和场景视图 [40]，是由 Philippe Kruchten 教授提出，是对逻辑结构进行描述，是架构设计的结构标准。方法的不同架构视图承载不同的架构设计决策，支持不同的目标和用途 [**?**]。对于场景视图，图3.2已经分析过，本文主要对另外四个视图进行分析。

（1）逻辑视图

逻辑视图可以用来描述系统的功能需求，即系统提供给最终用户的服务，本系统的逻辑视图如图3.4所示。FloatingActionButton 是众包工人使用引导插件功能的主要实现类。UserService 类主要是用来实现用户登录的类，User 是用户信

息类，UserService 通过调用 User 类来实现用户信息的存储和验证。EdgeService 是实现异常界面推荐和未覆盖页面推荐的类，推荐信息的实体是通过 Edge 类来实现。Edge 类所封装的推荐信息由自动化测试结果处理类 Edge(Auto) 和静态分析处理类 Gator 共同处理得到。PathService 是路径引导类，引导信息以 Edge 类的形式返回给前端进行响应。BugService 类是用来处理众包工人提交的测试结果，测试报告信息以 BugInfo 的形式提交到后端，并由 BugService 类进行处理。

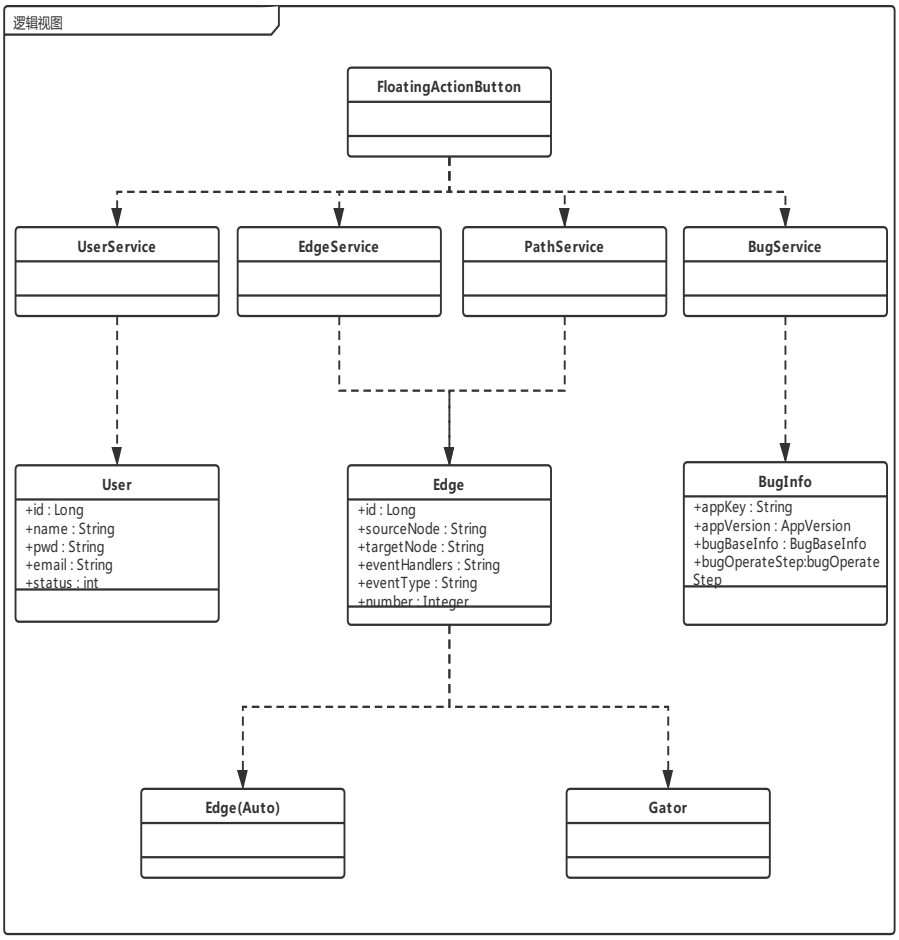

图 3.4: 逻辑视图

（2）开发视图

开发视图主要描述软件在开发环境下的静态组织，如图3.5所示，系统的开发视图可以分为三层。在展示层中，主要是 Android 端引导插件的类组织结构。所有的类都在 Bughunter 包中，包括引导工具按钮类 FloatingActionButton，异常截图上报类 Camera，系统崩溃上报类 CrashHandler 等。在业务逻辑层中，Controller 是控制类，主要是对前端的请求进行响应，并控制业务逻辑的执行顺序。Service 层是业务层，是由 Controller 层控制执行任务的实体，也是系统的核心。Dao 层

主要是通过对底层数据的封装，借助 SQL 语句实现对数据库的数据交互，包括读取和写入，同时对 Service 层提供数据支撑。Model 包中是系统用到的实体对象，wrapper 包则是对 Model 包中的实体对象进行再封装，是数据响应的主要形式。util 包中包含各种工具类，以保证更高效的实现系统服务。数据存储层主要使用的是 MariaDB 数据库和文件系统，MariaDB 主要是对需要被验证的异常信息、引导信息、未覆盖信息和众包测试过程中产生的数据进行存储，文件系统主要是对自动化测试数据和静态分析数据进行存储。

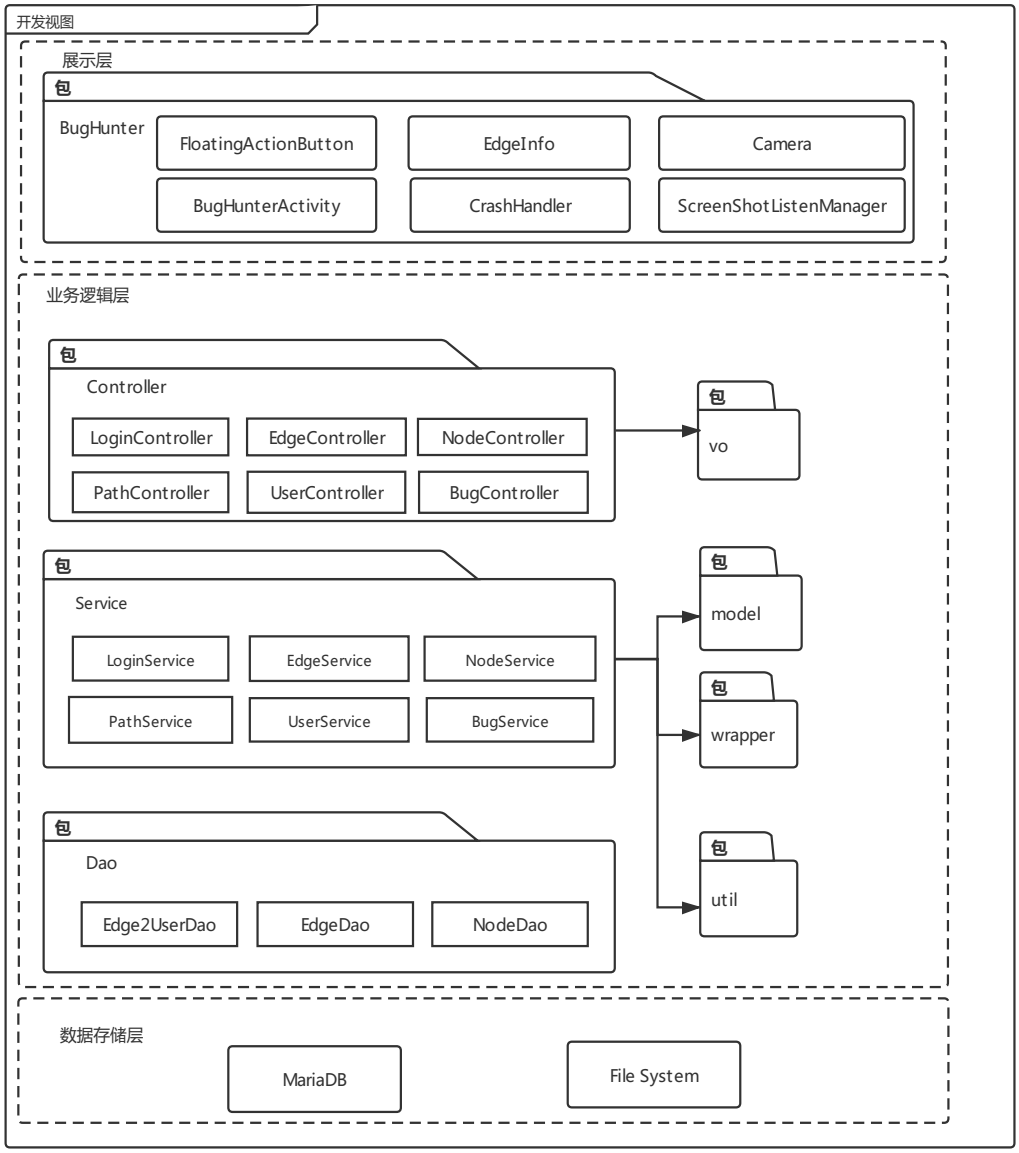

图 3.5: 开发视图

（3）物理视图

物理视图也叫做部署视图，从系统工程师的角度去解读系统，主要关注点在于软件的物理拓扑结构，以及通过部署机器和网络来配合软件系统的可靠性、性能和可伸缩性 [41]。本系统物理视图如图3.6所示。静态分析工具 Gator 部署在单独的服务器中，在对 Android APP 分析完成后以 HTTP 的形式将数据发送至系统服务器。自动化测试工具是独立系统，系统使用的是慕测自动化测试平台测试的数据。众包工人使用不同的 Android 设备，通过 Android APP 中的引导插件来访问系统服务器提供的接口。服务端程序部署在系统服务器中，服务器通过特定的端口对外提供服务。数据存储部分使用的是 MariaDB 数据库，在保证数据安全性的同时又使得数据具有较好的可移植性。在众包测试过程中需要图片引导，而图片则单独存储在云服务中，以便快速地对用户的请求进行响应，减少图片加载延迟。

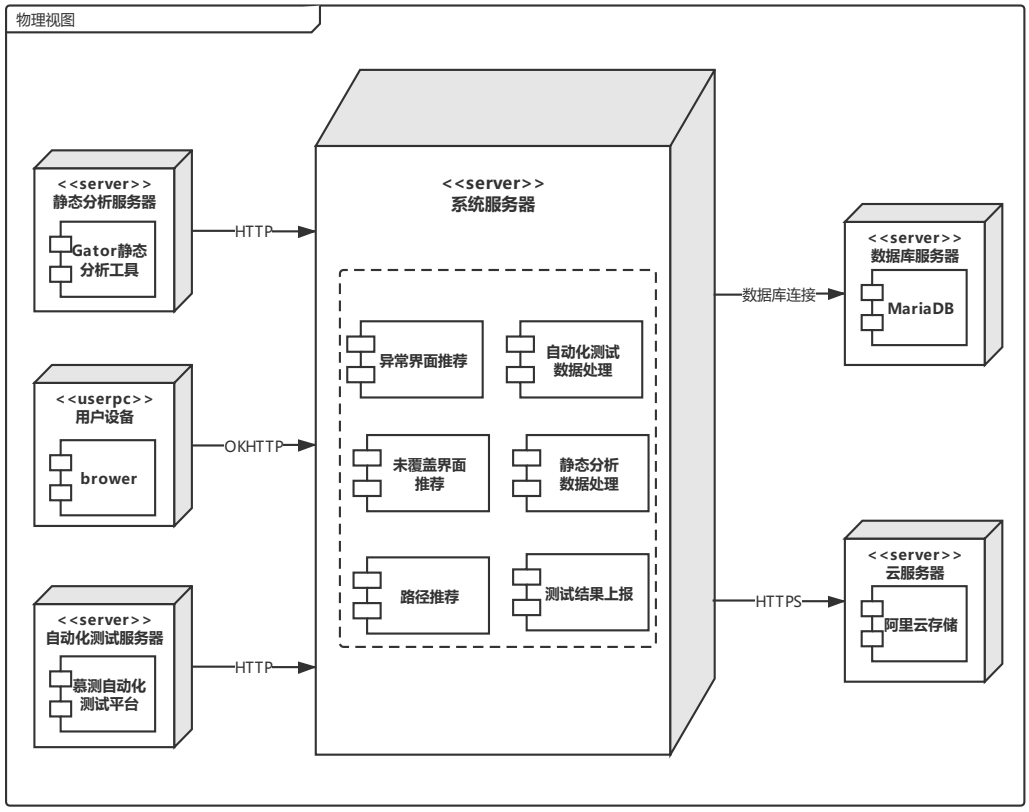

图 3.6: 物理视图

（4）进程视图

进程视图侧重于系统的功能特性，主要描述系统的并发和同步特性，以解决进程、线程、通信等各种问题 [42]。本系统的进程视图如图3.7所示。主线程

首先启动自动化测试结果处理进程，并将处理结果存入数据库中。然后启动静态分析进程，静态分析进程先对 Android APP 进行静态分析处理，然后再将处理结果和自动化测试处理结果进行比对，将比对结果存入数据库。服务进程是系统的核心进程，所有服务都是由服务进程来提供，服务进程会在运行时调用图片提供进程，再从数据库中读取数据后就将数据返回主进程。

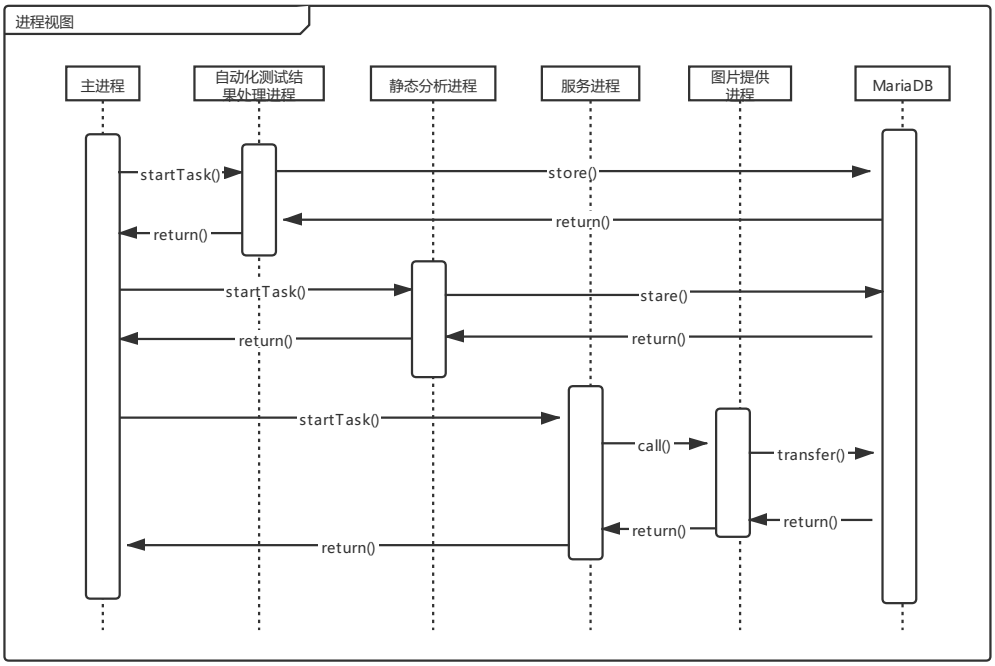

图 3.7: 进程视图

3.4.3　数据库设计

根据本系统需要对使用的数据库进行大量的数据添加和数据查询的特点，系统选择使用 MariaDB 数据库作为系统存储持久化数据的方式。MariaDB 数据库使用线程池技术可以满足系统频繁的数据查询，支持多连接操作，同时具有很高的可移植性，很好地满足了系统的需求 [43]。系统程序是事物处理型，因此数据库的设计要考虑发展的需求，设计出的表应具有可拓展性、可伸缩性，同时还需要具有适度冗余。

本系统数据库设计的数据存储表主要有推荐信息表 (edge)、用户信息表 (user)、窗口信息表 (node)、测试结果表 (app_bug_info)、应用信息表 (app_version_info)、设备信息表 (bug_device_info) 等。本系统数据库的实体关系设计如图3.8所示。

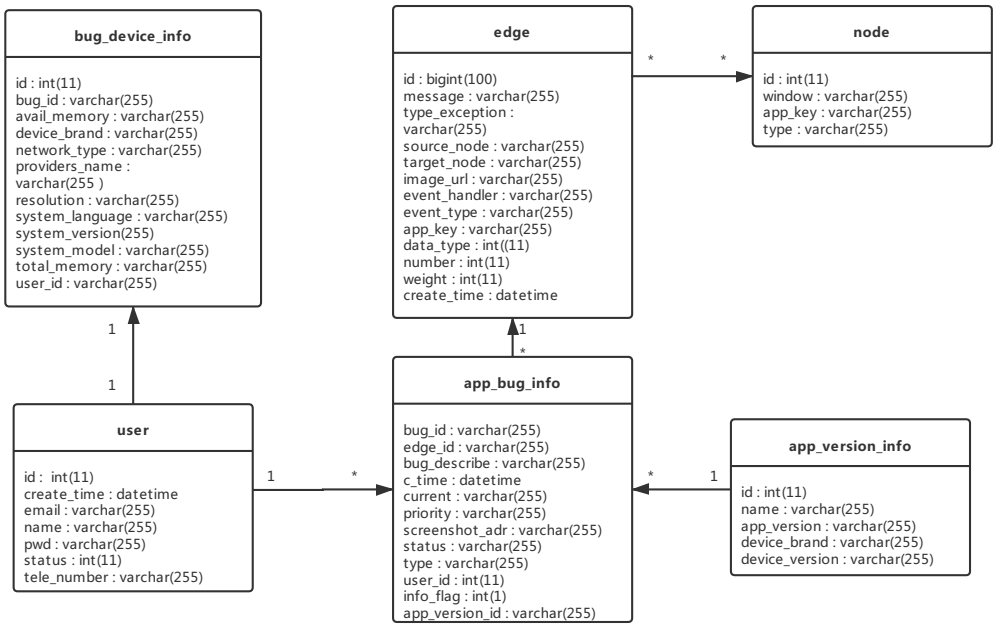

图 3.8: 实体关系图

用户信息表和设备信息表是一对一的关系，默认一个用户只使用一个设备进行众包测试。用户信息表和测试结果表是一对多关系，一个用户可以有多条测试结果，每条测试结果只对应一个用户信息。测试结果表和推荐信息表是多对一关系，一个测试结果对应一条推荐信息，一条推荐信息对应多个测试结果。推荐信息表和窗口信息表是多对多关系，一条推荐信息中有多个窗口，一个窗口对应多个推荐信息。应用信息表和测试结果表是一对多关系，一个应用对应多个测试结果，一个测试结果对应一个应用。接下来分别对用户信息表 (user)、推荐信息表 (edge)、节点信息表 (node)、测试结果表 (app_bug_info)、应用信息表 (app_version_info)、设备信息表 (bug_device_info) 进行详细介绍。

表3.12是用户信息表 (user)，主要用来存储众包工人的相关信息，如邮箱、姓名、密码、电话等信息，以便对众包工人进行身份验证，确保系统的安全性。

表 3.12: 用户信息表 (user)

字段	类型	默认值	说明
id	int	0	用户 id, 主键
create_time	datetime	NULL	创建时间
email	varchar	NULL	邮箱
name	varchar	NULL	用户名
pwd	varchar	NULL	密码
status	int	0	状态
tele_number	varchar	NULL	手机号

表3.13是推荐信息表 (edge)，用以存储异常界面、未覆盖界面和路径推荐界面等信息，主要字段有异常描述、异常等级、出发点、目标点、截图访问地址、事件类型、触发方式、测试应用、信息类型、测试次数、权重和创建时间。其中不同的信息用 data_type 字段来区分，0 表示未覆盖界面，1 表示正常路径界面，2 表示异常路径界面。source_node 代表起始节点，target_node 是目标节点，当用户根据 event_type，即触发方式的引导，就能从 source_node 进入 target_node，从而完成一次路径引导。推荐信息的选择会根据异常等级、测试次数和权重的大小，根据计算得到的数字对 TOPN 的信息进行推荐。

表 3.13: 推荐信息表 (edge)

字段	类型	默认值	说明
id	bigint	0	推荐信息 id, 主键
message	varchar	NULL	异常描述
type_exception	varchar	NULL	异常等级
source_node	varchar	NULL	出发点
target_node	varchar	NULL	目标点
image_url	varchar	NULL	截图访问地址
event_handler	varchar	NULL	事件类型
event_type	varchar	NULL	触发方式
app_key	varchar	NULL	测试应用
data_type	int	0	信息类型
number	int	0	测试次数
weight	int	0	权重
create_time	datetime	NULL	创建时间

表3.14是节点信息表 (node), 用来存储待测应用中界面节点，主要字段有节点名称、应用名称和类型。通过使用表中的数据可以用来构建窗口转换图，再根据众包工人提交的信息，计算窗口覆盖率。

表 3.14: 节点信息表 (node)

字段	类型	默认值	说明
id	int	0	窗口 id, 主键
window	varchar	NULL	节点名称
app_key	varchar	NULL	应用名称
type	varchar	NULL	类型

表3.15是测试结果表 (app_bug_info)，用来存储众包工人的测试结果，主要字段有推荐信息 id、结果描述、创建时间、当前节点信息、优先级、截图地址、状态、类型、用户 id、标志位和应用版本 id。其中 edge_id 用来记录推荐信息 id, 通过该字段和推荐信息表联系起来，是该表的外键。bug_decribe、priority 和 type 是众包工人对提交的异常信息的量化描述，info_flag 是标记位，用来标记该条信息是对异常信息的验证还是众包工人提交的新异常信息。

表 3.15: 测试结果表 (app_bug_info)

字段	类型	默认值	说明
bug_id	int	0	测试结果 id, 主键
edge_id	varchar	NULL	推荐信息表 id
bug_describe	varchar	NULL	结果描述
c_time	datetime	NULL	创建时间
current	varchar	NULL	当前节点信息
priority	varchar	NULL	优先级
screenshot_adr	varchar	NULL	截图地址
status	varchar	NULL	状态
type	varchar	NULL	类型
user_id	int	0	用户 id
info_flag	int	0	标志位
app_version_id	datetime	NULL	应用版本 id

表3.16是应用信息表 (app_version_info)，用来存储 Android APP 的信息和进行自动化测试的手机信息，主要字段有应用名称、应用版本、设备品牌和系统版本。

表3.17是设备信息表 (bug_device_info)，用来存储众包工人的设备信息。主要字段有异常信息 id、用户 id、可用内存、设备品牌、网络类型、网络提供商、分辨率、系统语言、系统版本、系统模式和总内存等信息。bug_id 是在众包测试过程中出现系统崩溃时，系统会对异常信息进行统计，avail_memory 则指向对应信息。收集可用内存、网络类型、分辨率等信息是便于后期分析 Android APP 出现系统崩溃的原因。

表 3.16: 应用信息表 (app_version_info)

字段	类型	默认值	说明
id	int	0	应用 id, 主键
app_version	varchar	NULL	应用名称
app_version	varchar	NULL	应用版本
device_brand	varchar	NULL	设备品牌
device_version	varchar	NULL	系统版本

表 3.17: 设备信息表 (bug_device_info)

字段	类型	默认值	说明
id	int	0	设备 id, 主键
bug_id	varchar	NULL	异常信息 id
user_id	varchar	NULL	用户 id
avail_memory	varchar	NULL	可用内存
device_brand	varchar	NULL	设备品牌
network_type	varchar	NULL	网络类型
providers_name	varchar	NULL	网络提供商
resolution	varchar	NULL	分辨率
system_language	varchar	NULL	系统语言
system_version	varchar	NULL	系统版本
system_model	varchar	NULL	系统模式
total_memory	varchar	NULL	总内存

3.5　本章小结

本章主要工作是系统需求分析与概要设计。首先对系统进行总体概述，以系统流程图展示系统运行的总体过程，然后从涉众、功能需求和非功能需求三个方面对系统需求展开分析，并以图表的方式对系统用例进行描述。其次，通过系统架构设计图和逻辑视图，开发视图、进程视图、物理部署视图、场景视图等不同的角度，对系统进行深入分析。最后，通过实体关系图展示数据库表之间的联系，并分别对各数据库表中的字段进行了详细的介绍。

第四章　系统详细设计与实现

4.1　自动化测试数据处理模块设计

4.1.1　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依赖于慕测平台自动化测试工具对 Android APP 进行自动化测试的结果。该模块主要负责对自动化测试结果进行模式匹配，提取其中的异常信息，同时保存自动化测试过程中产生的路径信息，用于引导众包工人复现异常。对于自动化测试过程中产生的截图，该模块将根据点位信息对其进行标注，便于精确引导。

4.1.2　自动化测试数据处理模块类图

图4.1为自动化测试数据处理模块类图。自动化测试结果以 JSON 数据形式存储在文件中，ParseJson 类用于对其进行解析，在遍历过程中依据规则匹配出异常信息和测试过程信息。Edge 类用于控制数据提取过程和数据存储，调用 DbDao 中的方法类将数据存储到数据库中。ScreenshotHandler 类用于处理自动化测试过程中产生的图片，根据测试结果中包含的点位数据，在指定位置进行标注。DbDao 和 Db 类用于连接数据库，进行数据的读取和写入。

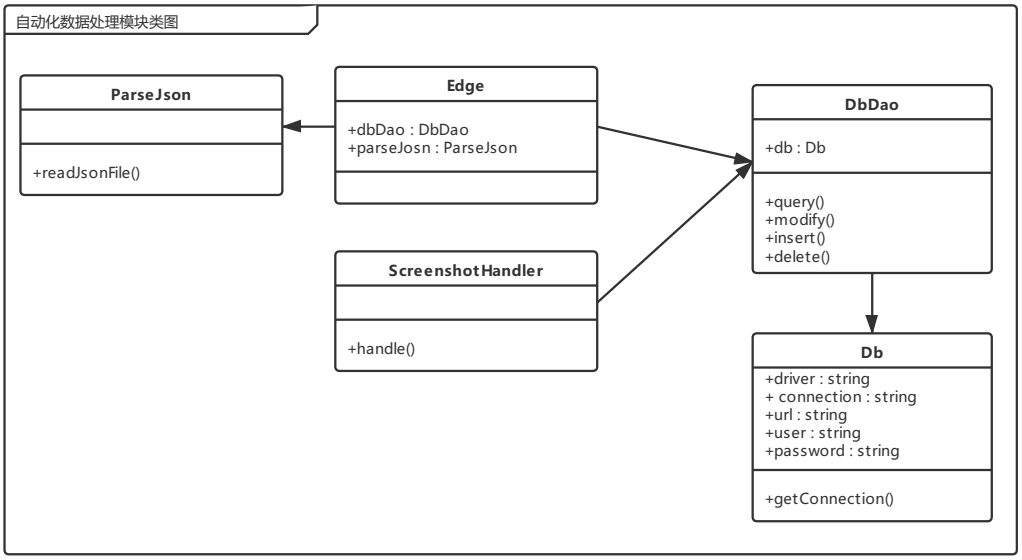

图 4.1: 自动化测试处理模块类图

4.1.3　自动化测试数据处理模块顺序图

图4.2为自动化测试数据处理模块顺序图。测试人员需要先读取对应的自动化测试文件，通过 ParseJson 类解析成对应的 BugInfo 数据，将数据存储进数据库中。另外，测试人员需要执行 ScreenshotHandler 类的 handle 方法，该方法先从数据库中读取点位信息，再将截图通过点位信息进行标注。

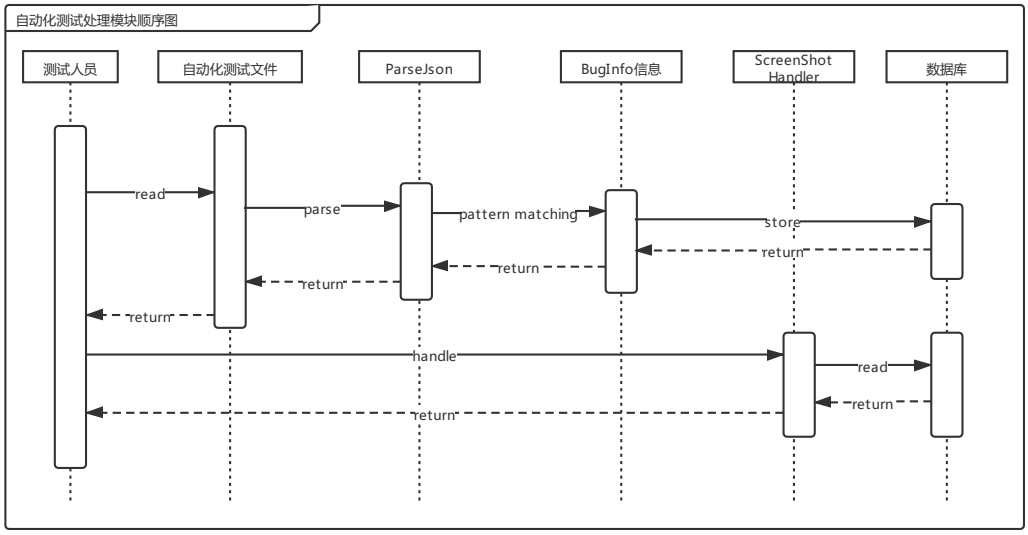

图 4.2: 自动化测试处理模块顺序图

4.1.4　自动化测试数据处理模块关键代码

图4.3是自动化测试结果处理模块中用于读取 json 文件的代码。该方法的功能是通过文件名获取对应文件，以 utf-8 的形式对数据进行读取，之后拼接到字符串中，将该字符串返回。之后会通过 JsonArray 类的 fromObject 方法将数据存储在 JsonArray 对象中，用于后续遍历。

图4.4是自动化测试结果处理模块中用于提取数据的代码。通过上面的代码，我们已经获得 jsonArray 对象，该部分代码主要展示我们需要从中提取的数据字段，包括 taskId、deviceId、bugCategory 等描述测试过程信息以及 pid、type、content 等描述异常的信息。

图4.5是自动化测试结果处理模块中用于标记图片的代码。该功能依赖 PIL.Image 包，通过其读取图片文件，并通过 Draw 方法转成画布对象。代码中 abcd 为需要标注图形的 4 个角的位置，之后指定颜色便可以完成图片标注，最后将图片的名称信息存储到数据库。

```
//读取json文件
public static String readJsonFile(String fileName) {
  String jsonStr = "";
  try {
    File jsonFile = new File(fileName);
    FileReader fileReader = new FileReader(jsonFile);
    Reader reader = new InputStreamReader(new FileInputStream(jsonFile), "utf-8");
    int ch = 0;
    StringBuffer sb = new StringBuffer();
    while ((ch = reader.read()) != -1) {
    sb.append((char) ch);
  }
    fileReader.close();
    reader.close();
    jsonStr = sb.toString();
    return jsonStr;
  } catch (IOException e) {
    e.printStackTrace();
    return null;
  }
}
```

图 4.3: 自动化测试结果处理模块-提取数据

```
 for (int i = 0; i < jsonArray.size(); i++) {
   //遍历提取数据,下面为具体内容
  JSONObject jsObject = jsonArray.getJSONObject(i);
  String taskId = jsObject.getString("taskId");
  String deviceId = jsObject.getString("deviceId");
  String bugCategoryId = jsObject.getString("bugCategoryId");
  String appLabel = jsObject.getString("appLabel");
  String packageName = jsObject.getString("packageName");
  String preActivity = jsObject.getString("preActivity");
  String currentActivity = jsObject.getString("currentActivity");
  String nextActivity = jsObject.getString("nextActivity");
  JSONObject bugInfo = jsObject.getJSONObject("bugInfo") ;
  String pid = bugInfo.getString("pid");
  String type = bugInfo.getString("type");
  if(!type.equals("E")&&!type.equals("W"))
  {
   continue;
  }
  String component = bugInfo.getString("component");
  String content = bugInfo.getString("content");
  String time = bugInfo.getString("time");
  String screenShot = bugInfo.getString("screenShot");
}
```

图 4.4: 自动化测试结果处理模块-读取 json 文件

```
// 打开图片文件路径
im = PIL.Image.open("D:\\"+apk+"\\"+picpath)
draw = PIL.ImageDraw.Draw(im)
// 进行标记
draw.line([(a, b), (c, b), (c, d), (a, d), (a, b)], width=14, fill='blue')
// 保存标记图片
im.save("D:\\"+apk+"\\" + z+picpath)
 // 将图片名称存入数据库
cursor.execute("update edge set image_url=%s
where id=%s","http://172.19.240.240:8080/"+apk+"/pic/"+z+picpath,str(id))
```

图 4.5: 自动化测试结果处理模块-标记图片

4.2　Android 静态分析模块的详细设计与实现

4.2.1　Android 静态分析模块设计

由于 Android 系统版本多样性和手机厂商闭源再开发，使得碎片化问题愈发突出，而自动化测试工具不能保证 Android APP 的所有界面都得到验证，即自动化测试结果具有不全面性。为了补全自动化测试未覆盖的页面，该系统借助静态分析工具 Gator 对 APP 进行反编译，通过对源码进行过滤查找所有窗口跳转信息，构建窗口转换图，并通过与自动化测试结果进行比对，找到所有未覆盖界面信息。

4.2.2　Android 静态分析模块类图

图4.6为 Android 静态分析模块类图。WTGClient 类是对 Android APP 进行静态分析的类，通过其找出所有窗口跳转信息并将结果输出保存在文件中。Node 类的功能是根据上述结果，将所有窗口信息以节点的形式保存到数据库。Edge 类的功能则是比对自动化测试结果与静态分析结果，将未覆盖数据保存到数据库中。DbDao 类负责对数据库进行写入，Db 类负责数据库连接。

4.2.3　Android 静态分析模块顺序图

图4.7是 Android 静态分析模块顺序图。测试人员需要提交 Android APP，之后通过命令行工具启动 Gator 对已提交 APP 进行静态分析，将分析结果导出保存在文件中。然后，测试人员通过 Node 类和 Edge 类对测试结果进行分析，Node 类用于提取所有窗口，Edge 类用于提取自动化测试未覆盖信息，最后将提取到的数据保存到数据库中。

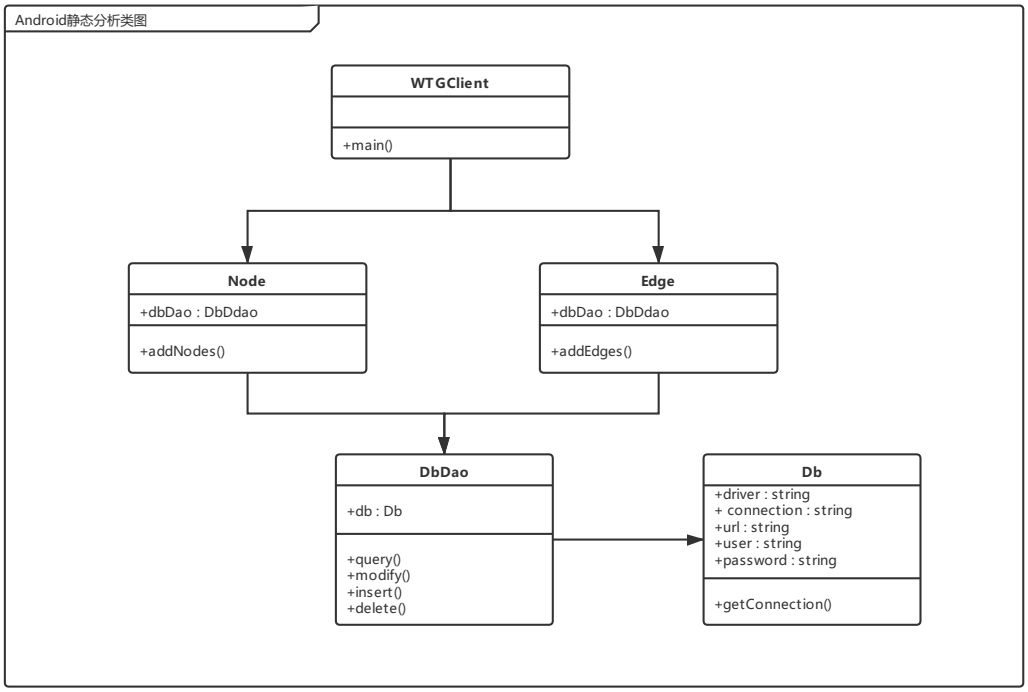

图 4.6: Android 静态分析模块类图

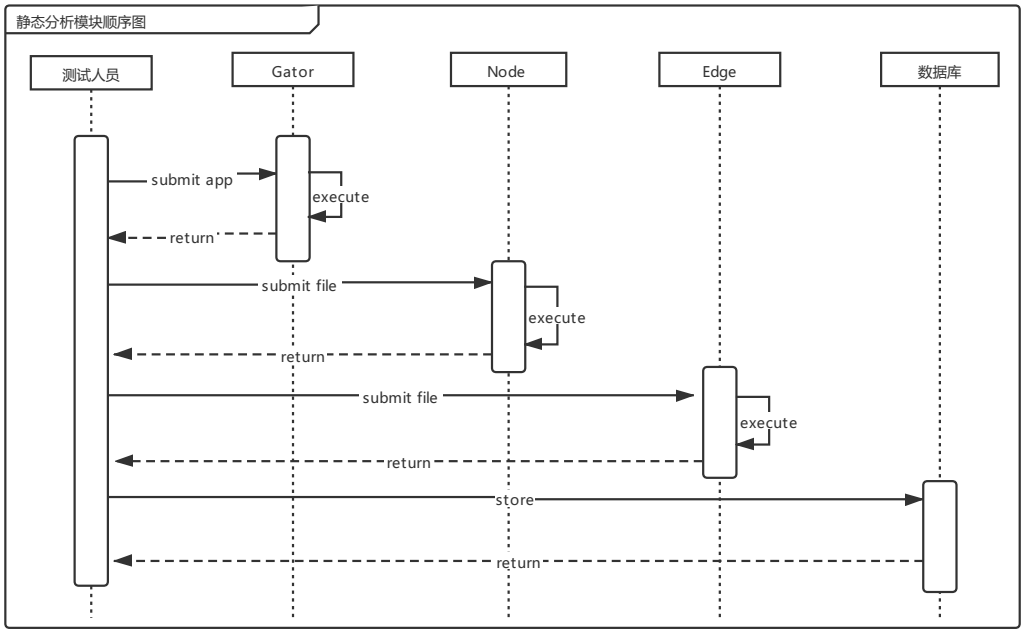

图 4.7: Android 静态分析模块顺序图

4.2.4　Android 静态分析模块模块关键代码

图4.8是 Android 静态分析模块中用于获取窗口信息的代码。该代码的主要流程是先读取静态分析结果信息，通过循环读取静态分析结果，对字符串信息进行分割，提取 activity、type 等信息，最后存储进数据库。

图4.9是 Android 静态分析模块中用于获取未覆盖路径的代码。在此代码之前，已经通过对字符串切割获得 sourceNode 节点和 targetNode 节点。该代码的功能是通过 SQL 查询数据库中是否已存在对应的 sourceNode 和 targetNode，若存在则不进行处理，若不存在，则说明该路径为自动化测试中未覆盖的路径，则在将其标记为未覆盖后存入数据库。

```
// 遍历提取窗口信息
while((line = bufferedReader.readLine())!=null){
// 如果是起点信息
if(line.startsWith("Current Node")){
   String[] s1 = line.split(":");
   String[] s2 = s1[1].split("\\[");
   if(s2[0].equals("LAUNCHER_NODE"))
   {
    activity = "Launcher";
    type = "LAUNCHER";
    continue;
 }else{
    String[] s3 = s2[1].split("\\]");
    String[] s4 = s3[0].split("\\.");
    if(s3[0].endsWith("Activity")){
     type = s2[0];
     activity = s4[s4.length-1];
    }else{
     continue;
    }
  }
}
```

图 4.8: Android 静态分析模块-窗口获取

4.3　Android 引导模块的详细设计与实现

4.3.1　Android 引导模块设计

Android 引导模块是通过在 Android APP 上加入插件代码的方式来进行嵌入的，因此使用该模块的前提是需要提供待测 Android APP 的源码。该模块以悬浮窗按钮的形式提供给众包工人，在点击按钮后会展示该插件的具体功能如用户登录、异常界面推荐、未覆盖界面推荐、路径引导、测试结果上报等功能，在方便其使用的同时减少了插件给测试过程中带来的影响。

异常界面推荐和未覆盖界面推荐均是以推荐列表的形式展示，众包工人选择其中一个用例进行测试。路径引导是以截图加信息描述的形式展示给众包工

```
// 判断是否是未覆盖路径
String sql = "select * from edge where source_node=? and target_node=?";
ResultSet resultSet = null;
resultSet = db.query(sql, source_node,target_node);
if(!resultSet.next())
 {
     String sql2 = "insert into edge(message,type_exception,source_node,target_node,image_url,"
       + "event_handlers,event_type,app_key,data_type,number,weight,create_time)"
       + "values(?,?,?,?,?,?,?,?,?,?,?,?)";
     int result = db.insert(sql2, "",0,source_node,target_node,"",event_type,"",
     Edge.apk,0,0,1,"2019-10-10 17:20:05");
     if(result==1)
     {
      System.out.println("ok");
     }
 }
```

图 4.9: Android 静态分析模块-未覆盖路径获取

人，引导其点击进入下一界面。测试结果上报会截图并进行弹窗，要求众包工人进行异常信息描述填写。此外，考虑众包测试过程中可能会发生系统崩溃，导致异常结束测试，而众包工人无法提交崩溃异常，该模块通过自定义错误处理在发生崩溃时主动向服务端推送崩溃信息。

4.3.2　Android 引导模块类图

图4.10是 Android 引导模块类图。BugHunterActivity 类是启动类，负责模块的初始化工作，如设置截图监听、开启悬浮窗、自定义崩溃信息上报等功能。ScreenShotListenManager 是监听类，负责启动监听以及销毁监听，这里主要监听截图事件。Camera 类是用于截取屏幕图片的类。CrashHandler 类是自定义异常类，系统崩溃后则会自动执行类中的向服务端发送信息的方法。SystemUtil 类是用于获取系统信息，包括系统版本、手机型号、网络状态等。FloatingActionButton 类是模块功能的主要实现类，该类通过 OkHttpRequest 类来发送请求，并接受服务端响应，进行信息展示。

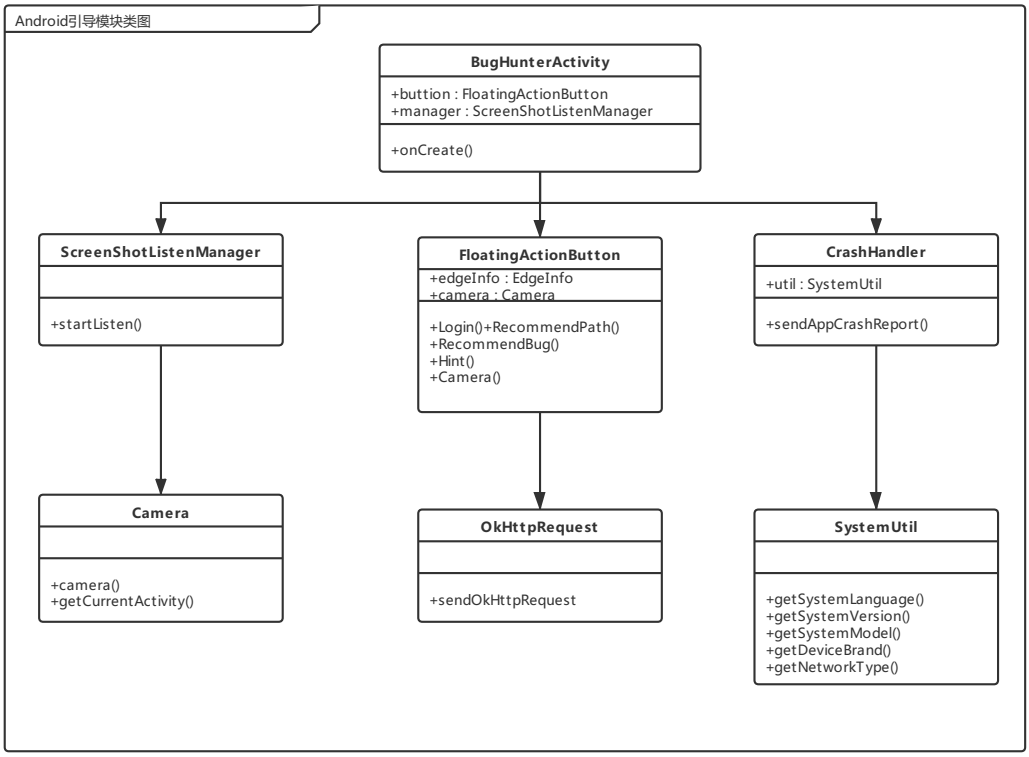

图 4.10: Android 引导模块类图

4.3.3　Android 引导模块顺序图

图4.11是 Android 引导模块顺序图。Android APP 运行时，首先通过 BugHunterActivity 类执行初始化方法，通过 ScreenShotListenManager 类的 startListen 方法进行截图监听注册，当 Camera 类的 camera 方法被调用时，则执行响应。CrashHandler 类是自定义的崩溃处理类，当系统出现异常时，handlerException 方法将会执行，在通过 SystemUtil 类获取系统信息后，将系统信息以及崩溃信息发送至服务端。众包工人通过操作 FloatingActionButton 类中的方法来使用插件引导自己进行测试任务，类中方法会通过 OkHttpRequest 类向服务端发送请求信息，并获取响应，将响应的结果展示给众包工人。

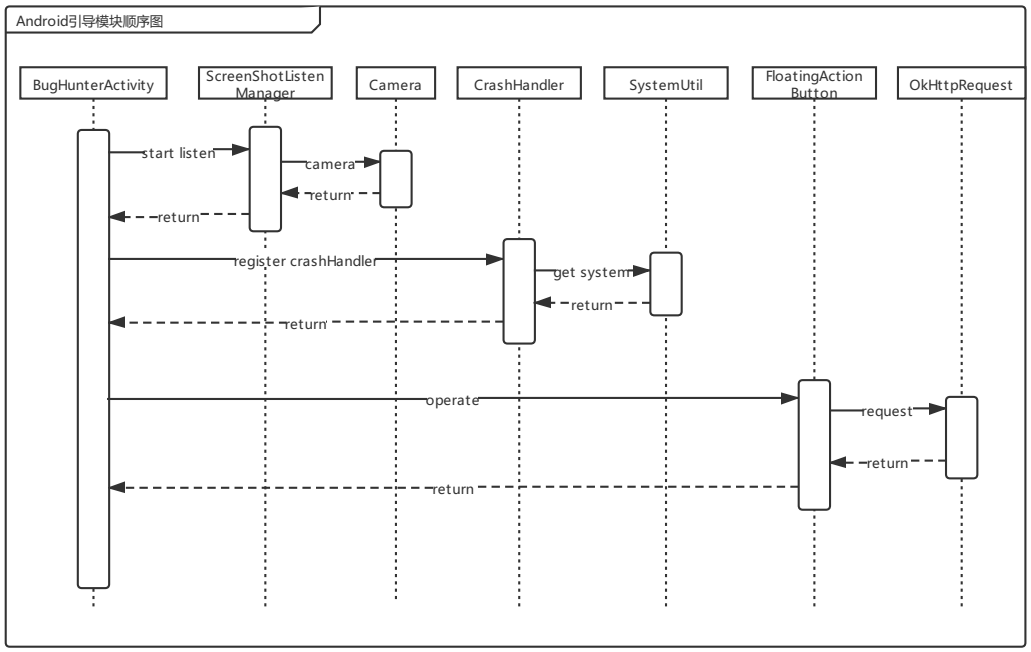

图 4.11: Android 引导模块顺序图

4.3.4　Android 引导模块关键代码

图4.12是 Android 引导模块中用于截图监听的代码。这部分代码通过 ScreenShotListenManager 类的 newInstance 方法构造监听对象，当截屏事件发生时则从缓存中获取当前屏幕的照片。

图4.13是 Android 引导模块中用于自定义崩溃处理的代码。当测试过程中程序发生崩溃时，则自动执行该代码，通过其获取安装包的信息，包括安装包版本、安装包 API 等级以及异常信息，作为 Android APP 崩溃异常报告。该报告信息会与测试手机系统信息一同作为报告数据返回给服务端。

图4.14是 Android 引导模块中用于发送请求并获取响应的代码。参数 json 是异常描述信息，address 是请求地址，file 是需要发送的文件，callback 则是返回信息。OkHttpClient 是发送请求的对象，在用异常描述信息和文件构造好结构体之后便可以通过 newCall 方法进行发送，响应结果则从 callback 参数中进行获取。

```
FloatingActionButton button=new FloatingActionButton(Camera.getCurrentActivity());
ScreenShotListenManager manager =
 ScreenShotListenManager. newInstance(Camera.getCurrentActivity());
//设置截屏监听
manager.setListener(
new ScreenShotListenManager.OnScreenShotListener() {
public void onShot(String imagePath) {
// do something
Camera camera=new Camera();
camera.camera(Camera.getCurrentActivity());
  }
 }
);
manager.startListen();
```

图 4.12: Android 引导模块-截图监听

```
private String getCrashReport(Context context, Throwable ex) {
PackageInfo pinfo = getPackageInfo(context);
StringBuffer exceptionStr = new StringBuffer();
// 获取系统版本
exceptionStr.append("Version:" + pinfo.versionName + "(" + pinfo.versionCode + ")");
// 获取API版本
exceptionStr.append("Android:" + android.os.Build.VERSION.RELEASE+ "(" +
android.os.Build.MODEL + ")");
// 获取异常信息
exceptionStr.append("Exception:" + ex.getMessage());
StackTraceElement[] elements = ex.getStackTrace();
for (int i = 0; i < elements.length; i++) {
exceptionStr.append(elements[i].toString());
 }
return exceptionStr.toString();
}
```

图 4.13: Android 引导模块-崩溃处理

4.4　推荐模块的详细设计与实现

4.4.1　推荐模块设计

在传统的众包测试中，众包工人只能根据自己已有的知识去测试 Android APP 中可能存在的 BUG 并进行提交，因为其专业能力有限，所以测试覆盖率低且测试结果具有极高的重复性，严重影响测试效率。推荐模块是为众包工人在众包测试过程中，加快其测试的速度和效率，在综合分析自动化测试结果处理模块和静态分析结果处理模块的数据，对其进行异常界面和未覆盖界面的推荐。在整个测试过程中，众包工人只需要验证异常是否属实，同时在少量未覆盖的

```
public static void sendOkPostPicData(String json, String address, File file, okhttp3.Callback
callback){
   //使用OkHttpClient对象
   OkHttpClient client=new OkHttpClient();
   RequestBody fileBody = RequestBody.create(MediaType.parse("image/png"), file);
   // 构造请求体
   RequestBody requestBody = new MultipartBody.Builder()
   .setType(MultipartBody.FORM)
   .addFormDataPart("screenshot", "head_image", fileBody)
   .addFormDataPart("bug", json)
   .build();
   Request request=new Request.Builder()
   .url(address)
   .addHeader("Accept","application/json")
   .post(requestBody)
   .build();
   // 发送请求并获取返回结果
   client.newCall(request).enqueue(callback);
}
```

图 4.14: Android 引导模块-请求发送

页面中进行测试就可以完成测试任务，可以在减小众包工人专业素养差异化的同时，大大提高了测试效率和测试覆盖率。

首先，对于异常界面推荐，会根据异常的严重程度、异常被测试的次数并结合众包工人以往的测试结果，对 TopN 的异常界面进行推荐。当一个异常被测试的次数超过规定次数时，就停止对其进行推荐，这样在一定程度上减少了重复测试。对于未覆盖界面，会根据当前众包工人在 Android APP 中所处的位置，进行单步骤推荐，推荐的方式会以截图加异常描述的方式返回给众包工人。

4.4.2　推荐模块类图

图4.15为推荐模块类图。FloatingActionButton 为 Android 端引导模块的类，众包工人通过其中的 RecommendPath 来请求推荐。EdgeController 类是控制类，用来接收用户的请求，并调用对应的处理类 EdgeService 进行处理，将处理结果即推荐信息作为响应返回给用户。EdgeService 是接口类，只有 getRecommendBugs 方法声明，业务逻辑是在 EdgeServiceImple 中实现的，是真正的逻辑层处理类。EdgeDao 是连接数据库并从数据库中读取数据的类，Edge 是实体类，数据库中读取的数据都封装在 Edge 类中。EdgeVO 类是对 Edge 类进行封装，响应结果也是以 EdgeVO 类的形式返回。Edge2UserDao 是用来剔除众包工人已经测试过的类，避免相同的推荐信息被重复测试。

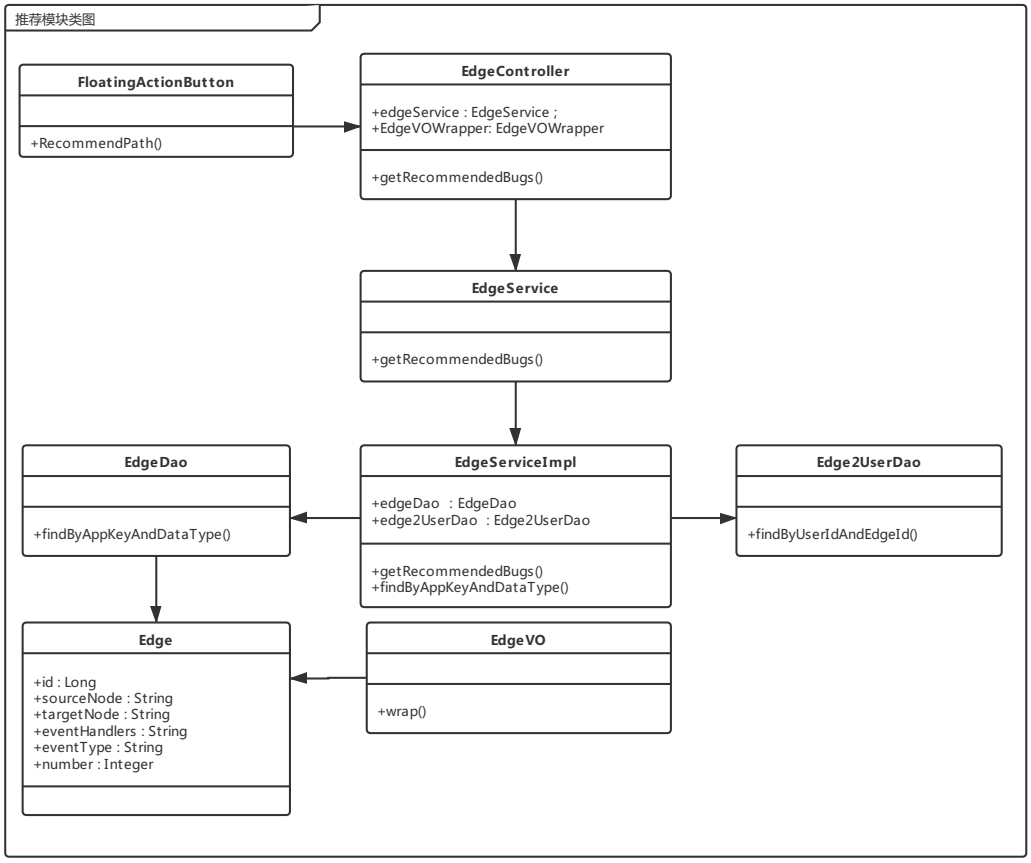

图 4.15: 推荐模块类图

4.4.3　推荐模块顺序图

图4.16为推荐模块顺序图。当 FloatingActionButton 类中的 RecommendPath 方法被触发执行的时候，请求到达 EdgeController 类中，EdgeController 类会执行获取推荐任务的 getRecommendBugs 方法，该方法中包含的 EdgeService 类会调用 findByAppKeyAndDataType 方法去获得当前需要推荐的信息。然后 EdgeController 类再通过 Edge2User 类获取当前用户已经测试的信息，并从将要推荐的信息中剔除已测试过的信息，再通过 EdgeVO 类中的 wrapper 方法进行封装，将最终结果返回到 FloatingActionButton 类中。

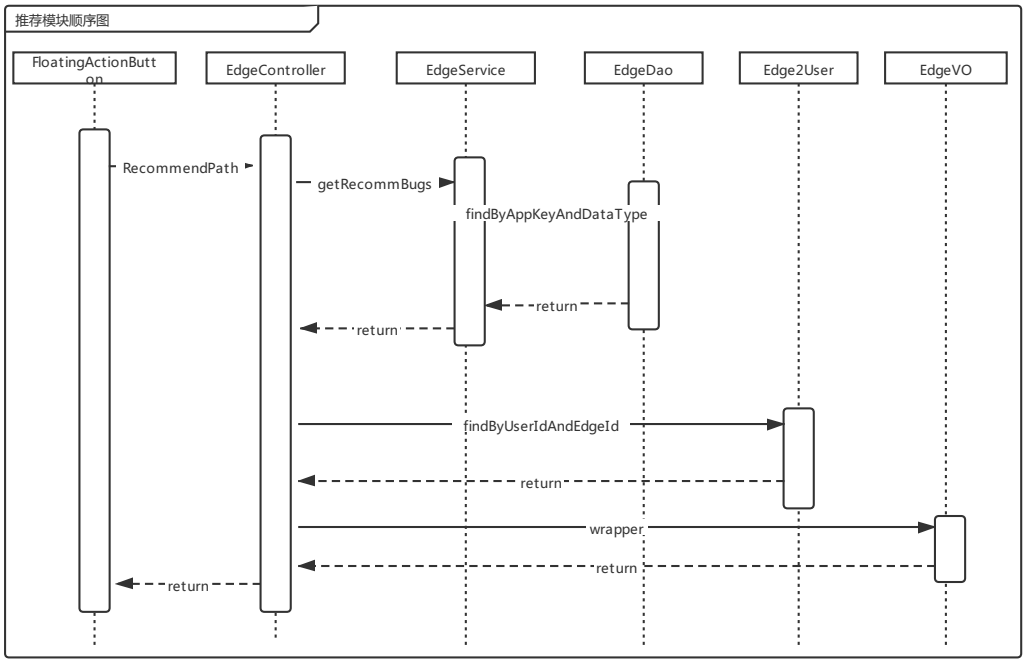

图 4.16: 推荐模块顺序图

4.4.4　推荐模块关键代码

图4.17是 EdgeController 类中用来处理推荐请求的代码，包括了请求路径和请求方式。其中参数 currentWindow 是用户当前所处位置，参数 isCovered 是用户请求推荐类型的标识，该方法的主要功能是调用 getRecommendBugs 方法来完成推荐任务。

图4.18是处理推荐请求的业务代码。当该方法被调用时，首先执行 findByAppKeyAndDataType 方法，该方法会根据异常的严重程度、异常被测试的次数计算得到所有的推荐信息，并按计算结果从大到小进行排序，然后判断当前请求是异常界面还是未覆盖界面。若是未覆盖界面，则遍历之前得到的推荐信息，并通过 findByUserIdAndEdgeId 方法去除已测试信息，然后将所有未覆盖界面通过 EdgeVO 类封装后返回。若是异常界面，则通过遍历，将所有未测试且路径不重复信息保存到列表中，且列表仅保留前五条信息。

```
@RequestMapping(value = "/{appKey}/{currentWindow}/bugList/{isCovered}/{userId}",
method = RequestMethod.GET)
public
@ResponseBody
ResultMessage getRecommendedBugs(HttpServletRequest request, @PathVariable
String appKey, @PathVariable String currentWindow, @PathVariable Integer
isCovered, @PathVariable Integer userId) {
  String[] infos = currentWindow.split("\\.");
  currentWindow = infos[infos.length - 1];
  List<EdgeVO> messages = edgeService.getRecommBugs(appKey, currentWindow,
  isCovered, userId);
  return ResultMessageFactory.getResultMessage(messages);
}
```

图 4.17: 推荐模块-推荐请求处理

```
public List<EdgeVO> getRecommBugs(String appKey, String currentWindow, Integer isCovered,
Integer userId) {
    List<Edge> edges = edgeDao.findByAppKeyAndDataType(appKey, isCovered);
    List<EdgeVO> edgeVOs = new ArrayList<>();
    // 推荐未覆盖路径
    if (isCovered == 0) {
        int i = 0;
        for (Edge e : edges) {
            // 已测试过的信息
            Edge2User byUserIdAndEdgeId = edge2UserDao.findByUserIdAndEdgeId(userId,
            e.getId());
            if (byUserIdAndEdgeId == null) {
                // 对推荐信息进行封装
                 ...
      } } }
    } else {
        // 推荐异常路径
        int i = 0;
        Set<List<String>> set = new HashSet<>();
        for (Edge e : edges) {
            Edge2User byUserIdAndEdgeId = edge2UserDao.findByUserIdAndEdgeId(userId,
            e.getId());
            if (byUserIdAndEdgeId == null) {
                List<String> list = new ArrayList();
                list.add(e.getTargetNode());
                list.add(e.getSourceNode());
                if (!set.contains(list)) {
                  // 对推荐信息进行封装
                  ...
                  // 推荐top5的信息
                  if(i>5) {break;}
    } } } }
      return edgeVOs;
}
```

图 4.18: 推荐模块-推荐处理

4.5　路径引导模块的详细设计与实现

4.5.1　路径引导模块设计

在 Android APP 比较小，窗口数量比较少，功能设计比较简单的情况下，依靠异常界面推荐和未覆盖界面能在一定程度上提高测试效率，这在下面的实验数据中会说明。但是当 Android APP 比较复杂，窗口数量又比较多的场景下，众包工人很难找到推荐的异常界面或者找到推荐的异常界面需要耗费更多的时间，这种情况下就更需要引导模块，来帮助众包工人快速定位到异常，便于进行众包测试。

该引导模块能够根据众包工人当前所处的窗口位置，计算到达目标界面的最短路径，以截图标记加信息描述的方式给众包工人提示下一步骤的具体操作，引导其触发异常并进行验证。

4.5.2　路径引导模块类图

图4.19为路径引导模块类图。FloatingActionButton 类是 Android 端引导模块的类，它包含请求路径引导的方法 Hint。PathController 是服务端控制类，用于接收客户端发来的请求，并调用 PathService 接口中的 getNextBugHint 方法来处理请求。PathServiceImpl 是接口的具体实现类，通过其进行业务逻辑处理。通过 EdgeDao 类来读取数据库中的数据，数据封装在 Edge 类中。在找到引导路径时，通过 EdgeVO 对数据进行再次封装，将结果返回。

4.5.3　路径引导模块顺序图

图4.20为路径引导模块顺序图。众包工人通过 FloatingActionButton 类中的 Hint 方法来请求路径引导，PathController 类能接收对应的请求，并调用执行 PathService 类中的 getNextHint 方法，该方法从数据库中读取正常路径，以逆路径推导的形式，计算最短路径后返回引导窗口信息。PathController 类通过 EdgeVO 类的方法进行封装后作为响应结果返回给客户端。

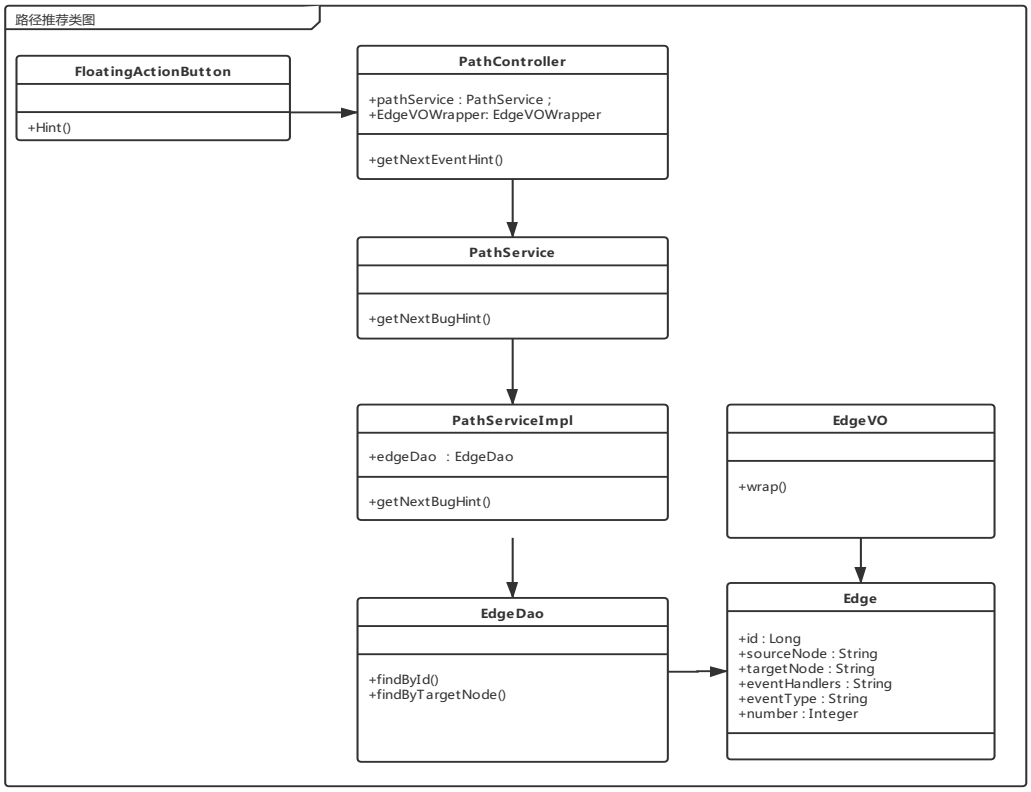

图 4.19: 路径引导模块类图

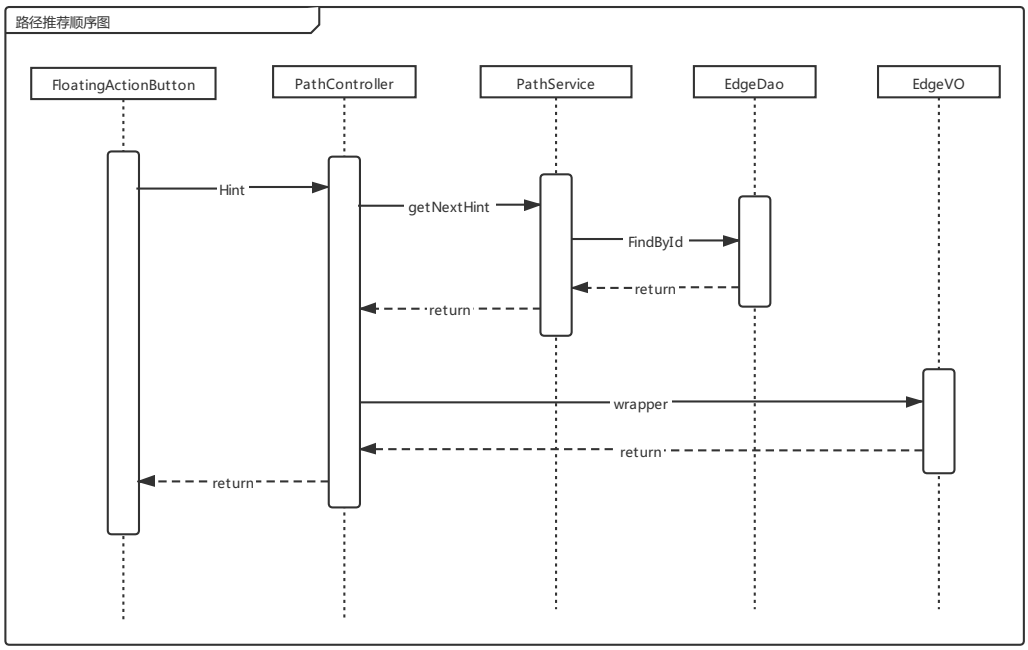

图 4.20: 路径引导模块顺序图

4.5.4　路径引导模块关键代码

图4.21是 PathController 类中用来处理路径引导的代码，请求方式为 Get 请求。参数 currentWindow 是用户当前所处的窗口位置，是计算最短路径的起点，参数 nextWindow 是需要到达的窗口位置，是计算最短路径的终点，参数 edgeId 是待测页面的 ID, 根据其找到对应的待测信息。这部分的主要功能是调用 getNextHint 方法来完成路径引导工作。

图4.22是 getNextBugHint 接口的具体实现方法。首先是通过 edgeId 找到待测信息。接下来进行判断，如果用户当前位置和推荐位置相同，则直接返回待测信息，若不同，则以 nextWindow 为起点，以逆路径的形式遍历查找所有能到达该节点的节点，若这些节点中包含 currentWindow，则把 currentWindow 到达 nextWindow 的下一个节点作为结果返回。

```
@RequestMapping(value = "/path/nextHint/{currentWindow}/{nextWindow}/{edgeId}",
method = RequestMethod.GET)
public
@ResponseBody
ResultMessage getNextEventHint(@PathVariable String currentWindow, @PathVariable
String nextWindow) {
    String[] infos = currentWindow.split("\\.");
    currentWindow = infos[infos.length - 1];
    return ResultMessageFactory.getResultMessage(edgeService.getNextHint(currentWindow,
nextWindow));
}
```

图 4.21: 路径引导模块-路径引导请求处理

4.6　测试结果上报模块的详细设计与实现

4.6.1　测试结果上报模块设计

对于众包工人来说，需要上报的数据主要分为两类, 包括在异常界面验证异常的数据和在未覆盖界面发现异常并进行新异常上报的数据。对于第一类，该模块会在异常引导的最后一步触发结果上报接口，众包工人需要根据提示上传测试结果。对于第二类，该模块以按钮的形式进行触发，系统会自动将众包工人当前所处的窗口进行截图，并引导众包工人填写对应的异常描述。

4.6.2　测试结果上报模块类图

图5.4是测试结果上报模块的类图。FloatingActionButton 类是 Android 端引导模块的类，包含了 camera 方法，用于截图以及收集测试报告数据。BugController 类用于接收客户端传来的数据，并调用 BugService 类的 addBugs 方法进行处理。

```
// 解析服务端传来的数据
JSONObject jsonObject = new JSONObject(msg.obj.toString());
JSONObject data = jsonObject.getJSONObject("data");
String id = data.getString("id");
String sourceNode = data.getString("sourceNode");
String targetNode = data.getString("targetNode");
// 当前是目标路径
if(edgeInfo.getId().equals(id)){step = 2;}
// 获取响应信息
String eventHandlers = data.getString("eventHandlers");
String message = data.getString("message");
String path = data.getString("path");
String imageUrl = data.getString("imageUrl");
String dateType = data.getString("dataType");
// 构造EdgeInfo对象
EdgeInfo new_edgeInfo = new EdgeInfo(id, sourceNode, targetNode, eventHandlers,
message, path, imageUrl);
View bottemView = LayoutInflater.from(context).inflate(R.layout.dialoglistview, null);
TextView textView = (TextView) bottemView.findViewById(R.id.dialoglist_id);
textView.setText("Transition Path:");
final ListView listView = (ListView) bottemView.findViewById(R.id.listView);
List<EdgeInfo> edgeInfos = new ArrayList<>();
edgeInfos.add(new_edgeInfo);
listView.setChoiceMode(AbsListView.CHOICE_MODE_SINGLE);
// 进行路径推荐展示
final EdgeInfoAdapter edgeInfoAdapter = new EdgeInfoAdapter(context,  R.layout.edgeitem,
edgeInfos);
listView.setAdapter(edgeInfoAdapter);
```

图 4.22: 路径引导模块-路径引导处理

BugServiceImpl 是具体实现类，执行底层业务逻辑。PicName 是生成截图名称的类，便于对文件进行管理。Constants 类中主要定义一些常量数据，BugInfoDao 则用于将测试数据存储进数据库。

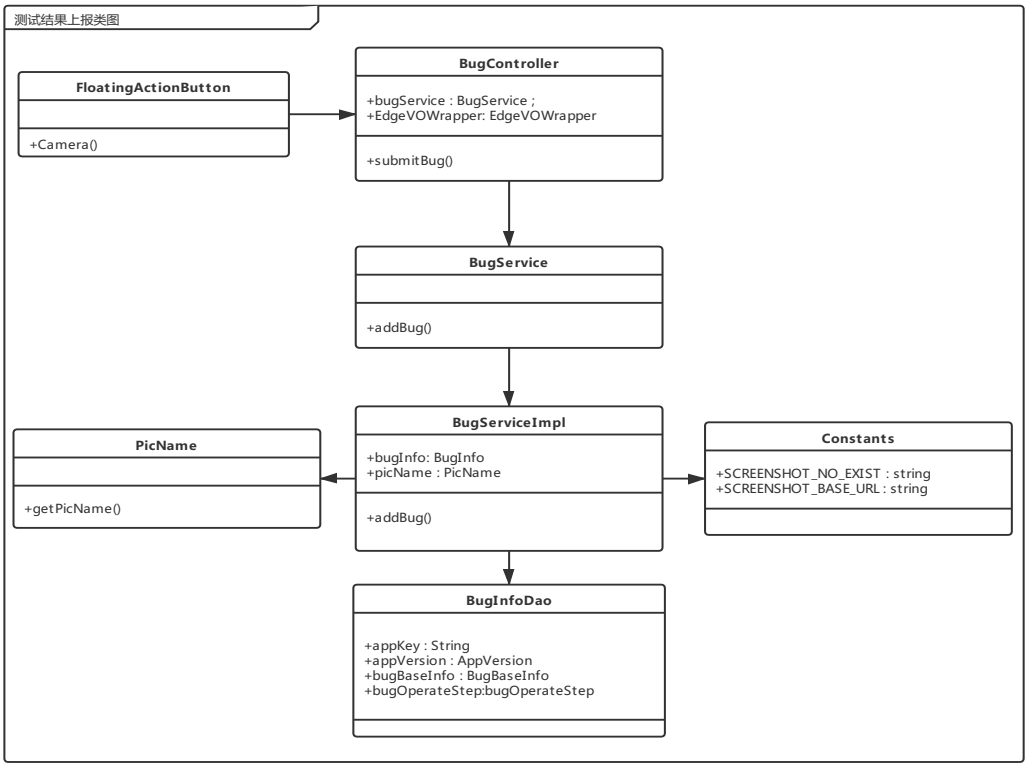

图 4.23: 测试结果上报模块类图

4.6.3　测试结果上报模块顺序图

图4.24为测试结果上报模块顺序图。用户通过 FloatingActionButton 类的 Camera 方法来请求进行数据上报，数据包括截图文件和异常描述信息。BugController 接收到数据上报请求后调用 BugService 的 submit 方法来执行业务。先通过调用 PicName 类中的 getPicName 方法获取文件名称，之后将截图数据通过 transferTo 方法存储至本地，最后通过 BugInfoDao 将数据存储进数据库。

4.6.4　测试结果上报上报模块关键代码

图4.25是测试结果上报模块中截图处理的代码。该模块主要通过 Bitmap 类和 BufferedOutputStream 类来生成截图信息。首先获取机型屏幕大小，用于设置图片规格。之后通过从缓存中来获取当前屏幕的图片，通过 Bitmap 设置好图片的类型及大小后，通过 BufferedOutputStream 类将数据写入文件。

图4.26是测试结果上报模块中结果上报处理部分代码。首先通过 getPicName 获取对应的文件名称，之后通过 transferTo 方法写入数据。之后通过 BugInfo 对象保存对应上传数据，通过 addBug 方法将对象保存到数据库中。

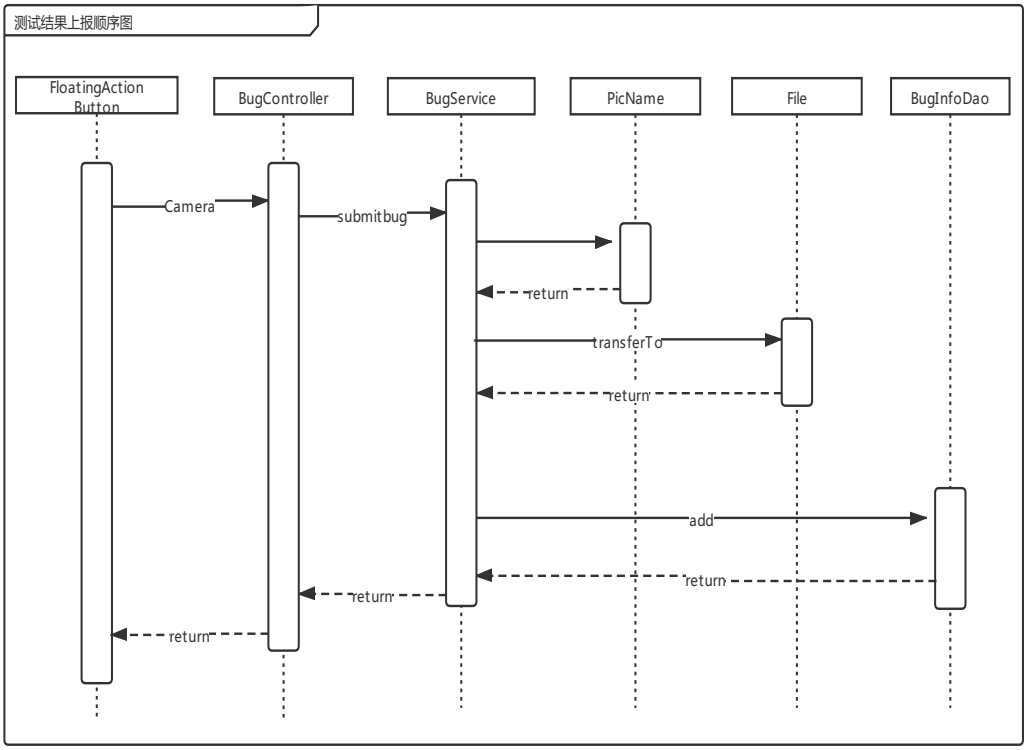

图 4.24: 测试结果上报模块顺序图

4.7　本章小结

本章主要介绍系统各个模块的详细设计与实现，包括自动化测试数据处理模块、Android 静态分析模块、Android 引导模块、推荐模块、路径引导模块和测试结果上报模块。主要从模块的类图和顺序图两个方面来对模块进行详细介绍，并通过展示关键代码对模块的主要功能逻辑进行解释说明。

```
//获取当前屏幕的大小
int width = activity.getWindow().getDecorView().getRootView().getWidth();
int height = activity.getWindow().getDecorView().getRootView().getHeight();
//生成相同大小的图片
Bitmap temBitmap = Bitmap.createBitmap(width, height,
Bitmap.Config.ARGB_8888);
//找到当前页面的跟布局
View view = activity.getWindow().getDecorView().getRootView();
//设置缓存
view.setDrawingCacheEnabled(true);
view.buildDrawingCache();
//从缓存中获取当前屏幕的图片
temBitmap = view.getDrawingCache();
String file_str = Environment.getExternalStorageDirectory().getPath();
File mars_file = new File(file_str + "/my_camera");
final File file = new File(file_str + "/my_camera/file.png");
if (Environment.MEDIA_MOUNTED.equals(Environment.getExternalStorageState())) {
    if (!mars_file.exists()) {
     mars_file.mkdirs();
   }
 }
 try {
     BufferedOutputStream bos = new BufferedOutputStream(new FileOutputStream(file));
     temBitmap.compress(Bitmap.CompressFormat.PNG, 100, bos);
     bos.flush();
     bos.close();
 } catch (IOException e) {
     e.printStackTrace();
 }
```

图 4.25: 测试结果上报模块-截图处理

```
// 存储图片
File logoSaveFile = new File(Constants.SCREENSHOT_BASE_URL);
if (!logoSaveFile.exists()) {
   logoSaveFile.mkdirs();
 }
screenshotName = picName+".png";
String screenshotFileName = Constants.SCREENSHOT_BASE_URL + File.separator +
screenshotName;
File screenshotFile = new File(screenshotFileName);
// 写入文件
file.transferTo(screenshotFile);
// 存数据库
JSONObject jsonObject = new JSONObject(jsonStr);
BugInfo bugInfo = new BugInfo(jsonObject, screenshotName);
BugInfoKeys bugInfoKey = bugService.addBug(bugInfo);
return ResultMessageFactory.getResultMessage(bugInfoKey);
```

图 4.26: 测试结果上报模块-结果上报处理

第五章　系统测试与结果分析

系统测试主要根据第三章的需求分析与第四章的详细设计与实现，对系统各模块的功能实现、系统响应性能、系统并发性能进行测试。本章首先介绍系统测试环境，然后对系统进行功能测试与非功能测试，并对系统测试结果进行分析。

5.1　测试准备

5.1.1　测试环境

表5.1展示了系统用到的工具以及测试环境配置。首先，Android 引导插件借助开发工具 Android studio 将插件嵌入到源码中，服务端开发工具使用的是 Intellij IDEA，系统框架使用 Spring Boot 进行开发，以上两种开发工具均需要 Java 开发环境。对 Android APP 进行静态分析使用的是开源工具 Gator，它和服务端一同部署在阿里云服务器。对自动化测试结果和静态分析结果进行分析使用的工具是 Eclipse，单独部署在本地环境中。根据自动化测试结果中的点位信息进行图片绘制使用的工具是 Pycharm, 它需要运行在 Python 环境下。所有代码开发均使用版本控制工具 Git。数据存储使用的是 MariaDB 数据库。

表 5.1: 测试环境配置表

工具或环境	配置或版本信息
Android studio	3.4.1
Intellij IDEA	2019.3.4
Eclipse	4.5.0
Pycharm	2017.2
Spring Boot	2.0
JDK	1.8.0_60
MariaDB	5.5.60
Gator	3.8
Git	2.29.2
操作系统	Ubuntu 4.8.2
系统内存	32G
浏览器	Chrome

5.2　功能测试

功能测试是对模块进行测试，验证其是否存在缺陷，并证明模块功能是否能够达到预期的效果 [44]。系统功能测试主要是对上述模块，包括自动化测试数据处理模块、Android 静态分析模块、Android 插件引导模块、推荐模块、路径引导模块和测试结果上报模块进行单元测试。自动化测试数据处理模块和静态分析模块均部署于本地环境，通过手动启动函数进行测试。推荐模块、路径引导模块和测试结果上报模块均依赖于 Android 插件引导模块，可以在部署服务端后，通过运行嵌入该插件的 Android APP 分别进行测试。

表5.2展示了自动化测试结果处理模块的测试过程与测试结果，其主要对表3.4描述的用例，进行关于异常界面数据和测试路径信息数据的提取以及图片标记的测试。此项测试用例覆盖了该模块所有的功能。

表 5.2: 自动化测试数据处理模块测试

ID	**TC1**
名称	自动化测试数据处理模块测试
测试功能	测试人员通过自动化测试数据处理模块提取异常界面信息以及自动化测试路径信息，并将数据存储进数据库。图片处理函数能根据数据库中的点位信息对图片进行绘制，生成引导图片。
测试步骤	1. 测试人员启动自动化数据处理函数 2. 测试人员启动图片处理函数
测试结果	1. 自动化测试数据处理模块执行成功，提取异常界面数据和自动化测试路径信息，并将数据存于数据库 2. 图片处理函数读取数据库成功，并获得点位信息 3. 根据点位信息对图片进行绘制，并将绘制生成的图片信息保存进数据库

表5.2展示了 Android 静态分析模块的测试过程与测试结果，其主要对表3.5描述的用例，进行关于静态结果分析、未覆盖窗口数据提取、截图标注等方面的测试。此项测试用例覆盖了该模块所有的功能。

表 5.3: Android 静态分析模块测试

ID	TC2
名称	Android 静态分模块测试
测试功能	测试人员通过使用 Gator，得到静态分析数据，通过界面窗口处理函数以及未覆盖界面处理函数，找出未覆盖页面信息
测试步骤	1. 测试人员使用 Gator 对 Android APP 进行静态结果分析 2. 调用界面窗口处理函数对静态结果进行分析，提取 Android APP 所有窗口信息 3. 调用未覆盖界面处理函数，通过与自动化测试处理结果对比，提取未覆盖窗口信息，存入数据库 4. 根据未覆盖窗口信息手动找出对应界面信息，截图后进行异常位置标注，将图片信息存入数据库
测试结果	1.Gator 工具执行静态分析成功，静态分析数据保存到文档中 2. 界面窗口处理函数将 3. 根据点位信息对图片进行绘制，并将绘制生成的图片信息保存进数据库

表5.4展示了 Android 引导模块的测试过程与测试结果。其主要对应表3.6描述的用例，进行 Android 引导插件显示、Android 引导插件响应以及发生系统崩溃强制退出时进行异常上报的测试。此项测试用例覆盖了该模块所有的功能。

表 5.4: Android 引导模块测试

ID	TC3
名称	Android 引导模块测试
测试功能	Android 插件引导模块在测试应用上能够正常显示，能够对用户点击做出响应，能在发生系统崩溃时提交错误信息报告
测试步骤	1. 测试人员使用装有 Android 引导模块的 Android APP 2. 测试人员点击插件按钮，观察是否弹出隐藏按钮 3. 测试人员跳转不同窗口，观察是否每个窗口均有插件按钮 4. 测试人员通过编写代码主动造成系统崩溃，查看后台是否有报告数据
测试结果	1. 点击插件按钮，弹出隐藏按钮，且每个隐藏按钮均能对用户的操作进行响应 2. 用户处于不同窗口时，插件按钮依然能够正常显示并响应 3. 当系统出现崩溃强制退出时，服务端能够收到该异常报告

表5.5展示了推荐模块的测试过程与测试结果。其主要对表3.8和表3.9，进行关于异常界面推荐和未覆盖界面推荐的测试。此项测试用例覆盖了该模块所有的功能。

表 5.5: 推荐模块测试

ID	**TC4**
名称	推荐模块测试
测试功能	Android 推荐模块能够正常进行异常界面推荐和未覆盖界面的推荐
测试步骤	1. 测试人员点击异常界面推荐按钮，观察是否根据用户进行异常信息推荐 2. 测试人员点击未覆盖界面推荐按钮，观察是否有推荐信息以及推荐信息是否属于当前界面
测试结果	1. 点击异常界面推荐按钮后，系统根据用户测试情况进行 TopN 异常信息推荐 2. 未覆盖界面推荐按钮后，系统根据当前用户所处窗口位置，进行未覆盖界面推荐

表5.6展示了路径引导模块的测试过程与测试结果。其主要对应表3.10描述的用例，进行对异常界面推荐引导和未覆盖界面推荐引导的测试。此项测试用例覆盖了该模块所有的功能。

表 5.6: 路径引导模块测试

ID	**TC5**
名称	路径引导模块测试
测试功能	路径引导模块能够将用户从当前页面引导到异常界面或未覆盖界面
测试步骤	1. 测试人员点击异常界面推荐或未覆盖界面推荐，并选择其中一个推荐，点击确定按钮 2. 测试人员点击引导按钮，根据引导界面的提示进行操作，进入下一页面，并重复此操作 3. 观察引导信息是否能够正确引导
测试结果	1. 在点击推荐按钮后，能够根据当前所处窗口位置进行下一界面的操作引导 2. 当到达目标窗口时，系统会进行提示，要求用户进行异常复现并提交测试报告

表5.7展示测试结果上报模块的测试过程与测试结果。其主要对应表3.11描述的用例，进行异常界面复现结果上报以及主动点击上报按钮进行上报的测试。此项测试用例覆盖了该模块所有的功能。

表 5.7: 测试结果上报模块测试

ID	TC6
名称	测试结果上报模块测试
测试功能	测试结果上报模块测试能够正常进行测试结果上报
测试步骤	1. 测试人员选择异常推荐，并执行路径推荐，重复进行路径推荐，直到复现异常 2. 测试人员根据提示进行异常描述，之后点击确定按钮 3. 测试人员主动发现异常，并点击测试结果上报按钮，描述异常信息，并点击提交按钮
测试结果	1. 在重复执行路径推荐后，能够弹出对话框要求用户进行异常描述，并在点击确定后把数据发送至服务端 2. 在点击测试结果上报按钮后，能够对当前屏幕进行截图，在点击提交按钮后，能够将截图和信息描述一同发送至服务端

5.3　非功能测试

5.3.1　性能测试

软件性能测试是检测系统性能瓶颈, 提升软件质量的重要手段 [45]，本系统需要提供给众包工人进行使用，众包测试中并发量高，因此必须在使用系统前对系统的性能进行测试。这里使用工具 Jmeter[46] 对异常界面推荐接口、未覆盖界面推荐接口、路径引导接口和测试结果上报接口进行压力测试。通过对 Jmeter 参数进行设置，压力测试并发线程数为 200,1 秒内请求次数为 10，并发量为 2000，各接口的压力测试结果如表5.7所示。

对于异常界面推荐接口，平均响应时间为 1065ms，最小响应时间为 5ms, 最大响应时间为 7941ms, 异常率为 0%。图5.1展示了该接口每秒事务处理数，可以看出，大部分时间每秒钟处理数都在 200 左右，满足了并发需求。

表 5.8: 接口压力测试汇总表

接口	并发量	平均值	最小值	最大值	异常率	吞吐量
异常界面推荐	2000	1065	5	7941	0%	155.2/sec
未覆盖界面推荐	2000	600	4	2941	0%	524.9/sec
路径引导	2000	467	3	3017	0%	325.4/sec
测试结果上报	2000	862	6	6422	0%	562.4/sec

对于异常界面推荐接口，平均响应时间为 1065ms，最小响应时间为 5ms, 最大响应时间为 7941ms, 异常率为 0%，吞吐量为 155.2/sec。图5.1展示了该接口每秒事务处理数，可以看出，大部分时间每秒钟处理数都在 200 左右，满足了并发需求。

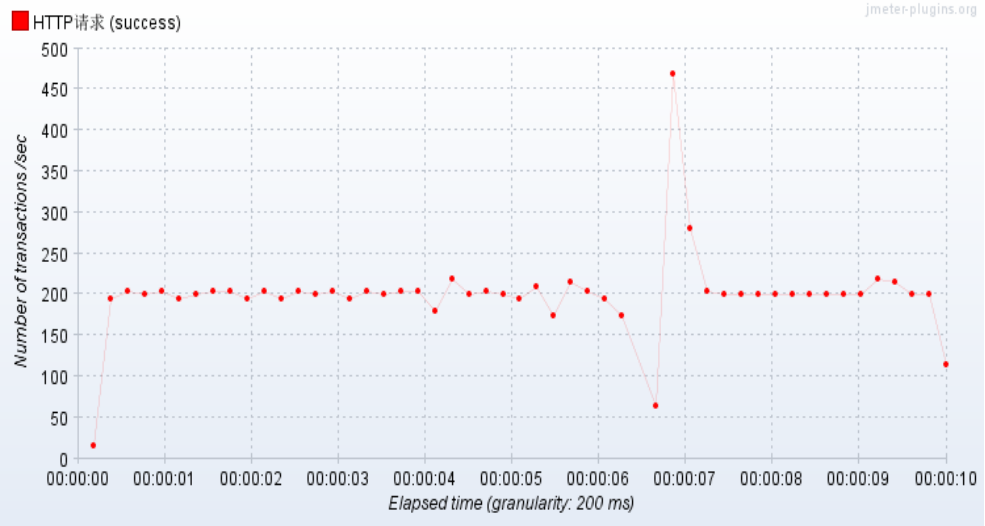

图 5.1: 异常界面推荐接口响应时间图

对于未覆盖界面推荐接口，平均响应时间为 600ms，最小响应时间为 4ms, 最大响应时间为 2941ms, 异常率为 0%, 吞吐量为 524.9/sec。图5.2展示了该接口每秒事务处理数，虽然在 2s-6s 之间，事务处理量有浮动，但浮动幅度不大，且仍在 200 左右，能够满足需求。

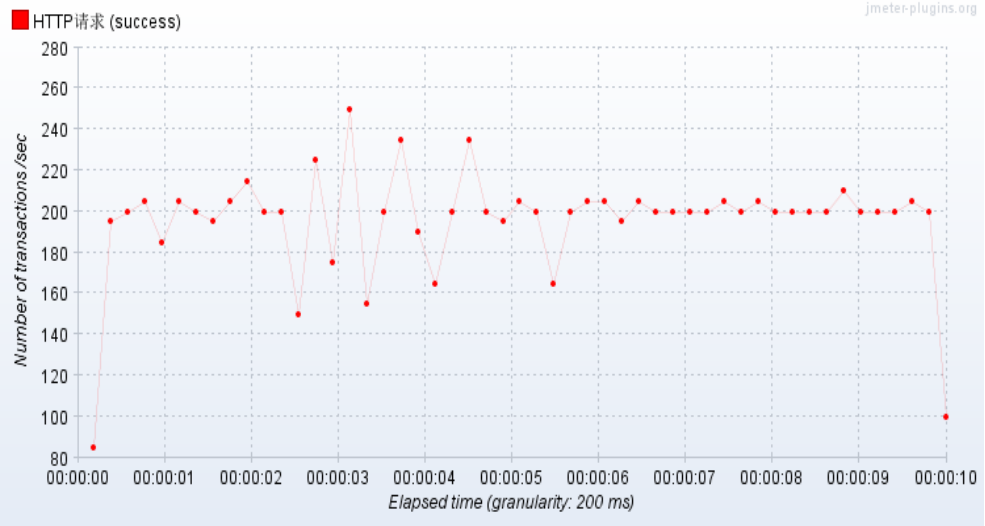

图 5.2: 未覆盖界面推荐接口响应时间图

对于路径引导接口，平均响应时间为 467ms，最小响应时间为 3ms, 最大响应时间为 3017ms, 异常率为 0%, 吞吐量为 325.4/sec。图5.2展示了该接口每秒事务处理数, 在整个测试过程中，接口只会偶尔出现轻微浮动，仍然满足每秒 200

个事务处理的能力，能够满足需求。

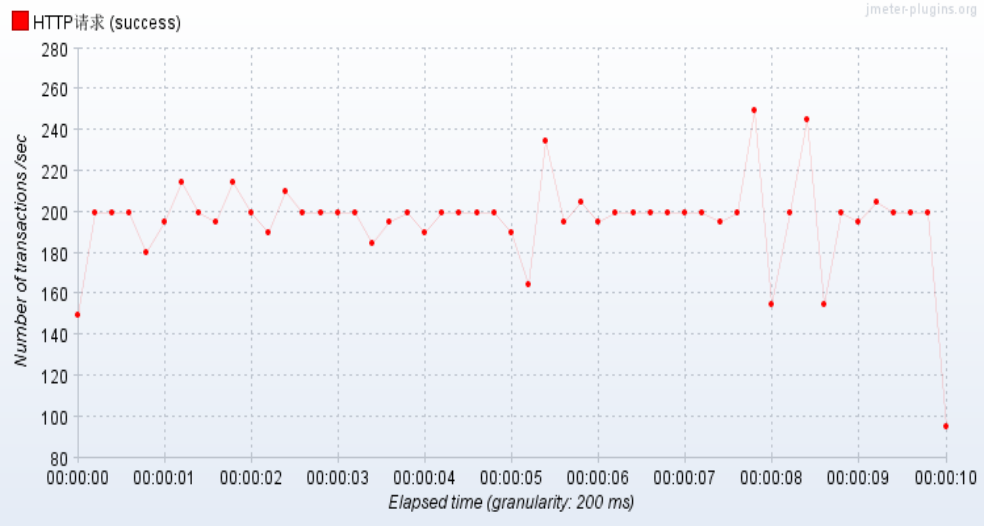

图 5.3: 路径引导接口响应时间图

对于测试结果上报接口，平均响应时间为 862ms，最小响应时间为 6ms, 最大响应时间为 6422ms, 异常率为 0%, 吞吐量为 562.4/sec。图5.2展示了该接口每秒事务处理数, 在整个测试过程中，接口处理事务能力浮动较大，在 2s-7s 的时间内处理事务数量整体低于 200，在 7s-9s 内处理事务的能力总体高于 200，考虑到该接口需要进行文件传输，因此该接口处理事务数量满足需求。

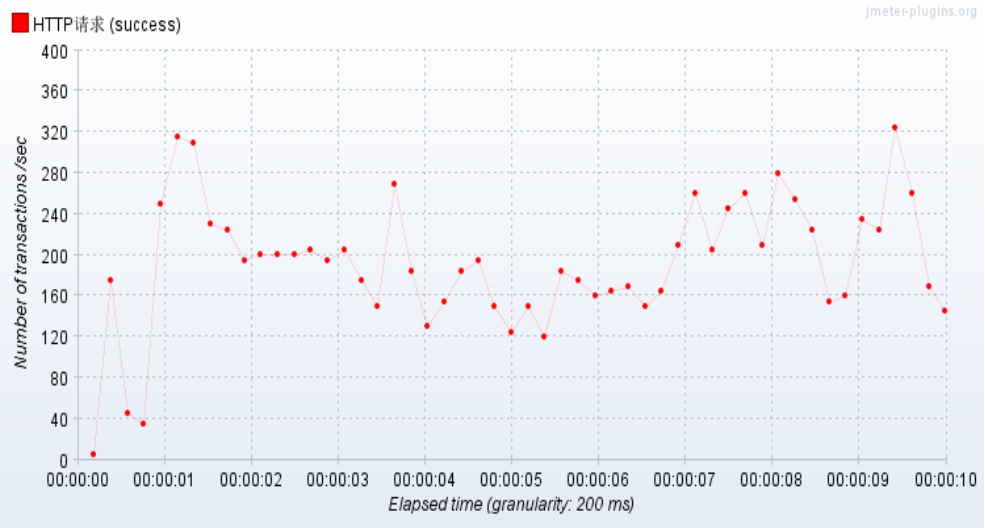

图 5.4: 测试结果上报接口响应时间图

5.4　系统分析

5.4.1　系统测试目标

为了验证面向移动应用众测的人机协同引导技术的可用性和有效性，需要通过实际的众包测试试验来进行验证。通过接下来的测试，本文需要解决的问题主要有两个：

（1）通过对 Android APP 进行自动化测试结果进行解析，提取异常信息交由众包工人验证，在验证过程中对工人加以引导，是否能够提高众包测试的速度和效率

（2）通过对 Android APP 进行静态分析，找出自动化测试未覆盖页面，通过引导众包工人进入未覆盖页面进行众包测试，能否提高窗口测试覆盖率

5.4.2　系统测试设置

本系统在慕测测试平台进行测试，系统共选用 10 款开源 Android APP，应用类型覆盖工具、娱乐、社交等多种类型，其源码均可以在 GitHub 中获取，应用详情如表5.9所示。众包工人是随机选取的 10 名学生，测试机型选用的当前市场中较为通用的 Android 手机，其详情如表5.10所示。

表 5.9: Android APP 详情汇总表

ID	名称	版本号	类型	代码行数
1	云阅	v1.1.1	新闻	38174
2	哔哩哔哩	v2.3.1	娱乐	50139
3	IT 之家	v1.0.1	新闻	15798
4	知乎	v1.0.2	社交	8603
5	滴滴	v1.0	工具	22171
6	手机图库	v0.7-alpha-2	工具	35507
7	热亡	v1.0.11	娱乐	12330
8	RGB 工具	v1.3.1	工具	12170
9	我的日记	v1.67	工具	11199
10	随身天气	v1.0.0	工具	5539

表 5.10: Android APP 详情汇总表

ID	品牌	型号	Android 版本	网络类型
1	VIVO	iQOO neo3	10	4G
2	VIVO	S7	9	4G
3	小米	Redmi9	9	4G
4	小米	10	10	4G
5	小米	Note9	10	4G
6	华为	P40	10	4G
7	华为	Mate40 pro	10	4G
8	荣耀	9X	8	4G
9	荣耀	20	9	4G
10	OPPO	K7X	8	4G

完整的系统测试流程如下：

（1）测试人员对 Android APP 自动化测试结果进行解析，将异常信息和自动化测试路径信息存储到数据库

（2）测试人员执行静态结果分析，将未覆盖窗口信息存储到数据库

（3）给待测 Android APP 嵌入引导模块，重新打包编译生成 APK 文件，由进行众包测试的学生进行安装

（4）学生在了解插件的基本使用后，开始对 Android APP 进行测试，测试时长为 20 分钟。

为了更加详细的展示人机协同的作用，本文设置四组不同的系统测试：

（1）既无异常界面和未覆盖界面推荐, 又无测试过程引导，即传统众包测试

（2）有异常界面和未覆盖界面推荐，无测试过程引导

（3）有异常界面和未覆盖界面乱序推荐，有测试过程引导

（4）既有异常界面和未覆盖界面推荐，又有引导

对于进行第二、三、四组测试的学生，在使用的过程中均会使用到 Android 引导模块的引导，接下来将详细介绍引导的过程及步骤。

图5.5是异常界面引导步骤。首先，点击异常界面推荐按钮，在推荐测试任务中选择一个任务，作为接下来进行测试的目标对象。然后，点击路径引导按钮弹出推荐路径，根据推荐路径中图片和文字提示，点击当前页面中的提示位置进入下一界面，并重复此过程。之后，系统会弹出对话框，表明已经完成此任务的测试，通过点击报告异常按钮，来填写此次测试任务中出现的异常，最后点击提交按钮进行提交。

图5.6是未覆盖界面引导步骤。首先，点击未覆盖页面推荐按钮，在未覆盖窗口推荐列表中选择一个路径，作为接下来进行测试的目标对象。然后，点击引导按钮，弹出未覆盖事件的引导窗口，根据提示进入目标界面。目标界面为自动化测试未覆盖的界面，该界面需要通过众包工人进行测试进行异常发现，即通过人工来弥补自动化测试的不足。在发现异常后，通过点击测试上报按钮，会对当前界面进行截图，在按照要求填写异常描述后，即可提交测试报告。

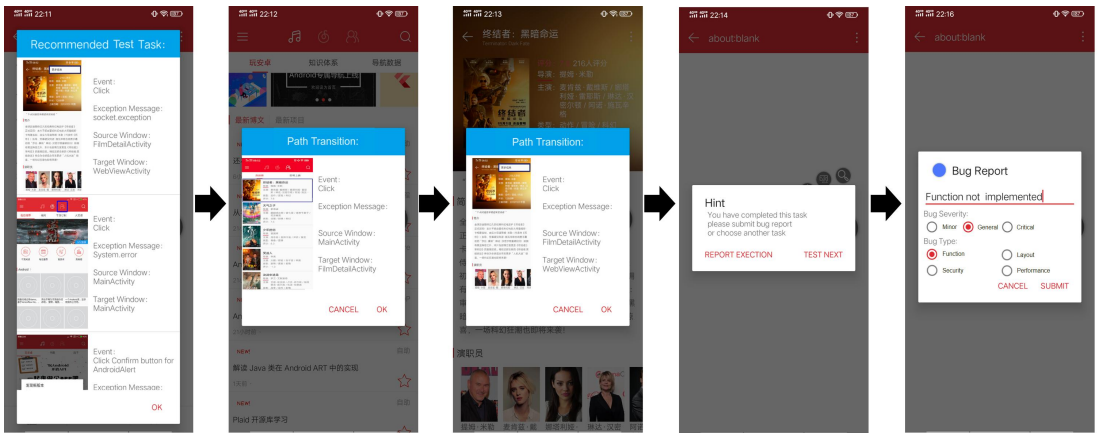

图 5.5: 异常界面引导步骤

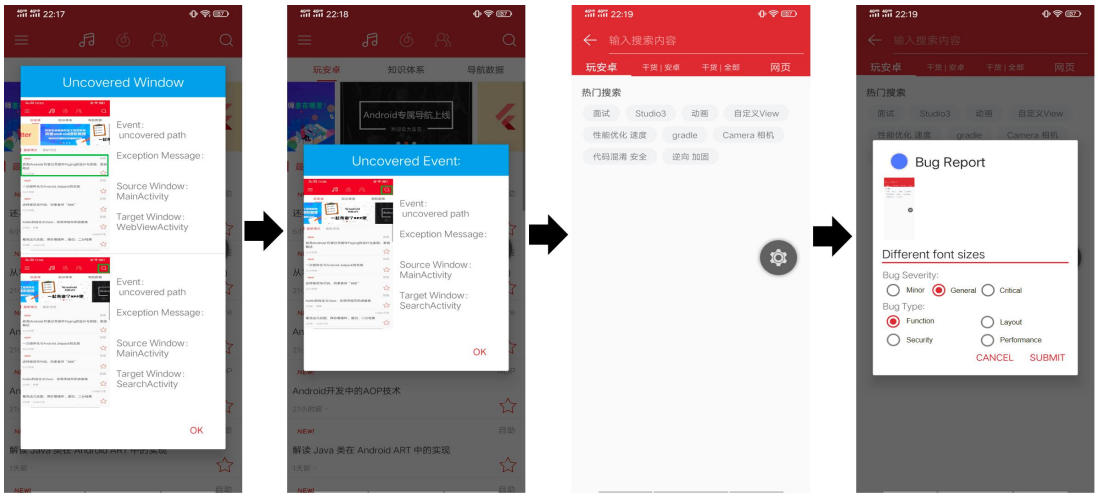

图 5.6: 未覆盖界面引导步骤

5.4.3　系统测试结果

图5.7为上述四组的系统测试结果。结果展示图左边部分为该 Android APP 在四组测试中得到的未覆盖窗口测试的异常数量和异常推荐测试的异常数量的

柱状统计图，图中通过四种测试方式的对比，对系统在找到异常数量的有效性方面进行了直观的展示。图右边部分为众包工人在单位时间内找到异常数量的折线统计图，该图从另一个角度展示了系统在异常复现以及从未覆盖界面发现异常的高效率，

从整体上看，第一组系统测试，即既无异常界面和未覆盖界面推荐又无测试过程引导，在总测试时长内找到异常数量明显低于其他三组。同时，在单位时间内找到异常的数量也低于其他三组测试。第二组系统测试，即有异常界面和未覆盖界面推荐，无测试过程引导，在测试异常数量以及速度上略低于第三、第四组测试，但是高于第一组测试，说明任务推荐能在一定程度上帮助众包工人高效率找到异常或者未覆盖界面。第三组系统测试，即有异常界面和未覆盖界面乱序推荐，有测试过程引导，在测试异常数量与第四组测试相差不大，但在测试速度上，略低于第四组测试，说明引导工具能够帮助众包工人快速到达目标界面并进行异常复现，但由于推荐的方式是乱序推荐，众包工人对任务的敏感度不同，使得其异常测试效率略低于第四组测试。第四组系统测试，即既有异常界面和未覆盖界面推荐，又有引导，在测试异常数量和测试速度上，都明显高于其他三组测试，是测试效果最好的一组。

从上述测试结果中可以得知，通过对自动化测试结果进行异常推荐，对静态分析结果进行未覆盖界面推荐，并通过路径引导的方式帮助众包工人进行测试，能有效提高众包工人在测试异常数量以及测试异常的速度。从左边的统计图中我们可以看出，对于个别 Android APP 在四组测试过程中得到未覆盖窗口数和异常数量相差不大，其原因是这些应用都是体积较小的应用，窗口界面比较少，应用功能比较简单，使得其对异常界面推荐和未覆盖页面推荐以及引导具有较低的依赖性。而对于界面复杂，功能多样的应用，任务推荐和路径引导能够发挥更大的作用。

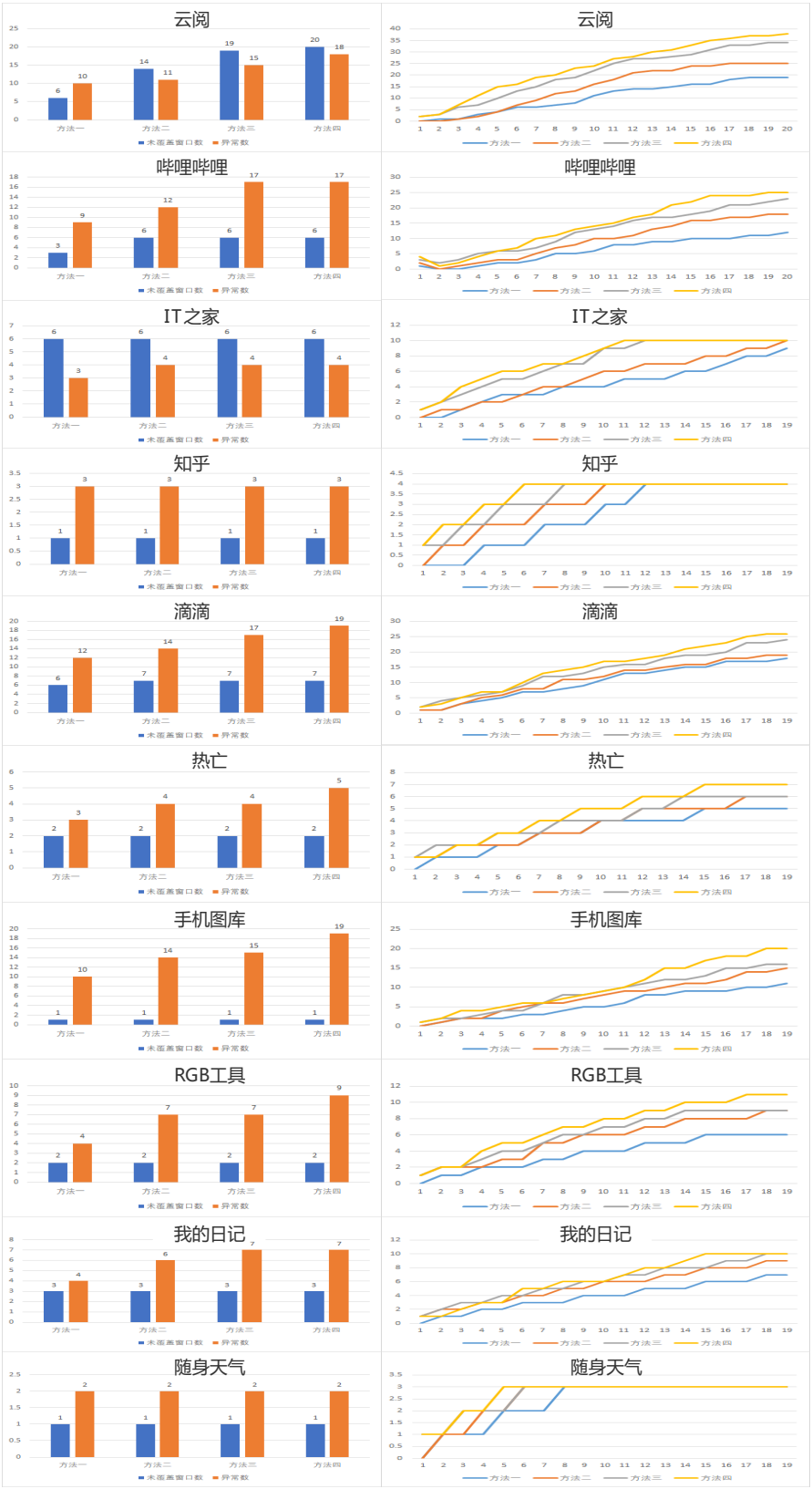
云阅
哔哩哔哩
IT之家
知乎
滴滴
热亡
手机图库
RGB工具
我的日记
随身天气
方法一
方法二
方法三
方法四
未覆盖窗口数
异常数

图 5.7: 系统测试结果

图5.8是众包测试过程中提交的异常类型报告统计图。异常类型主要分为四大类，包括功能异常、布局异常、安全异常和性能异常。从总体上看，异常主要集中在功能异常和布局异常两个方面，说明该系统对于测试这两种异常具有比较好的效果。

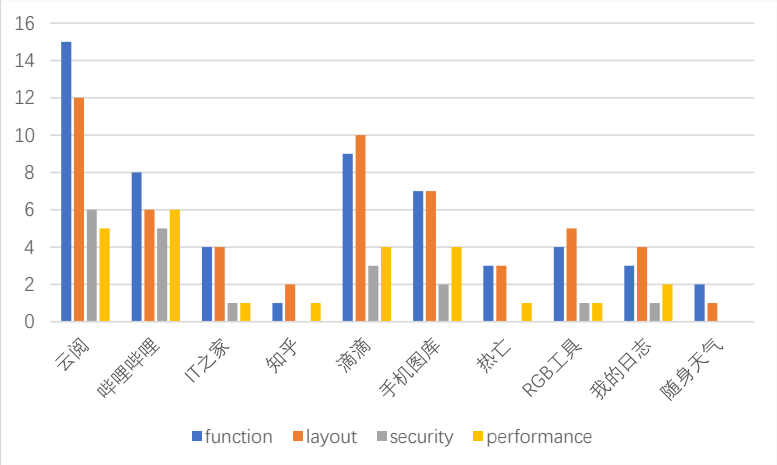

图 5.8: 异常类型统计

表5.11是十款 Android APP 的窗口总数以及分别对其进行四种不同形式的系统测试得到的窗口覆盖数目。图5.9是对其窗口覆盖数目和其总窗口数进行对比获得的窗口覆盖率。

从表中可以看出，从测试方式一至测试方式四，窗口覆盖数目总是保持平稳或者上升，从图中可以看出，窗口覆盖率也是保持这种趋势。首先对于 Bihu、HotDeath 和 Weather 这三款移动应用，其窗口覆盖率在四组测试中均达到 100%，其原因我们可以从表中看出，因为它们窗口总数都是 3，说明其窗口量少，众包工人在不借助引导工具的情况下，也能够通过自己的能力找到所有的窗口，进行测试。而对于其他移动应用，比如 Yunyue、Bilibili 和 DidiCar 等移动应用，因为其窗口总数较多，单独通过传统众包测试很难对所有的窗口路径进行测试，因此测试方式一窗口覆盖率比较低。通过与其他组系统测试进行对比可以发现，其窗口覆盖率在有窗口推荐的情况下有明显增高，在有路径引导的情况下又进一步提升，说明该系统能有效提高窗口覆盖率。

表 5.11: 窗口覆盖数目

APP 名称	窗口总数	窗口覆盖数目			
		测试一	测试二	测试三	测试四
云阅	31	23	26	30	31
哔哩哔哩	42	36	39	42	42
IT 之家	18	17	17	18	18
知乎	3	3	3	3	3
滴滴	18	12	13	18	18
手机图库	14	8	10	11	14
热亡	3	3	3	3	3
RGB 工具	6	4	5	6	6
我的日记	6	5	6	6	6
随身天气	3	3	3	3	3

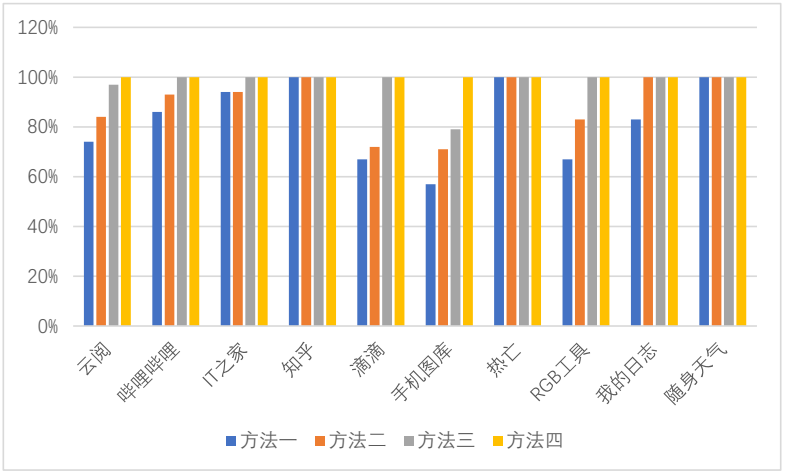

图 5.9: 窗口覆盖率

根据以上测试结果，现对系统测试目标进行回答。

（1）通过对自动化测试结果进行解析，提取异常信息交由众包工人验证，在验证次数达到一定次数后就不在对异常进行推荐，可以减少众包工人对已经经过自动化测试的事件进行多次重复测试的行为，且通过系统引导，减少众包工人自己摸索到达目标窗口验证异常的时间，因此有效地提高了众包测试的速度和效率，达到了系统测试目标。

（2）通过对 Android APP 进行静态分析，找出自动化测试未覆盖页面，在通过引导的方式引导众包工人进入未覆盖界面，即先通过系统进行筛选，再进行

路径推荐，使众包工人能明确未覆盖界面位置。众包工人进入未覆盖界面进行测试，以人工的方式弥补了自动化测试未覆盖的异常，有效地提高了窗口覆盖率，达到了系统目标。

5.5　本章小结

本章主要是对系统进行测试与结果分析。先介绍了系统开发使用的工具以及系统开发所用的环境，然后对系统的各个功能模块进行功能测试，确保了各模块方法的正确性。之后对系统的性能，主要是系统的并发量以及系统处理事务的能力进行了测试。最后是对测试进行分析，通过展示系统测试结果，并对测试进行分析，证明了系统的可用性和有效性。

第六章　总结与展望

6.1　总结

随着 Android 操作系统的市场占有率增高，Android 移动应用的数量每年以极高的速度在增加，这同时促进了 Android 移动应用测试的发展。但是，由于 Android 操作系统版本多种多样且更新迭代速度快，Android 手机厂商基于 Android 操作系统进行闭源再开发，导致现有的测试方案不能很好地解决 Android 移动应用测试的问题。在此背景之下，本文提出了一种面向移动应用众测的人机协同引导技术，通过融合自动化测试和众包测试的优点，提供了进行 Android 移动应用测试的另一种解决方案。

本文首先对国内外自动化测试技术和众包测试技术进行了调研分析，接着介绍了系统用到的相关技术。Android 端主要是通过 OKHTTP 来处理网络请求，通过 CrashHandle 来处理系统突发性崩溃事件，通过 FloatingActionButton 来设计悬浮按钮。后端主要通过 Spring Boot 框架来开发和部署服务端代码，并通过使用 MyBatis 来实现业务功能。数据库使用 MariaDB，确保系统数据的安全性和可靠性。系统通过使用 GitLab 来对代码进行托管，并通过 Git 来进行版本控制。

系统的涉众主要包括测试需求方和众包工人两个方面，通过对涉众进行分析，明确了系统的功能需求，包括自动化测试数据处理、静态分析处理、用户登录、异常界面推荐、未覆盖界面推荐、路径推荐和测试结果上报等功能。同时，对于非功能需求，包括可靠性、可用性、安全性、易用性和可维护性等，也是系统开发过程中不可忽视的部分。本文通过系统用例图，以表格的方式对系统设计的用例进行了详细地描述，然后通过系统架构图、4+1 视图模型即逻辑视图、开发视图、进程视图、物理视图和场景视图等多个角度对系统进行了进一步的分析。为了更加详细的描述各模块的设计与实现，本文从类图、顺序图和关键代码等几个角度对模块进行了进一步说明其具体实现细节。

最后，本文先介绍了系统开发所用的工具以及环境配置，然后通过对系统各功能模块进行功能测试与非功能测试，确保了各模块的可用性和稳定性，然后设计四组对比实验，对十款类型不同的 Android 移动应用进行了测试。本文对测试过程中提交的实验结果，从异常提交数量、异常提交速度、异常提交类型以及窗口覆盖率等多个角度进行分析，证明系统在提高众包工人众包测试速度和效率以及提高窗口覆盖率方面具有明显的效果。

6.2 展望

目前，本文设计的面向移动应用众测的人机协同引导技术已经部署于服务器上使用，为进一步提高系统的效果，未来将根据以下几个方面对系统进行改进。

（1）本文的引导方式是根据众包工人当前所在的窗口位置进行实时的单步路径推荐，当路径比较复杂时，这种单步路径推荐方式就比较耗时。之后可以让用户选择路径步骤推荐的个数，更加个性化进行推荐，以便提高测试效率。

（2）本系统的 Android 引导模块通过在 Android APP 的源码中嵌入引导模块代码才能使用，这影响了 Android APP 的封装性和安全性，后续可以考虑通过以包的形式对其提供支持，实现可插拔效果。

（3）本文的推荐方式只能对用户进行较为粗略的个性化推荐，没有完全实现针对具体品牌、具体系统版本以及众包工人的测试倾向进行精确度更高的推荐。之后将扩大机型种类和系统版本范围，获得更多的数据，再通过模型训练，进行更加准确更加个性化的推荐，提高用户体验。

参考文献

[1] Statcounter, 全球移动操作系统市场份额 (2021).
URL https://gs.statcounter.com/os-market-share/mobile/worldwide

[2] 柳晓华, 移动应用自动化测试策略的研究与应用, Ph.D. thesis (2017).

[3] Y. D. Lin, J. F. Rojas, T. H. Chu, Y. C. Lai, On the accuracy, efficiency, and reusability of automated test oracles for android devices, IEEE Transactions on Software Engineering 40 (10) (2014) 957–970.

[4] 曹羽中、吴国全、陈伟、魏峻、黄涛、王溯, 一种基于录制/重放的 android 应用众包测试方法, 软件学报 v.31 (08) (2020) 230–251.

[5] D. Amalfitano, A. R. Fasolino, P. Tramontana, B. Ta, A. Memon, Mobiguitar – a tool for automated model-based testing of mobile apps, IEEE Software (2014) 1–1.

[6] H. Zheng, D. Li, B. Liang, X. Zeng, W. Zheng, Y. Deng, W. Lam, W. Yang, T. Xie, Automated test input generation for android: towards getting there in an industrial case, in: 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP), 2017.

[7] 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: the 13th International Workshop, 2018.

[8] 兰娅勋, 李振坤, Monkeyrunner 环境用例脚本化的 android 软件测试模型, 科技通报 33 (009) (2017) 96–99.

[9] Zadgaonkar, Robotium Automated Testing for Android, Packt Publishing, 2013.

[10] 刘会娟, 基于 uiautomator 的 mtbf 自动化测试工具案例的设计与实现, Ph.D. thesis, 山东大学 (2017).

[11] 蒋子豪, 基于 appium 的移动端自动化测试项目的设计与实现, Ph.D. thesis (2016).

[12] D. Vajak, R. Grbi, M. Vranje, D. Stefanovi, Environment for automated functional testing of mobile applications, in: 2018 International Conference on Smart Systems and Technologies (SST), 2018.

[13] J. Howe, Crowdsourcing: Why the Power of the Crowd Is Driving the Future of Business, Crown Publishing Group, 2010.

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

[15] 谭婷婷, 蔡淑琴, 胡慕海, 众包国外研究现状, 武汉理工大学学报: 信息与管理工程版 (02) (2011) 263–266.

[16] Y. H. Tung, S. S. Tseng, A novel approach to collaborative testing in a crowdsourcing environment, Journal of Systems Software 86 (8) (2013) 2143–2153.

[17] G. Bano, Q. Ali, S. S. Khuwaja, I. Farah, A. Zubedi, Comparative analysis of mobile application testing and crowd source software testing, in: 2019 8th International Conference on Information and Communication Technologies (ICICT), 2019.

[18] 张守刚, 刘海波, 人工智能的认识论问题, 人工智能的认识论问题, 1984.

[19] J. Krueger, Human-Machine Collaboration, Human-Machine Collaboration, 2014.

[20] 刘杰, 韩烨, 陈剑锋, 毛得明, 饶志宏, 一种基于人机协同的软件漏洞模糊测试方法 (2020).

[21] 代杰, 邹仁平, Network request method and device based on okhttp and gson (2016).

[22] A. K. Jha, W. J. Lee, Capture and replay technique for reproducing crash in android applications, in: The 12th IASTED International Conference on Software Engineering, 2013.

[23] M. H. Nasri, R. Hartanto, A. E. Permanasari, N. Arfian, The user experience effect of applying floating action button (fab) into augmented reality anatomy cranium media learning prototype, in: 2020 3rd International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), 2020.

[24] 田兵, 王玮, 苏琦, 刘荫, 盛健荣, 王永根, 基于微服务架构的应用性能监控平台研究, 信息技术与信息化 01 (2018) 125–128.

[25] A. Ciuffoletti, Automated deployment of a microservice-based monitoring infrastructure, in: K. G. Jeffery, D. Kyriazis (Eds.), 1st International Conference on Cloud Forward: From Distributed to Complete Computing, October 6-8, 2015, Pisa, Italy, Vol. 68 of Procedia Computer Science, Elsevier, 2015, pp. 163–172.

[26] J. D. Blischak, E. R. Davenport, W. Greg, O. Francis, A quick introduction to version control with git and github, Plos Computational Biology 12 (1) (2016) e1004668.

[27] H. Suryotrisongko, D. P. Jayanto, A. Tjahyanto, Design and development of backend application for public complaint systems using microservice spring boot, Procedia Computer Science 124 (2017) 736–743.

[28] 温晓丽, 苏浩伟, 陈欢, 邹大毕, 基于 springboot 微服务架构的城市一卡通手机充值支撑系统研究, 电子产品世界 (10) (2017) 59–62.

[29] Y. Guo, M. Chen, K. Wei, Research of recycling resource website based on spring and mybatis framework, Vol. 455 of Advances in Intelligent Systems and Computing, 2015, pp. 307–314.

[30] 陈小虎, 邓惠俊, 基于 mybatis 的数据持久层研究, 成都电子机械高等专科学校学报 023 (002) (2020) 29–32.

[31] 阳小兰, 罗明, 基于 spring+springmvc+mybatis 网上论坛的设计与实现, 科学技术创新 (36) (2016) 279–280.

[32] J. Lindström, D. Das, N. Piggin, S. Konundinya, T. Mathiasen, N. Talagala, D. Arteaga, An NVM aware mariadb database system and associated IO workload on file systems, Open J. Databases 4 (1) (2017) 1–21.

[33] S. Yang, H. Zhang, H. Wu, Y. Wang, D. Yan, A. Rountev, Static window transition graphs for android (T), IEEE Computer Society, 2015, pp. 658–668.

[34] 孙冬婷, 何涛, 张福海, 推荐系统中的冷启动问题研究综述, 计算机与现代化 05 (2012) 59–63.

[35] 罗辛, 欧阳元新, 熊璋, 袁满, 通过相似度支持度优化基于 k 近邻的协同过滤算法, in: Ndbc 中国数据库学术会议, 2010.

[36] B. Jiang, J. Yang, Y. Qin, T. Wang, M. Wang, W. Pan, A service recommendation algorithm based on knowledge graph and collaborative filtering, IEEE Access 9 (2021) 50880–50892.

[37] 自建连, An improved similarity measurement method in collaborative filtering algorithm, Pure Mathematics 10 (5) (2020) 404–413.

[38] S. Sengupta, S. Bhattacharya, Formalization of uml use case diagram-a z notation based approach, in: International Conference on Computing Informatics, 2009.

[39] H. Byelas, A. Telea, Visualization of areas of interest in software architecture diagrams, in: Proceedings of the ACM 2006 Symposium on Software Visualization, Brighton, UK, September 4-5, 2006, 2006.

[40] 陆远蓉, 基于 uml 的”4+1” 视图软件体系结构描述研究, 现代计算机 (专业版) (4) (2011) 27–30.

[41] A. Russell, N. Chevallier, Representing a project's physical view in support of project management functions, Canadian Journal of Civil Engineering 25 (4) (2011) 705–717.

[42] 柳丹, 陈志刚, 雷卫军, 基于 uml 描述的”4+1” 视图模型及应用, 计算技术与自动化 20 (4) (2001) 46–46.

[43] K. Krocz, O. Kizun, M. Skublewska-Paszkowska, Perfomance analysis of relational databases mysql, postgresql, mariadb and h2, Journal of Computer Sciences Institute 14 (2020) 1–7.

[44] P. Piwowarski, M. Ohba, J. Caruso, Coverage measurement experience during function test, in: International Conference on Software Engineering, 1993.

[45] 孟祥丰, 软件性能测试解析与优化, 电子设计工程 (19) (2011) 26–28.

[46] R. Abbas, Z. Sultan, N. Shahid, S. B. Nazir, Comparative analysis of automated load testing tools: Apache jmeter, microsoft visual studio (tfs), loadrunner, siege, in: Internation Conference on Communication technologies COMTECH-2017, 2017.